FUJITSU

FUJITSU Software
Interstage Interaction Manager V10.1.2

Ajaxフレームワーク ユーザーズガイド

Windows/Solaris/Linux

J2UL-1573-05Z0(00)
2016年4月

まえがき

本書の目的

本書は、Ajaxフレームワークを利用してWebアプリケーションを開発するために必要な情報を記載しています。

本書の目的は、以下のとおりです。

- Ajaxフレームワークの機能を理解できること
- Ajaxフレームワークを利用してWebアプリケーションを開発できること

本書の読者

本書は、以下の読者を対象としています。

- Ajaxフレームワークを利用してWebアプリケーションを開発する人
- Ajaxフレームワークを利用してWebシステムを構築する人

前提知識

本書を読むにあたっては、以下の知識が必要です。

- HTML/XHTML/JavaScript/Java/CSSに関する基本的な知識
- Webアプリケーションに関する基本的な知識
- Apcoordinatorに関する基本的な知識

本書の構成

本書の構成は以下のとおりです。

第1章 概要

Ajaxフレームワークの概要を説明しています。

第2章 クライアントフレームワーク

クライアントフレームワークについて説明しています。

第3章 通信フレームワーク

通信フレームワークについて説明しています。

第4章 マッシュアップフレームワーク

マッシュアップフレームワークについて説明しています。

第5章 アプリケーションの開発

Ajaxフレームワークを利用したWebアプリケーション(以降、Ajaxフレームワークアプリケーションと呼びます)の開発方法について、説明しています。

付録A 環境定義ファイル

Ajaxフレームワーク環境定義ファイルについて説明しています。

付録B マッシュアップ定義ファイル

マッシュアップ定義ファイルについて説明しています。

付録C JavaScript API一覧

Ajaxフレームワークアプリケーションで利用可能なJavaScript APIの一覧です。

付録D サンプルアプリケーション

Ajaxフレームワークを利用したサンプルアプリケーションについて説明しています。

付録E モックアップの作成

Ajaxフレームワークを利用した画面モックアップの作成方法について説明しています。

付録F Ajaxページエディタ

Ajaxページエディタについて説明しています。

付録G スクレイピングツール

スクレイピングツールについて説明しています。

付録H 互換情報

旧バージョンのAjaxフレームワークとの互換情報について説明しています。

付録I 開発資産の移行

旧バージョンの開発資産から本バージョンへの移行方法について説明しています。

付録J エラーメッセージ

Ajaxフレームワークのメッセージについて説明しています。

本書の表記

本書に記載されているマニュアルの名称は、以下のように省略して表記します。

正式名称	略称
Ajaxフレームワーク UI部品リファレンス	UI部品リファレンス

本書では、説明するうえで、以下に示す略称を使用する場合があります。

正式名称	略称
Windows(R) Internet Explorer(R) 9 Windows(R) Internet Explorer(R) 10 Windows(R) Internet Explorer(R) 11	Internet Explorer
Windows(R) Internet Explorer(R) 9 標準 Windows(R) Internet Explorer(R) 9 互換	Internet Explorer 9
Windows(R) Internet Explorer(R) 10 標準 Windows(R) Internet Explorer(R) 10 互換	Internet Explorer 10
Windows(R) Internet Explorer(R) 11 標準 Windows(R) Internet Explorer(R) 11 互換	Internet Explorer 11

そのほか、本書で使用しているマニュアル名称や記号などの表記については、「マニュアル体系と読み方」を参照してください。

本書のコメント

Internet Explorer 9の各モードについては、以下の組み合わせを対象としています。

	ブラウザモード	ドキュメントモード
Internet Explorer 9 標準	Internet Explorer 9	Internet Explorer 9 標準
Internet Explorer 9 互換	Internet Explorer 9 互換表示	Internet Explorer 7 標準

Internet Explorer 10の各モードについては、以下の組み合わせを対象としています。

	ブラウザモード	ドキュメントモード
Internet Explorer 10 標準	Internet Explorer 10	標準
Internet Explorer 10 互換	Internet Explorer 10 互換表示	Internet Explorer 7 標準

Internet Explorer 11のモードについては、以下の組み合わせを対象としています。

	ブラウザモード	ドキュメントモード
Internet Explorer 11 標準	Internet Explorer 11	Edge
Internet Explorer 11 互換	Internet Explorer 11 互換表示	Internet Explorer 7 標準

輸出管理規制

本書を輸出または第3者へ提供する場合は、お客様が居住する国および米国輸出管理関連法規などの規制をご確認のうえ、必要な手続きをおとりください。

商標

Microsoft、Active Directory、Internet Explorer、Windows、Windows Server、Windows Vistaは、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。

Linuxは、Linus Torvalds氏の日本およびその他の国における登録商標または商標です。

Red HatおよびRed Hatをベースとしたすべての商標とロゴは、Red Hat, Inc.の米国およびその他の国における登録商標または商標です。

OracleとJavaは、Oracle Corporation およびその子会社、関連会社の米国およびその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

そのほか、本書に記載されている会社名および製品名は、それぞれ各社の商標または登録商標です。

また、本書に記載されている会社名、システム名、製品名などには、必ずしも商標表示(TM・(R))を付記しておりません。

出版年月および版数

版数	マニュアルコード
2013年 3月 初版	J2UL-1573-01Z0(00)/J2UL-1573-01Z2(00)
2013年 8月 2版	J2UL-1573-02Z0(00)/J2UL-1573-02Z2(00)
2014年 5月 3版	J2UL-1573-03Z0(00)/J2UL-1573-03Z2(00)
2014年11月 4版	J2UL-1573-04Z0(00)/J2UL-1573-04Z2(00)
2015年 5月 4.1版	J2UL-1573-04Z0(01)/J2UL-1573-04Z2(01)
2016年 4月 5版	J2UL-1573-05Z0(00)/J2UL-1573-05Z2(00)

お願い

本書を無断で他に転載しないようお願いします。
本書は予告なしに変更されることがあります。

著作権

目　次

第1章 概要

本章では、Ajaxフレームワークの概要および特長について説明します。

1.1 Ajaxフレームワークとは

Ajaxフレームワークは、Ajaxを利用したWebアプリケーションを開発するためのフレームワークです。

ここでは、AjaxとAjaxフレームワークの概要を説明します。

Ajax

Ajaxとは、Asynchronous JavaScript＋XMLの略で、JavaScriptとXMLを使ってデータを非同期に通信する技術のことです。また、以下のような特長をもった、Webアプリケーションの開発手法でもあります。

- 運用容易性・レスポンス向上・操作性向上をバランス良く実現
  - Webと同等の管理容易性
  - 非同期通信による画面の部分更新によるレスポンス向上
  - JavaScriptを利用し、画面の操作性を向上
- オープン性
  - Webブラウザだけの機能で実現
  - 大手ベンダー/ISVに無依存
- Webシステムの自然な延長
  - 標準的なWebブラウザがあれば動作
  - 現状のWebシステムを拡張可能
- インターネットで急速に普及
  - Web2.0に沿ったサービスで採用

Ajaxフレームワーク

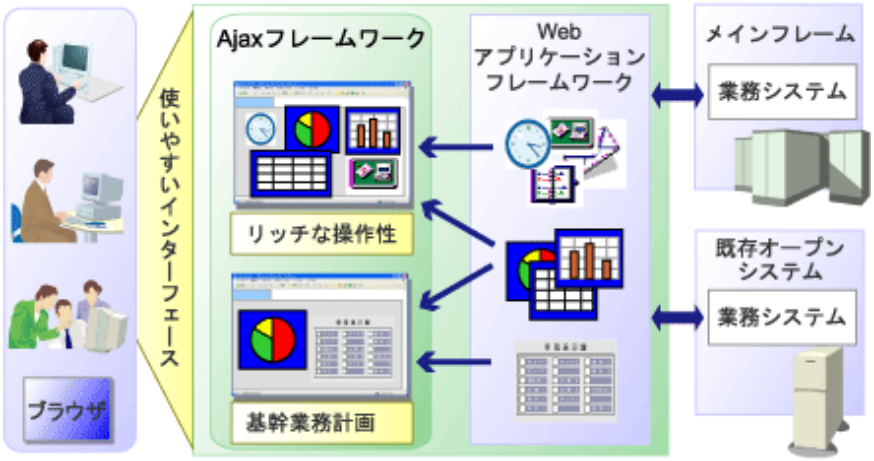
Ajaxを採用したAjaxフレームワークは、基幹Webシステムのクライアント業務の操作性、生産性の改善を実現します。

Ajaxフレームワークを利用することによって、以下のような効果があります。

- Webアプリケーションのユーザビリティ改善
  従来のWebシステムのような画面全体の書換えによってエンドユーザの作業が中断されることなく、レスポンスの良い操作/入力を実現することができます。
  また、業務に最適化された直感的で高品質なユーザーインターフェースを実現できます。
- クライアント運用コストの削減
  クライアントサーバやアプレット/Flashを利用したシステムから、ブラウザだけで動作するシステムへ移行することにより、クライアントへのアプリケーションやプラグインの初期導入・バージョンアップが不要になります。
- リッチWebアプリケーション開発生産性向上
  JavaScriptのフレームワークを利用することによって、開発作業の分業が容易になり、互いに仕様変更の影響を受けなくなるため、JavaScriptを利用した開発における保守性・再利用性が向上します。また、高機能・高品質な部品の利用により開発量が削減されます。

以下の図に、Ajaxフレームワークを利用したWebシステムの例を示します。

図1.1 Ajaxフレームワークを利用したWebシステム

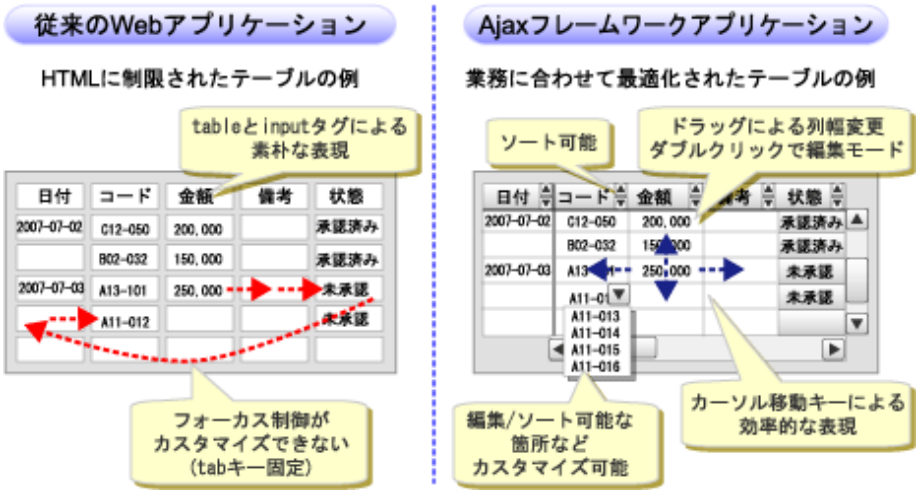


1.2 Ajaxフレームワークの特長

ここでは、Ajaxフレームワークの特長を説明します。

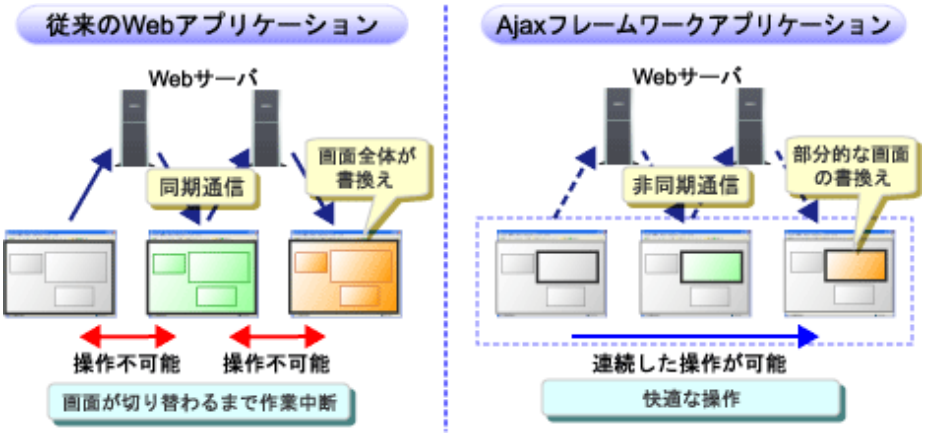
Webアプリケーションの操作性・画面表現力の向上

従来のWebアプリケーションは、HTMLで表現される画面部品群を利用して画面を開発していました。
Ajaxフレームワークは、HTMLで表現できる画面部品群に比べて、高機能、高品質、かつカスタマイズ容易な部品群を提供します。これにより、従来のHTMLでは実現困難だったレベルの操作性・画面表現が実現可能となり、Webアプリケーションのユーザビリティが大幅に向上します。



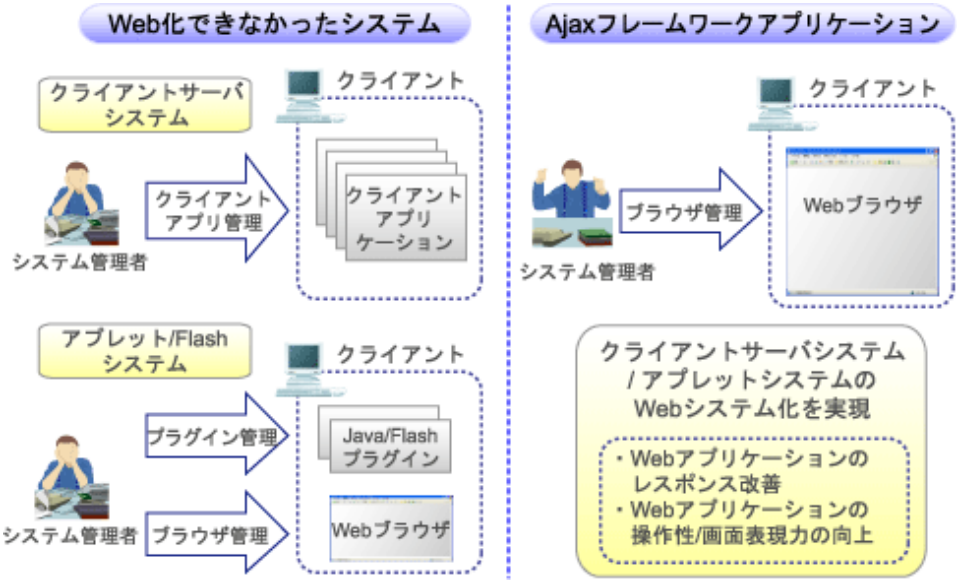
Webアプリケーションのレスポンス改善

従来のWebアプリケーションでは、画面が遷移すると、画面全体の書換えが発生し、書換えが完了するまで、操作を行うことができませんでした。
Ajaxフレームワークは、非同期通信と画面の部分書換えにより、操作できない時間を大幅に短縮して、Webアプリケーションのレスポンスを改善します。これにより、ユーザーに快適な操作性を提供し、データ入力業務などの作業効率化を実現します。



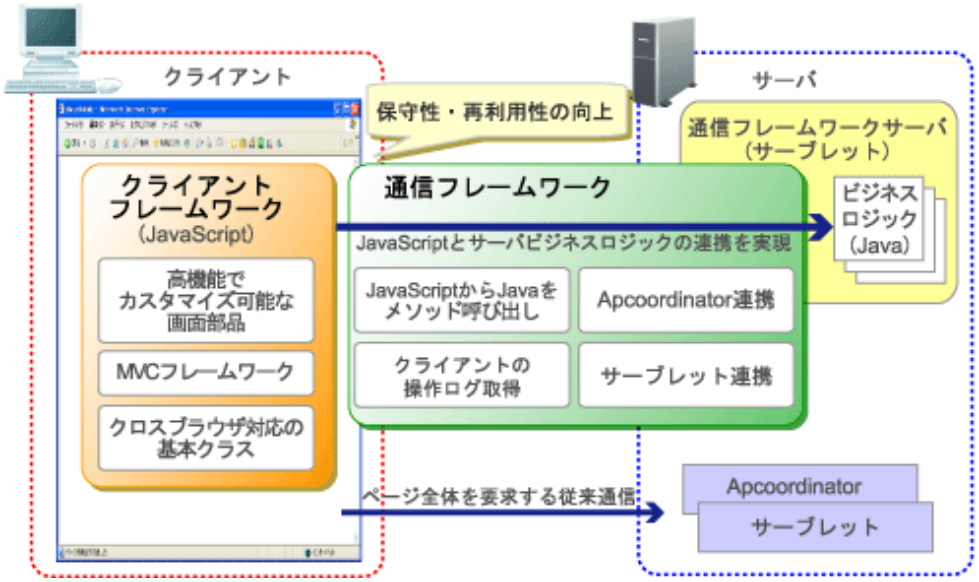
クライアント運用のコスト削減

従来のクライアントサーバシステムでは、システムの初期導入時やバージョンアップ時に、クライアントアプリケーションの配付やインストールなどを管理する必要がありました。また、アプレットやFlashを利用したシステムでも、ランタイム(ブラウザプラグイン)を管理する必要がありました。これらの管理工数を削減するためには、クライアントサーバシステムやアプレットシステムをWeb化する方法がありますす。しかし、高い操作性を求められるシステムは、Webの操作性が低いため移行できませんでした。
Ajaxフレームワークでは、ユーザビリティの向上により、これらのシステムをWebシステム化できるので、クライアントアプリケーションやJava/Flashプラグインなどのクライアント管理コストが不要になります。



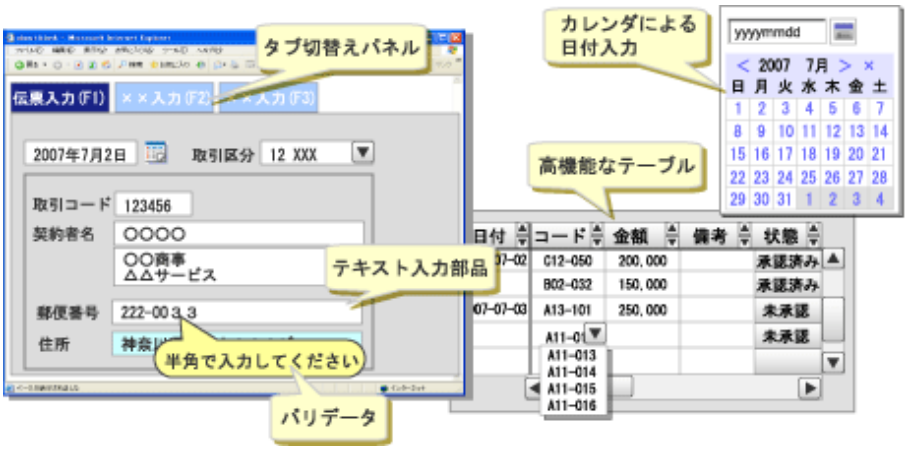
保守性・再利用性の向上

Ajaxフレームワークは、クライアントフレームワークと通信フレームワークを提供します。これらにより保守性・再利用性の高いシステムを短期間で構築できます。クライアントフレームワークは、高機能なHTML(JSP)+JavaScriptシステム開発を可能にします。通信フレームワークは、クライアント側データとサーバ側データの連携を容易に実現します。



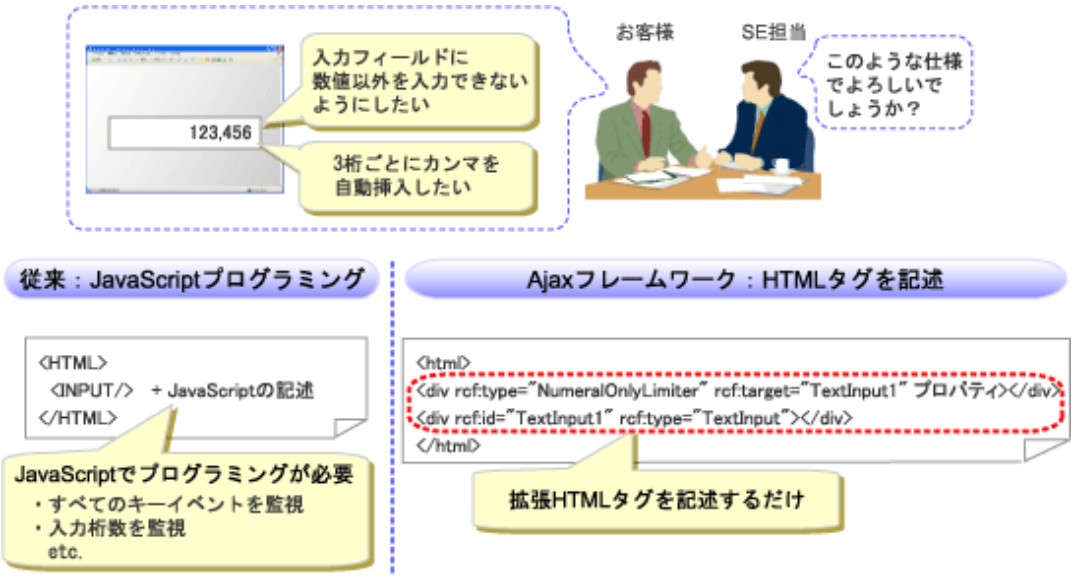
開発量の削減

Ajaxフレームワークは、タブ切替え可能なパネル、ソート/編集可能なテーブル、カレンダ日付入力部品、フォーカス制御、PFキーの利用、テキストフォーマッタ、テキスト入力バリデータなど、業務アプリケーション開発時に必要となるJavaScriptコードを部品として提供します。開発者が画面の随所で開発していた、低レベルなJavaScriptコードが不要になり、システム全体の開発量を大幅に削減できます。



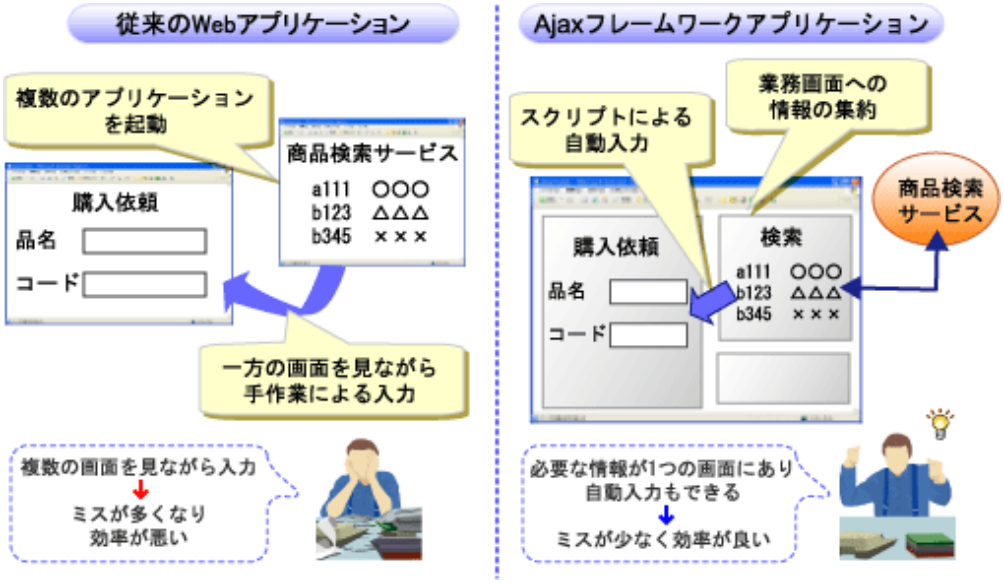
モックアップ作成の簡易化

顧客要件を確認するために業務システム画面のHTMLモックアップで、フォーカスの制御や、キー入力の制御などを行いたいといった要求があります。この場合、従来はJavaScriptによるプログラミングが必要でした。Ajaxフレームワークでは、HTMLを記述する感覚で、モックアップを作ることができます。



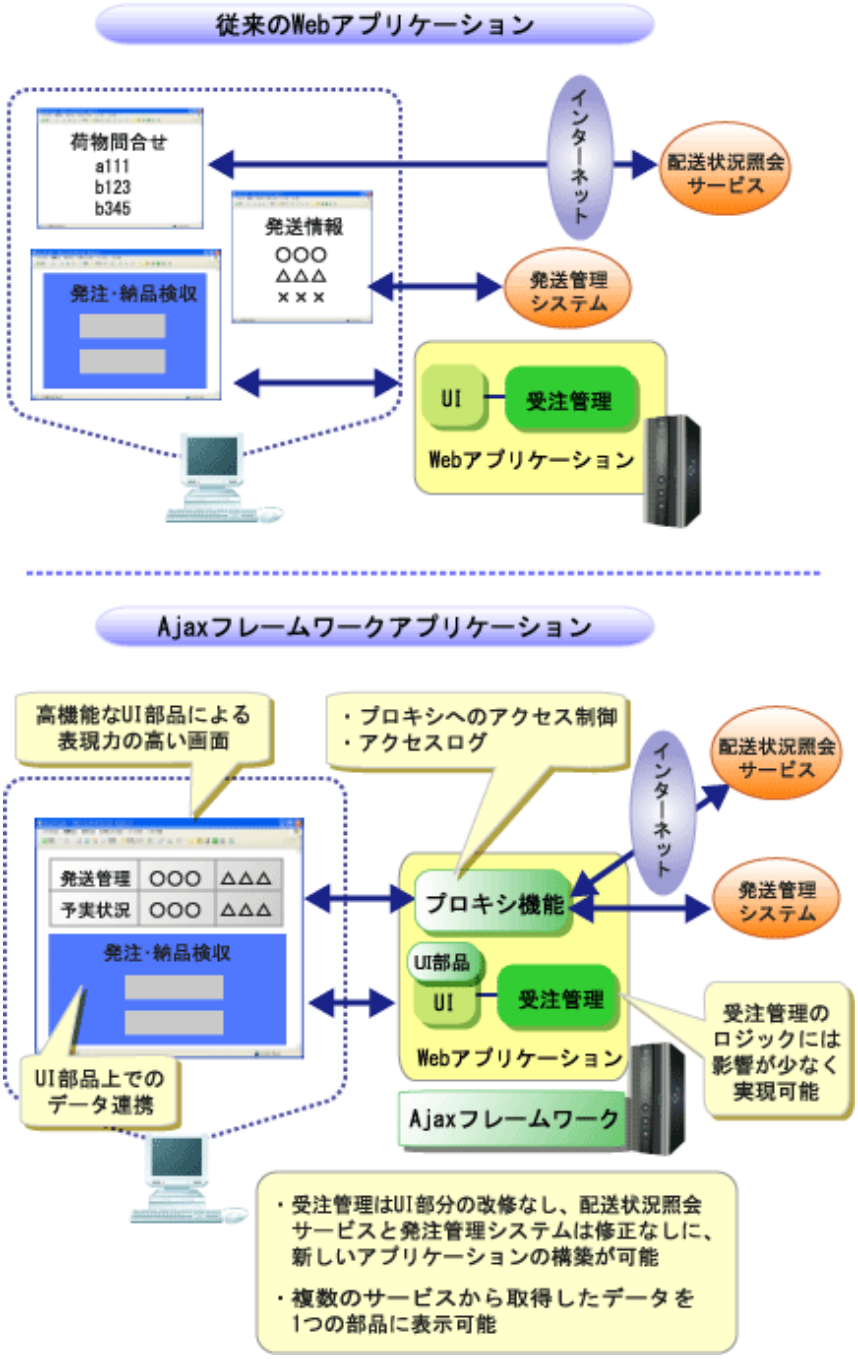
業務画面への情報の集約による作業効率の改善

従来、企業内では、業務ごとに異なったシステムが利用されていました。このため、ある入力業務を行うには、複数の業務アプリケーション画面を起動したあと、一方で表示されている情報を見ながらもう一方で手作業による入力を行う必要がありました。
Ajaxフレームワークでは、企業内のサービスやインターネット上にあるWebサービスを組み合わせて、関連する情報を1つの業務画面に集約(マッシュアップ)することができます。これにより、手番や入力ミスを削減し、ユーザーの作業効率の改善を実現することができます。



既存システムやサービスを組み合わせたWebアプリケーションの短期構築

最近の業務システム構築では、既存システムを有効活用するために、既存システムを組み合わせて新たなシステムを構築したいというニーズが高まっています。また、インターネット上のWebサービスを業務システムで活用したいというニーズも高まっています。
Ajaxフレームワークでは、インターネット上にあるWebサービスや企業内の既存システムをブラウザ上で連携させることができます。これにより、既存のアプリケーションへの影響が少なく、より使いやすいWebアプリーションが短期に構築できるようになります。
一方、インターネット上のWebサービスをブラウザから直接呼び出す場合、クロスドメイン制約などのセキュリティ上の課題があります。Ajaxフレームワークでは、アクセス制御やロギングを備えたプロキシ機能を提供しており、このプロキシを利用することにより、より安全にサービスを呼び出すことができます。

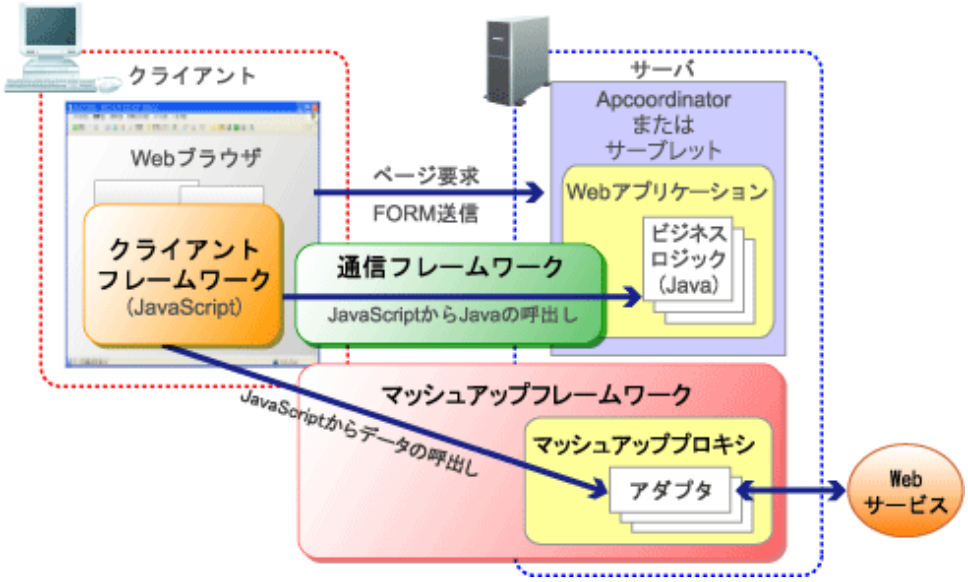


1.3 Ajaxフレームワークの機能構成

Ajaxフレームワークは、Webブラウザ上のJavaScriptで動作するクライアントフレームワーク、JavaScriptからサーバ側のビジネスロジックを呼び出す通信フレームワーク、およびJavaScriptからWebサービスを呼び出すマッシュアップフレームワークから構成されています。

以下に、Ajaxフレームワークの機能構成を説明します。

図1.2 Ajaxフレームワークの機能構成



クライアントフレームワーク

クライアントフレームワークは、Webブラウザで高い操作性を実現するとともに、開発生産性の向上のために、Webブラウザ上で動作するJavaScriptのフレームワークを提供します。

クライアントフレームワークの詳細は、「第2章 クライアントフレームワーク」を参照してください。

通信フレームワーク

通信フレームワークは、JavaScriptから、サーバ側のビジネスロジックを非同期で呼び出すためのフレームワークを提供します。

通信フレームワークの詳細は、「第3章 通信フレームワーク」を参照してください。

注意

通信フレームワークが連携できるサーバ側のアプリケーションは、サーブレットアプリケーションまたはApcoordinatorのアプリケーションです。対応するServlet仕様のバージョンは、以下のとおりです。

- サーブレットアプリケーションの場合: 2.4または2.5
- Apcoordinatorのアプリケーションの場合: 2.4

マッシュアップフレームワーク

マッシュアップフレームワークは、WebブラウザとWebサービスの通信を中継するフレームワークを提供します。複数のWebサービスを組み合わせたWebアプリケーションの作成が可能となります。

マッシュアップフレームワークの詳細は、「第4章 マッシュアップフレームワーク」を参照してください。

ポイント

Ajaxフレームワークのクライアントアプリケーションが使用するデータ形式は、JSON形式です。

参考

システムの運用/保守について

Ajaxフレームワークを利用したシステムの運用および保守については、ご利用のアプリケーションサーバのマニュアルを参照してください。

第2章 クライアントフレームワーク

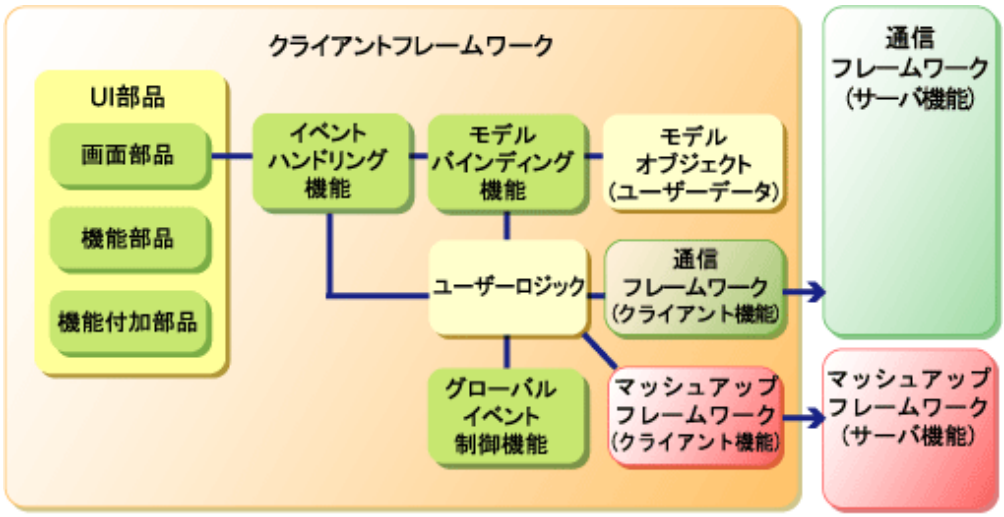
本章では、クライアントフレームワークの機能および使用方法について説明します。

2.1 クライアントフレームワークの構成

クライアントフレームワークは、Webブラウザで高い操作性を実現するとともに、開発生産性の向上のために、Webブラウザ上で動作するJavaScriptのフレームワークを提供します。

以下の図に、クライアントフレームワークの構成を示します。

図2.1 クライアントフレームワークの構成



クライアントフレームワークでは、UI部品(画面部品、機能部品、機能付加部品)、イベントハンドリング機能、モデルバインディング機能、およびグローバルイベント制御機能を提供します。

2.1.1 UI部品

UI部品とは、クライアントで使用するAjaxフレームワークの部品群の総称です。
UI部品は、画面部品、機能部品、機能付加部品の3種類に分けられます。

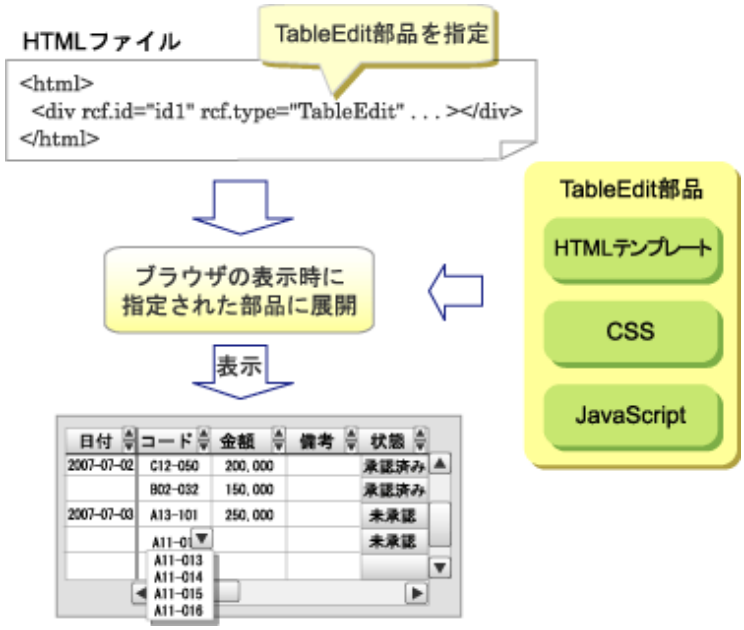
画面部品

画面部品は、画面に表示される機能を提供する部品群です。

それぞれの部品は、HTMLテンプレート(画面定義)、CSS(スタイル)、JavaScript(動作)から構成されています。HTMLに決められた記述方式で部品を指定すると、Webブラウザでの実行時に動的に指定された部品に展開され、その結果が表示されます。

以下の図に、画面部品の実装例を示します。

図2.2 画面部品の実装例



機能部品

機能部品は、画面に表示されない機能を提供する部品群のうち、単体で使用できるものです。データモデルの定義などをします。

機能付加部品

機能付加部品は、画面に表示されない機能を提供する部品群のうち、画面部品に機能を付加するものです。入力フィールドへの入力時に文字列を補完するオートコンプリーション機能、フォーカス制御機能など、画面部品に対して機能を付加します。

2.1.2 イベントハンドリング機能

イベントハンドリング機能は、画面に対するイベントリスナを管理し、イベントの発生時に適切なユーザーロジック(JavaScript)を呼び出す機能です。

画面とユーザーロジックを分離することにより、これらの独立性が高まります。これにより、画面とユーザーロジックの保守・再利用が容易になります。

詳細は、「2.4.2 ユーザーロジックの定義(イベント)」、「2.4.3 イベントオブジェクト」、および「2.4.4 画面初期表示時のユーザーロジックの呼出し」を参照してください。

2.1.3 モデルバインディング機能

モデルバインディング機能は、画面部品から入力されたデータをモデルオブジェクトに反映したり、モデルオブジェクト内のデータが更新されたときに画面部品に値を反映したりする機能です。

画面とモデルオブジェクトを分離することにより、これらの独立性が高まります。これにより、画面とモデルオブジェクトの保守・再利用が容易になります。

詳細は、「2.5.3 モデルバインディング」を参照してください。

2.1.4 グローバルイベント制御機能

グローバルイベント制御機能は、画面そのものに対して操作したときに発生するイベントを制御する機能です。F1～F12までのファンクションキーに対して、keydownおよびkeyupのイベントを検出することが可能です。

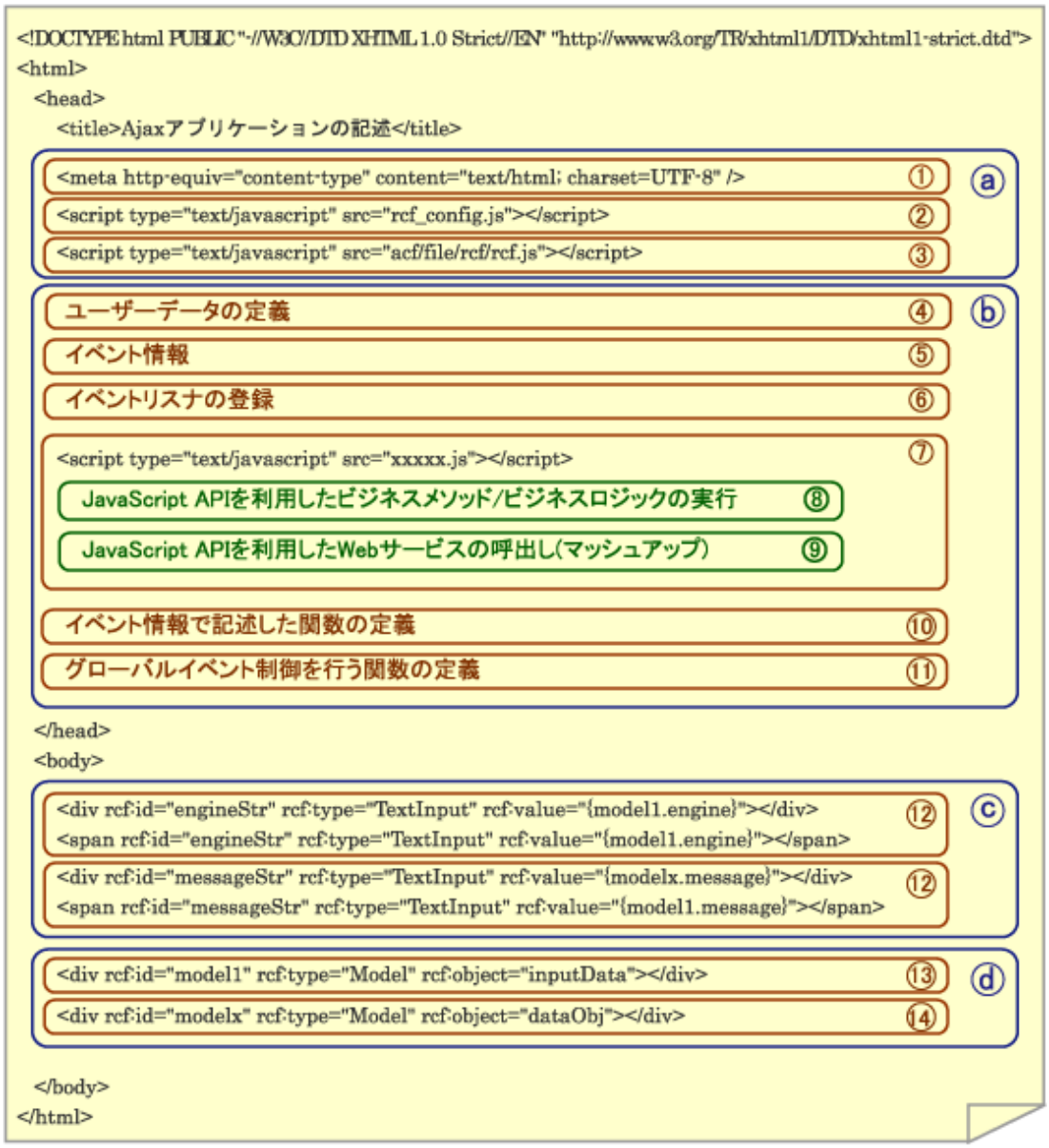
詳細は、「2.6.2 グローバルイベント制御」を参照してください。

2.2 HTML/JSPファイルの記述方法

Ajaxフレームワークの画面は、HTMLおよびJSPファイルで定義します。

以下に、HTML/JSPファイルの記述内容の概要を示します。

図2.3 HTML/JSPファイルの記述内容



a. Ajaxフレームワークの宣言部
   1) content-typeの宣言
   2) Ajaxフレームワークの動作定義
   3) Ajaxフレームワークの初期化処理

b. ユーザーデータ/ユーザーロジックの定義部
   4) ユーザーデータの定義
   5) イベント情報
   6) イベントリスナの登録
   7) ユーザーロジックの定義
   8) JavaScript APIを利用したビジネスメソッド/ビジネスロジックの実行
   ・Apcoordinator連携
   ・サーブレット連携(汎用通信方式)

・サーブレット連携(簡易通信方式-非同期通信)
・サーブレット連携(簡易通信方式-同期通信)
9) JavaScript APIを利用したWebサービスの呼出し(マッシュアップ)
10) イベント情報で記述した関数の定義
11) グローバルイベント制御を行う関数の定義

c. 画面情報定義部
12) 部品を使用するための定義

d. 機能定義部
13) ユーザーデータをモデルとして利用するための機能定義
14) データBeanに設定するデータ定義をモデルとして利用するための機能定義

- Ajaxフレームワークの画面には、必ずXHTMLのDOCTYPE宣言を記述し、XHTMLで内容を記述してください。
- スクリプトレットなどを使用する場合には、動的に生成されたXHTML中のタグがAjaxフレームワークで処理されます。このため、動的に生成されたXHTMLが、本章および「UI部品リファレンス」に記載されている規則を満たす必要があります。

2.3 Ajaxフレームワークの宣言

Ajaxフレームワークの宣言部は、以下の3つから構成されています。

- content-typeの宣言
- Ajaxフレームワークの動作定義
- Ajaxフレームワークの初期化処理

Ajaxフレームワークの宣言部は、ユーザーデータ/ユーザーロジックの定義部、画面情報定義部、機能定義部よりも前に記述してください。

ここでは、それぞれの記述内容について説明します。

2.3.1 content-typeの宣言

Ajaxフレームワークを用いたHTML/JSPファイルは、UTF-8またはwindows-31j(Shift_JIS)で作成する必要があります。content-typeは、以下のとおり記述してください。

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
```

Ajaxフレームワークのプロジェクトで使用するHTML/JSPファイルのエンコードは統一する必要があります。

2.3.2 Ajaxフレームワークの動作定義

Ajaxフレームワークの動作定義を行うために、HTML/JSPファイルに、以下の内容を記述します。

```
<script type="text/javascript" src="rcf_config.js"></script>
```

rcf_config.jsでは、RCF_configオブジェクトの変数を設定することで、Ajaxフレームワークの動作を定義することができます。

以下に、RCF_configオブジェクトの設定例を示します。

```
RCF_config = {
  logLevel: 0,
  eventLogLevel: 'ERROR',
```

```
  utc: false
};
```

以下の表に、RCF_configオブジェクトの設定項目を示します。

表2.1 RCF_configオブジェクトの設定項目

変数名	意味	省略時の動作
logLevel	画面上に出力するログのレベルを以下の数値で指定します。数値が大きいほど、出力されるログの情報量が増えます。 ・ 0 ERRORレベルのログだけを出力 ・ 1 ERRORおよびWARNレベルのログを出力 ・ 2 ERROR、WARN、およびINFOレベルのログを出力 ・ 3 ERROR、WARN、INFO、およびTRACEレベルのログを出力	ログは出力されません。
eventLogLevel	サーバ側に出力するログのレベルを以下の文字列で指定します。 ・ ERROR ERRORレベルのログだけを出力 ・ WARN ERRORおよびWARNレベルのログを出力 ・ INFO ERROR、WARN、およびINFOレベルのログを出力 ・ TRACE ERROR、WARN、INFO、およびTRACEレベルのログを出力	ログは出力されません。
utc	Dateオブジェクトを利用するUI部品において、UTCとローカル時間のどちらを利用するかを指定します。 ・ true UTCで動作する ・ false ローカル時間で動作する	false
queryString	URLに付加するクエリ文字列を連想配列で指定します。 設定例については、「3.9.1 クエリ文字列の追加」を参照してください。	クエリ文字列は付加されません。
jsessionid	URLリライティングを行う場合に、URLに付加するセションIDを指定します。 設定例については、「3.9.2 URLリライティング」を参照してください。	セションIDは付加されません。

変数名	意味	省略時の動作
apcConnection_compatible	Apcoordinator連携機能を互換モードで動作させる場合に、そのバージョンをString型で指定します。 ・ 9.0 V9.0の互換モードで動作する 詳細は、「H.1.2 Apcoordinator連携機能の互換情報」を参照してください。	現在のバージョンの動作になります。
connection_compatible_errorCodeRCF0700	Apcoordinator連携機能のビジネスメソッドの実行中にサーバが停止している場合、エラーオブジェクト内のerrorCodeプロパティに文字列「RCF」を付加して通知するかどうかを指定します。 ・ true 文字列「RCF」を付加して通知する ・ false 数値だけで通知する 詳細は、「H.1.1 共通の互換情報」を参照してください。	false
basePoint_inContainer	コンテナ部品内にUI部品およびHTML要素を配置したときに、コンテナ部品の左上を配置の基準位置にするかどうかを指定します。 ・ true コンテナ部品の左上を配置の基準位置にする (Panel、TabPanel、およびWindowの場合は、ボディ部の左上が基準位置になります。) ・ false コンテナ部品の左上を配置の基準位置にしない (V9.0.1以前の開発資産が該当します。) falseを指定した場合、基準位置が親要素にあるときは親要素が、基準位置の指定がないときは通常の配置が基準になります。 なお、Ajaxページエディタで作成していない業務画面を、Ajaxページエディタの設計ビューを使用して編集する場合は、trueを指定してください。	false
UIComponent_compatible_viewsize	UI部品の表示サイズをV9.0.1およびV9.1までと同様に表示する場合に、そのバージョンをString型で指定します。 ・ 9.0.1 V9.0.1までと同様の表示サイズで表示する ・ 9.1.0 V9.1までと同様の表示サイズで表示する 詳細は、「H.2.1 UI部品の互換情報」を参照してください。	width/heightで指定したとおりのサイズで部品が表示されます。

変数名	意味	省略時の動作
	なお、Ajaxページエディタの設計ビューで画面を編集する場合は、本変数を指定しないでください。	
UIComponent_compatible_error	UI部品のプロパティの型チェックをV9.1.1までと同様にする場合に、そのバージョンをString型で指定します。 ・ 9.1.1 V9.1.1までと同様の型チェックを行う 詳細は、「H.2.1 UI部品の互換情報」を参照してください。	現在のバージョンの動作になります。

UIComponent_compatible	UI部品の動作を従来と同様にする場合に指定します。
	・ DataGrid上での文字列選択
	以下を指定した場合に文字列の選択動作ができなくなります。
	DataGrid: {
	selection: "9.0.1"
	}
	・ ComboBoxの非活性時のボタン
	以下を指定した場合にボタンの表示が従来と同様になります。
	ComboBox: {
	buttonDisabled: "9.1.1"
	}
	詳細は、「H.2.1 UI部品の互換情報」を参照してください。
		現在のバージョンの動作になります。

StyleSheet_compatible	スタイルシート適用の優先順位をV9.1.1までと同様にする場合に、そのバージョンをString型で指定します。 ・ 9.1.1 V9.1.1までと同様の優先順位でスタイルシートを適用する 詳細は、「H.2.1 UI部品の互換情報」を参照してください。	現在のバージョンの動作になります。

注意

- Ajaxフレームワークの動作定義(rcf_config.js)は、Ajaxフレームワークの初期化処理(rcf.js)よりも先に記述しなければなりません。
- サブフォルダに配置されているHTML/JSPファイルを直接表示する場合、srcプロパティで指定するrcf_config.jsについて、パスの考慮が必要です。アプリケーションフォルダの直下に存在するrcf_config.jsが参照できるように相対パスで指定してください。
  アプリケーションフォルダ直下のJSPフォルダにJSPファイルが存在する場合の記述例を以下に示します。

```
<script type="text/javascript" src="../rcf_config.js"></script>
```

  HTML/JSPファイルをインクルードしている場合には、インクルード元のHTML/JSPファイルが存在するフォルダを基点とした相対パスで指定してください。

ポイント

Ajaxフレームワークの動作定義は、画面ごとに直接記述することも可能ですが、複数画面で共通に使用する場合は、外部ファイルに記述することを推奨します。

2.3.3 Ajaxフレームワークの初期化処理

Ajaxフレームワークを起動するためには、HTML/JSPファイルに、以下の内容を記述します。

```
<script type="text/javascript" src="acf/file/rcf/rcf.js"></script>
```

この記述により、サーバ側に配置されているrcf.jsファイルがロードされて、Ajaxフレームワークの初期化処理が実行されます。
初期化処理で実行される処理は、以下のとおりです。

- 基本ライブラリのロード
- HTMLの解析と、フレームワークの画面部品への変換
- 各画面部品の初期化

注意

サブフォルダに配置されているHTML/JSPファイルを直接表示する場合、srcプロパティで指定するrcf.jsについて、パスの考慮が必要です。アプリケーションフォルダを基点とした「acf/file/rcf/rcf.js」が参照できるように相対パスで指定してください。
アプリケーションフォルダ直下のJSPフォルダにJSPファイルが存在する場合の記述例を以下に示します。

```
<script type="text/javascript" src="../acf/file/rcf/rcf.js"></script>
```

HTML/JSPファイルをインクルードしている場合には、インクルード元のHTML/JSPファイルが存在するフォルダを基点とした相対パスで指定してください。

2.4 ユーザーデータ/ユーザーロジックの定義部

ここでは、ユーザーデータ、ユーザーロジックの定義部の記述内容について説明します。
また、入力データの検証方法についても説明します。

2.4.1 ユーザーデータの定義

ユーザーデータの定義では、画面表示時の初期値や、入力データを格納するためのデータ格納域を定義します。

ユーザーデータの定義は、データを格納するためのJavaScript記述から構成されます。

定義例

以下に、ユーザーデータの定義例を示します。

```
<script type="text/javascript">
//<![CDATA[
  var inputData = {
    engine: '',
    audio: '',
    transmission: '',
    otherOptions: {
      rack: '',
      package: '',
      gps: ''
    }
  };
//]]>
</script>
```

```
（省略）

// ユーザーデータをモデルとして利用するための機能定義部
<div rcf:id="model1" rcf:type="Model" rcf:object="inputData"></div>
```

定義したユーザーデータをモデルとして利用するためには、機能定義部を記述します。詳細は、「2.5.2 機能定義部」を参照してください。

ポイント

ユーザーデータを複数画面で共通に利用する場合は、外部ファイルに記述することを推奨します。
外部ファイルの記述については、「2.6.4 外部ファイルの定義」を参照してください。ユーザーデータを記述した外部ファイルは、FragmentContainer部品で使用します。FragmentContainer部品の使用方法については、「UI部品リファレンス」の「2.2.5 FragmentContainer」を参照してください。

2.4.2 ユーザーロジックの定義(イベント)

画面部品にイベントリスナを定義すると、ユーザーの操作実行時に発生したイベントを監視し、制御を行います。
このイベントリスナの定義をユーザーロジックの定義と呼びます。

画面上での操作に対して発生するイベントには、以下の2種類があります。

a. 画面部品に対してユーザーが何らかの操作を行った結果、発生するイベント

b. 画面部品に関係なく画面そのものに対して行われた操作の結果、発生するイベント

ここでは、a.のイベントについて、以下の項目を説明します。b.のイベントについては、「2.6.2 グローバルイベント制御」を参照してください。

- イベント情報
- イベント名
- イベントリスナの登録
- イベント情報で記述した関数の定義

イベント情報

ある画面部品で発生したイベントに対して、何らかの動作を定義するイベントリスナを対応付けるには、以下の方法があります。また、この対応付けた内容をイベント情報と呼びます。

- 画面部品ごとに複数のイベントを一括して定義する方法
- 画面部品とイベントの組ごとに定義する方法
- 画面部品に直接定義する方法

以下に、それぞれの方法について説明します。

画面部品ごとに複数のイベントを一括して定義する方法

任意のオブジェクトに対して、画面部品のid属性名(以降、画面部品IDと呼びます)と、イベント名とイベントリスナ関数名から成る1つ以上の連想配列を、連想配列で定義します。また、複数の画面部品IDを一括して定義することができます。

以下に、記述形式を示します。

```
オブジェクト={画面部品ID:{イベント名:関数名,イベント名:関数名,...},
              画面部品ID:{イベント名:関数名,イベント名:関数名,...},
              ...}
```

以下に、記述例を示します。

```
<script type="text/javascript">
//<![CDATA[
  eventmap = {                          ←任意のオブジェクト名
```

```
    engineStr : {                    ←画面部品IDがengineStrのイベントを定義
      mouseover: func_emphasize,     ←マウスオーバー時にfunc_emphasize関数を実行
      mouseout: func_deemphasize,    ←マウスアウト時にfunc_deemphasize関数を実行
      click: func_eventListener      ←クリック時にfunc_eventListener関数を実行
    },
    audioStr : {
      mouseover: func_emphasize,
      mouseout: func_deemphasize,
      click: func_eventListener
    },
    transmission : {
      click: function() {            ←無名関数としても定義可能
        RCF.debug("transmission clicked");
      }
    }
  };
//]]>
</script>
```

画面部品とイベントの組ごとに定義する方法

任意のオブジェクトに対して、画面部品IDとイベント名をアンダーバー(_)でつないだものと、そのイベントリスナ関数名を連想配列で定義します。このとき、複数の画面部品IDを一括して定義することができます。

以下に、記述形式を示します。

```
オブジェクト={画面部品ID_イベント名:関数名, 画面部品ID_イベント名:関数名,...}
```

以下に、記述例を示します。

```
<script type="text/javascript">
//<![CDATA[
  myevent = {                                ←任意のオブジェクト名
    TreeView1_datachange: func_datachange,  ←IDがTreeView1という画面部品のデータが変更された場合
    TreeView1_treenodeexpand: func_treenodeexpand,
    TreeView1_treenodecollapse: func_treenodecollapse,
    TreeView1_treenodeselect: func_treenodeselect
  };
//]]>
</script>
```

画面部品に直接定義する方法

画面部品に対して、直接、イベントリスナを定義します。

以下に、記述例を示します。

```
<div rcf:type="TextInput" rcf:onValueChange="myValueChange()">
```

ポイント

イベント情報は、通常、画面(HTML/JSPファイル)単位に記述します。ただし、複数画面で共通に利用する場合は、外部ファイルに記述することを推奨します。
外部ファイルの記述については、「2.6.4 外部ファイルの定義」を参照してください。イベント情報を記述した外部ファイルは、FragmentContainer部品で使用します。FragmentContainer部品の使用方法については、「UI部品リファレンス」の「2.2.5 FragmentContainer」を参照してください。

イベント名

イベント名は、各画面部品で利用可能なイベントリスナ名の先頭の「on」を削除し、残りをすべて小文字で表現したものです。

例えば、onValueChangeというイベントリスナが画面部品に定義されている場合は、valuechangeをイベント名として使用します。

イベントリスナの登録

オブジェクトに定義されたイベントリスナは、APIを利用してフレームワークに登録することで、利用可能となります。

イベントリスナの登録には、rcf.event.EventRegistrar.registerEvents関数を使用します。イベントリスナの登録解除には、rcf.event.EventRegistrar.unregisterEvents関数を使用します。

以下に、それぞれの使用例を示します。

- イベントリスナの登録

```
rcf.event.EventRegistrar.registerEvents(eventmap, 'myeventmap1'); ←myeventmap1という名前を付けて登録
rcf.event.EventRegistrar.registerEvents(eventmap);                ←名前を付けずに登録
```

- イベントリスナの登録解除

```
rcf.event.EventRegistrar.unregisterEvents('myeventmap1');         ←登録時に付けた名前を指定して登録解除
```

注意

イベントリスナ登録時に名前を付けずに登録した場合は、イベントリスナの登録解除はできません。

ポイント

イベントリスナの登録は、画面(HTML/JSPファイル)単位に記述します。

イベント情報で記述した関数の定義

イベント情報の定義とイベントリスナ(イベント情報で記述した関数)の定義は別に行います。つまり、別の定義ブロックに記述します。
なお、定義済みのイベントリスナをイベント情報に定義する場合があります。

以下に、記述例を示します。

```
<script type="text/javascript">
//<![CDATA[
  function func_emphasize() {
    RCF.debug("mouse over");
  }
  function func_deemphasize() {
    RCF.debug("mouse out");
  }
  function func_eventListener() {
    RCF.debug("mouse over");
  }
//]]>
</script>
```

以下に、イベント情報で記述した関数がイベントオブジェクトを受け取る場合の記述例を示します。
以下の例は、「イベント情報」の「画面部品とイベントの組ごとに定義する方法」の記述例に対応する関数定義です。

```
<script type="text/javascript">
//<![CDATA[
  var modelData = {
    treeData: {
      name: "root",
      children: [
        { // 1つ目のノード
          name: "fj",
          label: "Fujitsu Group",
          isExpanded: false,
          children: []
```

```
        }
      ]
    }
  };

  function treenodeexpand(eventObject) {
    // 追加する子ノード
    var newTreeNode = {
      name: "fja",
      label: "Fujitsu Australia",
      isExpanded: false
    };

    //イベントで通知されたノードパスの部分データを取得
    var provider = model1.getNodeDataProvider("treeData", eventObject.nodePath);
    //イベントで通知されたノードの子ノード配列を取得
    provider = provider.getDataProvider("children");
    //子ノード配列に1つのノードデータを追加
    provider.addItem(newTreeNode);
  }
//]]>
</script>
```

ポイント

イベントリスナの定義は、通常、画面(HTML/JSPファイル)単位に記述します。ただし、複数画面で共通に利用する場合は、外部ファイルに記述することを推奨します。
外部ファイルの記述については、「2.6.4 外部ファイルの定義」を参照してください。イベントリスナ定義を記述した外部ファイルは、FragmentContainer部品で使用します。FragmentContainer部品の使用方法については、「UI部品リファレンス」の「2.2.5 FragmentContainer」を参照してください。

2.4.3 イベントオブジェクト

イベントが発生してイベントリスナが呼ばれた場合、イベントリスナの引数にイベントオブジェクトが渡されます。

イベントオブジェクトには様々な種類がありますが、ここでは、アクションイベントとプロパティチェンジイベントについて説明します。そのほかのイベントオブジェクトについては、「UI部品リファレンス」の「付録A イベントオブジェクト」を参照してください。

アクションイベント

アクションイベントは、画面部品が操作されたときに発生するイベントで、実体はrcf.event.ActionEventです。

以下に、記述例を示します。

```
{
  target: イベントを送出したオブジェクト,
  type: 'イベント名',
  browserEvent: ブラウザのイベント
}
```

以下に、rcf.event.ActionEventのプロパティを示します。

プロパティ名	概要
target	イベントを送出したオブジェクトです。
type	イベント名です。
browserEvent	ブラウザのイベントです。 JavaScriptが提供するイベントオブジェクトをラップしたオブジェクトが格納されています。

プロパティチェンジイベント

プロパティチェンジイベントは、モデルの値が変更されたときに発生するイベントで、実体はrcf.event.PropertyChangeEventです。

以下に、記述例を示します。

```
{
  target: イベントを送出したオブジェクト,
  type: 'イベント名',
  propertyName: '変更された値のプロパティ名',
  oldValue: '変更前のプロパティの値',
  newValue: '変更後のプロパティの値'
}
```

以下に、rcf.event.PropertyChangeEventのプロパティを示します。

プロパティ名	概要
target	イベントを送出したオブジェクトです。
type	イベント名です。
propertyName	値が変更されたプロパティ名です。
oldValue	値が変更される前のプロパティの値です。
newValue	値が変更されたあとのプロパティの値です。

2.4.4 画面初期表示時のユーザーロジックの呼出し

画面を初期表示するときにユーザーロジックを呼び出す場合は、RCF.addLoadedListener関数またはRCF.addInitializedListener関数を利用します。

- RCF.addLoadedListener
  Ajaxフレームワークの初期化処理が開始される前に呼ばれる関数
- RCF.addInitializedListener
  Ajaxフレームワークによって画面が初期化された直後に実行される関数

これらの関数を利用してイベントリスナを登録しておくと、任意のユーザーロジックを実行することができます。

RCF.addLoadedListener関数は、以下のどちらかの方法の代わりに使用します。

- HTMLの<body>タグにonloadアトリビュートを記述する
- JavaScriptのwindowオブジェクトのonloadに関数を設定する

ただし、RCF.addLoadedListener関数は、Ajaxフレームワークの初期化が行われていないため、Ajaxフレームワークが提供するJavaScriptのAPIを利用することはできません。ユーザーロジックやユーザーデータの初期化などに利用してください。

RCF.addInitializedListener関数は、Ajaxフレームワークの初期化が完了しているため、Ajaxフレームワークが提供するJavaScriptのAPIを利用することができます。

以下に、RCF.addInitializedListener関数を利用して、イベントリスナの登録とFragmentContainer部品の有効化を行う例を示します。

```
<script type="text/javascript">
//<![CDATA[
myEvent = {
    // イベントリスナを記述
};

RCF.addInitializedListener(function() {
  rcf.event.EventRegistrar.registerEvents(myEvent, "myEvent"); // イベントリスナを登録
  myFragmentContainer1.activate();                             // FragmentContainerの有効化
  myFragmentContainer2.activate();                             // FragmentContainerの有効化
});
//]]>
</script>
```

```
  (省略)

<div rcf:id="myViewStack" rcf:type="ViewStack" rcf:selectedIndex="0">
  <div rcf:id="myFragmentContainer1" rcf:type="FragmentContainer" rcf:src="..."></div>
  <div rcf:id="myFragmentContainer2" rcf:type="FragmentContainer" rcf:src="..."></div>
</div>
```

UI部品でエラーが発生した場合の処理

UI部品でエラーが発生した場合は、RCF.addErrorListener関数に登録された関数が実行されます。

RCF.addErrorListener関数に登録する関数には、引数を1つだけ定義することができます。引数には、エラーの内容を格納したオブジェクトが渡され、errorプロパティを参照すると、エラーオブジェクトが取得できます。

以下に、RCF.addErrorListener関数を利用した例を示します。

```
<script type="text/javascript">
//<![CDATA[
RCF.addErrorListener(function(eventObject) {
    // エラーオブジェクトを取得してアラートを表示
    var message = eventObject.error.message;
    if (!message) {
       message = eventObject.error.toString();
    }
    alert("エラーが発生しました：" +  message);
});
//]]>
</script>
  (省略)
```

2.4.5 入力データの検証

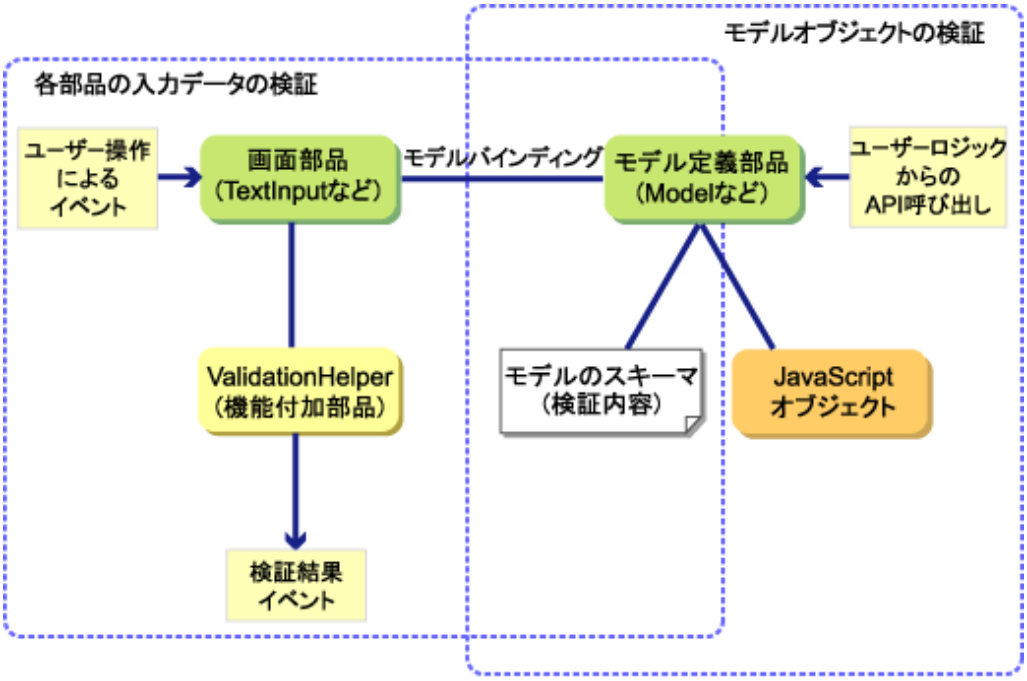
入力データの検証では、モデルオブジェクトに格納されたデータや画面部品から入力されたデータに誤りがないかを検証します。

入力データを検証するには、以下の2つの方法があります。

- モデルオブジェクトの検証
  モデル定義部品のAPIを呼び出し、モデルに関連付けられたスキーマにより、モデルオブジェクト全体を検証します。
- 各部品の入力データの検証
  画面部品のイベントをトリガとし、画面部品のプロパティのうち、モデルとバインディングしているプロパティの値に関して検証します。

以下の図に、入力データの検証方法の概要を示します。

図2.4 入力データの検証方法の概要

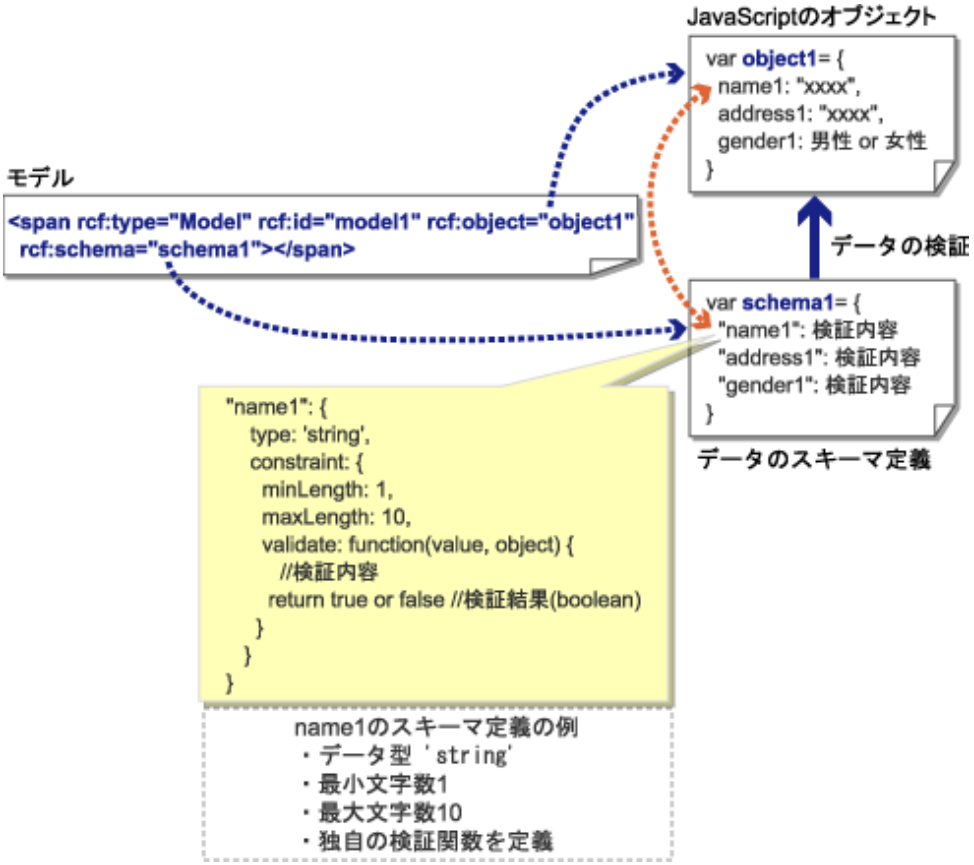


モデルオブジェクトの検証

モデルを定義するモデル定義部品に、そのモデルのデータのスキーマを定義することにより、モデルオブジェクトを検証することができます。

以下の図に、モデルオブジェクトの検証例を示します。

図2.5 モデルオブジェクトの検証例



モデル定義部品の定義には、rcf:objectでオブジェクトを指定し、そのオブジェクトを検証するスキーマをrcf:schemaで指定します。スキーマ定義は、JavaScriptの連想配列で記述し、要素名には検証対象のプロパティのパスを、値には検証内容を指定します。
上記の例では、name1の検証内容として、データ型(string)、最小文字数、最大文字数、および独自の検証関数を指定しています。

モデルデータの検証を実行するには、JavaScriptで以下のようにモデル定義部品のAPIを実行します。

```
var result = model1.validate();
```

検証結果(result)は、成功した場合はnullが返されます。エラーがあった場合は、検証エラー情報を含んだ連想配列が返されます。
例えば、name1の検証内容に指定した「最大文字数」および「独自の検証関数」に対して検証エラーがあった場合、以下のような連想配列が返されます。

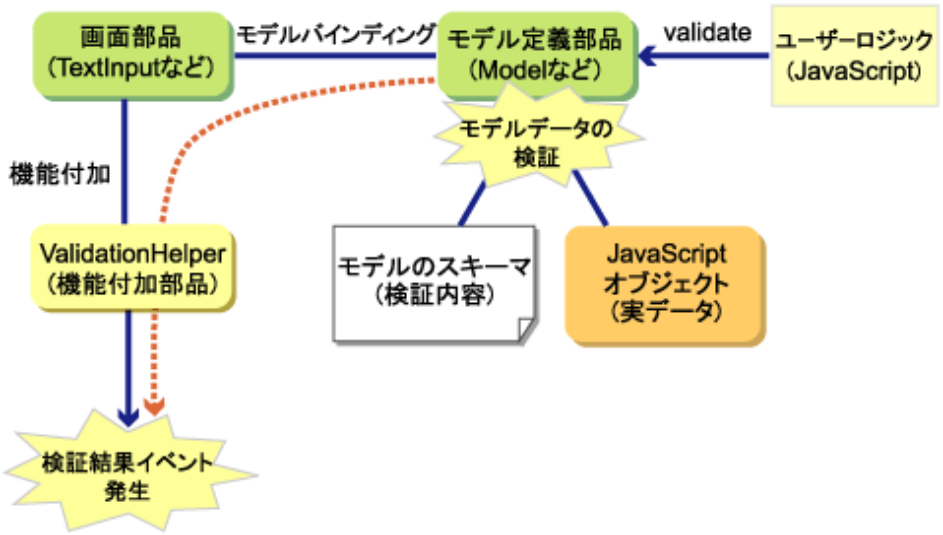
```
{
 "name1": [
   {最大文字数に対する検証エラー情報},
   {独自の検証関数に対する検証エラー情報} ],

 "address1": [
    {...}, {...}
 ]
}
```

連想配列の要素名には、検証でエラーとなったプロパティのパス名が含まれます。値には、エラーとなった項目の情報が配列で含まれます。詳細は、「UI部品リファレンス」の「3.1 モデル定義部品」を参照してください。

また、検証を実行した場合、検証したモデルのプロパティが画面部品にバインディングしていて、かつ、画面部品にValidationHelperが付加されている場合、ValidationHelperの検証結果イベント(validationerrorまたはvalidationsuccess)が発生します。

以下の図に、その仕組みを示します。

図2.6 検証結果イベントの送出

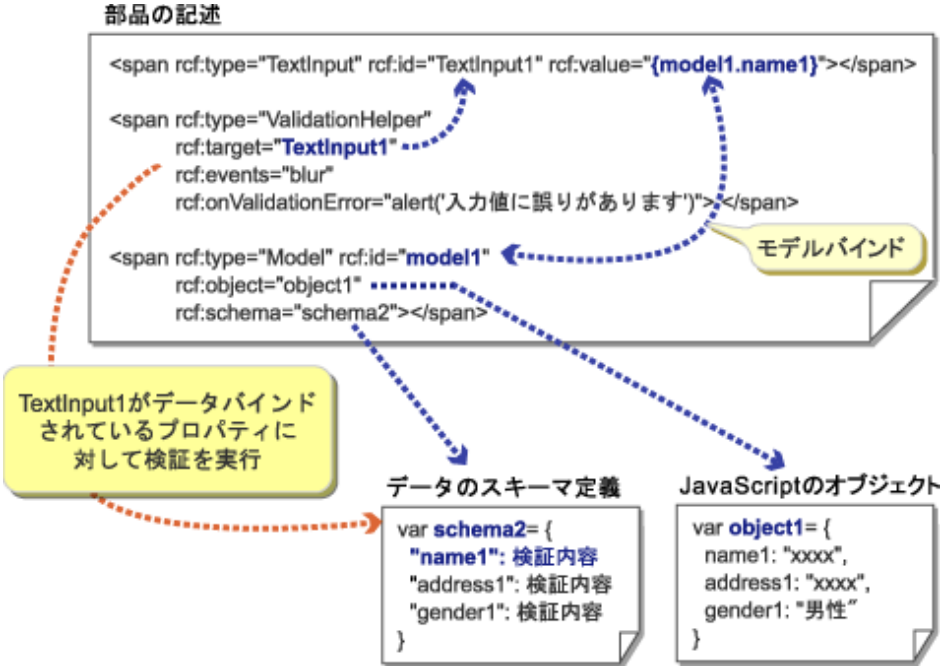


各部品の入力データの検証

TextInputなどの画面部品でイベントをトリガとし、データの検証を行う場合は、画面部品に機能付加部品であるValidationHelperを付加します。これにより、画面部品のプロパティのうち、モデルとバインディングしているプロパティの値を検証することができます。

以下の図に、部品の入力データの検証例を示します。

図2.7 部品の入力データの検証例



ValidationHelperでは、targetプロパティで対象とする画面部品、eventsプロパティで検証を行うトリガとなるイベントを指定します。

上記の例のように、ValidationHelperを設定した場合、フォーカスを失ったときに、TextInputのvalueプロパティとバインディングしているモデルの値(実体はobject1のname1プロパティ)が検証されます。検証は、モデルに指定したデータのスキーマ定義に基づいて実行されます。そのため、モデルにスキーマ定義が指定されている必要があります。

ValidationHelperでは、検証の結果により、以下のイベントリスナが呼ばれます。

- onValidationSuccess : 検証に成功した場合
- onValidationError : 検証に失敗した場合

ユーザーは、これらのイベントに対する処理を記述することによって、検証結果に対する動作を規定することができます。上記の例では、onValidationErrorイベントリスナで「入力値に誤りがあります」というアラートを表示させています。

2.5 画面情報定義部・機能定義部

ここでは、画面情報定義部と機能定義部の記述内容について説明します。

2.5.1 画面情報定義部

画面情報定義部では、HTMLの タグまたはタグを利用して、Ajaxフレームワークが提供するUI部品の使用を宣言します。

タグとタグでは、以下のように、改行の挿入方法が異なります。

- タグ
  部品の前後に改行が挿入される
- タグ
  部品の前後に改行が挿入されない

UI部品を使用するための記述方法の詳細は、「UI部品リファレンス」の「1.3 UI部品の使い方」を参照してください。

以下に、定義例を示します。

```
// TextInputを表示する場合の定義例
<div rcf:id="engineStr" rcf:type="TextInput" rcf:value="{model1.engine}"></div>
<span rcf:id="engineStr" rcf:type="TextInput" rcf:value="{model1.engine}"></span>
```

上記例では、「engineStr」というIDでTextInput部品を定義しています。TextInput部品は、UI部品が提供する画面部品の1つです。
また、ユーザーデータとバインディングするために、valueとして「model1.engine」を記述しています。
model1は、「2.4.1 ユーザーデータの定義」で定義したように、inputDataであるため、そのengineプロパティが結びついていることになります。

注意

使用する部品によっては、 タグとタグのどちらか一方の記述しかできない場合があります。詳細は、「UI部品リファレンス」の「1.3 UI部品の使い方」を参照してください。

2.5.2 機能定義部

機能定義部では、以下の2種類の定義を行います。

- ユーザーデータをモデルとして利用するための機能定義
  ユーザーデータの定義部で定義したユーザーデータをモデルとして利用するために定義します。
  以下に、定義例を示します。

```
<div rcf:id="model1" rcf:type="Model" rcf:object="inputData"></div>
```

- データBeanに設定するデータ定義をモデルとして利用するための機能定義
  以下に、定義例を示します。

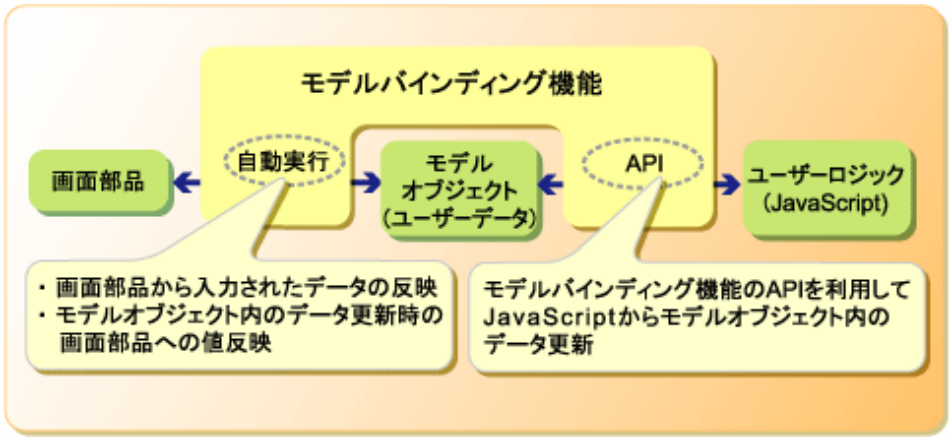
```
<div rcf:id="modelx" rcf:type="Model" rcf:object="dataObj"></div>
```

2.5.3 モデルバインディング

クライアントフレームワークでは、画面部品とユーザーデータ(モデルオブジェクト)を分離するためのモデルバインディング機能を提供します。

以下の図に、モデルバインディング機能の構成を示します。

図2.8 モデルバインディング機能の構成



バインディングを行うには、その対象に応じて、以下の方法があります。

- 画面部品とモデルのバインディング
- ユーザーロジックとモデルのバインディング
- 画面部品と画面部品のバインディング

バインディング式

画面部品とモデルオブジェクトを関連付けるには、バインディング式を利用します。

バインディング式を利用する場合は、JavaScriptオブジェクトのプロパティ参照を{ }で囲みます。

以下の例では、myModelオブジェクトのmypropプロパティをバインディング式で表現しています。

```
{myModel.myprop}
```

画面部品とモデルのバインディング

画面部品とモデルのバインディングを行う場合は、ユーザーデータに対して機能部品でモデル名を定義し、画面部品でバインディング式を利用して関連付けます。

以下に、記述例を示します。なお、この例の「rcf:type="Model"」が機能部品の定義です。

```
// JavaScriptのオブジェクトとしてmyDataを宣言
<script type="text/javascript">
//<![CDATA[
    var myData= {
      string1: 'データ'
    };
//]]>
</script>

// myDataをモデルオブジェクトmyModelとして利用
<div rcf:id="myModel" rcf:type="Model" rcf:object="myData"></div>

//モデルオブジェクトmyModelのstring1プロパティをTextInputの値にバインディング
<div rcf:id="myTextInput" rcf:type="TextInput" rcf:value="{myModel.string1}"></div>
```

ユーザーロジックとモデルのバインディング

モデルの値をユーザーロジックから参照/更新する場合は、モデルオブジェクトに対してgetPropertyメソッドまたはsetPropertyメソッドを利用します。

以下に、それぞれの記述形式を示します。

- 値の参照

```
Object getProperty(<string> key)
```

- 値の更新

```
void setProperty(<string> key, <Object> value)
```

以下に、ユーザーロジックからモデルの値を参照/更新する例を示します。

```
// JavaScriptのオブジェクトとしてmyDataを宣言
<script type="text/javascript">
//<![CDATA[
    var myData= {
      string1: 'データ'
    };
//]]>
</script>

// myDataをモデルオブジェクトmyModelとして利用
<div rcf:id="myModel" rcf:type="Model" rcf:object="myData"></div>

// myModelの値を参照・更新
<script type="text/javascript">
//<![CDATA[
    // 値の参照
    var v = myModel.getProperty('string1');
    // 値の更新
    myModel.setProperty('string1', 'newData');
//]]>
</script>
```

モデルバインディング機能を利用している場合は、モデルオブジェクトのプロパティを直接操作することはできません。直接操作した場合は、モデルバインディング機能が動作せず、操作した値だけが変更されます。

画面部品と画面部品のバインディング

2つ以上の画面部品をバインディングする場合は、一方の画面部品のid属性名(画面部品ID)を利用することで、もう一方と関連付けることができます。

以下に、記述例を示します。この例では、valueプロパティを利用して、画面部品の値を参照しています。

```
// JavaScriptのオブジェクトとしてmyDataを宣言
<script type="text/javascript">
//<![CDATA[
    var myData= {
      string1: 'データ'
    };
//]]>
</script>

// myDataをモデルオブジェクトmyModelとして利用
<div rcf:id="myModel" rcf:type="Model" rcf:object="myData"></div>

// myDataモデルオブジェクトのstring1プロパティをTextInputの値にバインディング
<div rcf:id="myTextInput" rcf:type="TextInput" rcf:value="{myModel.string1}"></div>

// myTextInputの値を別のTextInputの値にバインディング
<div rcf:id="myTextInput2" rcf:type="TextInput" rcf:value="{myTextInput.value}"></div>
```

2.6 その他の定義

ここでは、クライアントフレームワークの以下の機能について説明します。

- デバッグ機能
- グローバルイベント制御機能
- データプロバイダ
- 外部ファイルの定義
- UI部品のフォーカス制御

2.6.1 デバッグ機能

ブラウザ上のデバッグのためのAPIを利用して、任意のウィンドウの、任意の位置にデバッグメッセージを出力することができます。

デバッグ機能を利用する場合は、Ajaxフレームワークの動作定義(RCF_configオブジェクト)に、以下のように指定します。

```
RCF_config = {
  logLevel: 3              ←logLevelには0～3の数値を指定
};
```

プロパティについては、「2.3.2 Ajaxフレームワークの動作定義」を参照してください。

デバッグメッセージは、 タグまたはタグでidに「rcf-Logger」が指定されている箇所に出力されます。

以下に、 タグを利用した指定例を示します。

```
<div id="rcf-Logger"></div>
```

以下に、デバッグメッセージの出力方法を示します。

```
 <script type="text/javascript">
 //<![CDATA[
   RCF.debug("メッセージ");       ←logLevelに3が指定されている場合に出力
   RCF.info("メッセージ");        ←logLevelに2以上が指定されている場合に出力
   RCF.warn("メッセージ");        ←logLevelに1以上が指定されている場合に出力
   RCF.error("メッセージ");       ←logLevelに0以上が指定されている場合に出力
 //]]>
 </script>
```

2.6.2 グローバルイベント制御

ファンクションキーが押された場合などのイベントは、グローバルイベント制御機能を利用して検出できます。グローバルイベント制御機能では、F1～F12までのファンクションキーに対して、keydownおよびkeyupのイベントを検出します。

グローバルイベントを取得する場合は、HTMLの<body>タグに検出したいイベントに該当する属性を記述します。以下に、記述例を示します。

```
  (省略)
  // グローバルイベント制御を行う関数の定義
  <script type="text/javascript">
  //<![CDATA[
    function func_keydownF1() {
      RCF.debug("keydown F1");
    }
  //]]>
  </script>
</head>

<body rcf:onKeyDownF1="func_keydownF1()">       ←検出したイベントに該当する属性を記述
  (省略)
</body>
```

以下に、記述できる属性を示します。

属性名	意味
onKeyDownF1	F1キーのkeydownに対応するイベントリスナ
onKeyDownF2	F2キーのkeydownに対応するイベントリスナ
onKeyDownF3	F3キーのkeydownに対応するイベントリスナ
onKeyDownF4	F4キーのkeydownに対応するイベントリスナ
onKeyDownF5	F5キーのkeydownに対応するイベントリスナ
onKeyDownF6	F6キーのkeydownに対応するイベントリスナ
onKeyDownF7	F7キーのkeydownに対応するイベントリスナ
onKeyDownF8	F8キーのkeydownに対応するイベントリスナ
onKeyDownF9	F9キーのkeydownに対応するイベントリスナ
onKeyDownF10	F10キーのkeydownに対応するイベントリスナ
onKeyDownF11	F11キーのkeydownに対応するイベントリスナ
onKeyDownF12	F12キーのkeydownに対応するイベントリスナ
onKeyUpF1	F1キーのkeyupに対応するイベントリスナ
onKeyUpF2	F2キーのkeyupに対応するイベントリスナ
onKeyUpF3	F3キーのkeyupに対応するイベントリスナ
onKeyUpF4	F4キーのkeyupに対応するイベントリスナ
onKeyUpF5	F5キーのkeyupに対応するイベントリスナ

属性名	意味
onKeyUpF6	F6キーのkeyupに対応するイベントリスナ
onKeyUpF7	F7キーのkeyupに対応するイベントリスナ
onKeyUpF8	F8キーのkeyupに対応するイベントリスナ
onKeyUpF9	F9キーのkeyupに対応するイベントリスナ
onKeyUpF10	F10キーのkeyupに対応するイベントリスナ
onKeyUpF11	F11キーのkeyupに対応するイベントリスナ
onKeyUpF12	F12キーのkeyupに対応するイベントリスナ

ポイント

複数画面で共通のグローバル制御を行う場合は、グローバルイベント制御を行う関数の定義を外部ファイルに記述することを推奨します。外部ファイルの記述については、「2.6.4 外部ファイルの定義」を参照してください。グローバルイベント制御を行う関数を記述した外部ファイルは、FragmentContainer部品で使用します。FragmentContainer部品の使用方法については、「UI部品リファレンス」の「2.2.5 FragmentContainer」を参照してください。

2.6.3 データプロバイダ

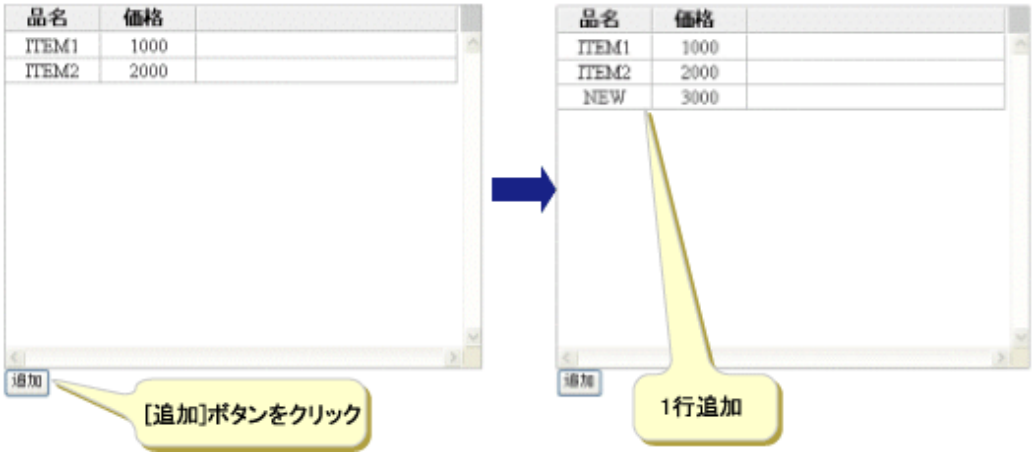
UI部品のプロパティのうち、object型プロパティおよびarray型プロパティについては、objectのプロパティを更新したり、arrayの要素を追加/削除したりするなど、値の一部を更新して、表示内容に反映することができます。このプロパティの値の一部分を更新することを、部分更新と呼びます。
この部分更新を行うために、データプロバイダがあります。

以下に、データプロバイダを利用して、テーブル部品(TableView)に新たな行を追加する例を示します。

```
<script type="text/javascript">
//<![CDATA[
var modelData = {
    tableData: [
      {name: 'ITEM1', price:1000},
      {name: 'ITEM2', price:2000}
    ]
};

function add() {                                                    // (1)
    var newItem = {name:'NEW', price:3000};
    model1.getDataProvider("tableData").addItem(newItem);
}
//]]>
</script>
  (省略)
<div rcf:id="model1" rcf:type="Model" rcf:object="modelData"></div>

<div rcf:id="table1" rcf:type="TableView" rcf:data="{model1.tableData}">  // (2)
  <div rcf:type="ViewColumn" rcf:name="name" rcf:label="品名"></div>
  <div rcf:type="ViewColumn" rcf:name="price" rcf:label="価格"></div>
</div>

<div rcf:type="Button" rcf:onClick="add()">追加</div>                    // (3)
```

上記の例では、(3)のボタンがクリックされると、(1)のadd関数が呼ばれるようになっています。
add関数では、Modelからテーブルのデータであるtable Dataのデータプロバイダを取得し、addItem関数を利用して新しい配列要素を追加しています。
また、(2)のテーブル部品(TableView)がtableDataをバインディングしており、addItem関数で変更されたことにより、テーブルの表示内容に1行追加されます。

以下に、画面例を示します。



以下に、部分更新に関する注意事項を示します。

- boolean、string、numberなどのプリミティブ型については、値の一部を変更することはできませんので、部分更新という概念はありません。
- object型またはarray型のプロパティには、部分更新に対応しているプロパティと対応していないプロパティがあります。対応しているかどうかは、「UI部品リファレンス」の各部品のプロパティの説明を参照してください。
- データプロバイダのAPIまたはモデル定義部品のsetProperty以外を利用して、プロパティの一部を変更しないでください。この場合、表示内容への反映ができません。また、モデルバインディング機能を利用している場合はバインディング先との同期ができないため、データに不整合が生じます。

データプロバイダの種類およびAPIについては、「UI部品リファレンス」の「付録B データプロバイダ」を参照してください。

2.6.4 外部ファイルの定義

複数の画面から共通に使用するHTML/JSPファイルは、外部ファイルに作成します。

外部ファイルで作成できる機能は以下のとおりです。

- ユーザーデータ定義
- イベント定義
- ユーザーロジック定義
- グローバルイベント制御を行う関数の定義

外部ファイルの作成方法は、「5.7.4 ユーザーロジック定義の作成」を参照してください。

以下に、外部ファイルの定義例を示します。この例のように、外部ファイルには、<html><head><body>といった、htmlの構造を表すタグは使用しません。

```
<%@ page contentType= "text/html; charset=UTF-8" %>

<!-- Javaスクリプト -->
<script type="text/javascript">
//<![CDATA[
// ユーザーデータ定義、イベント定義、ユーザーロジック定義を記述
//]]>
</script>

<!-- 画面情報定義部 -->
<div>
伝票番号 <span rcf:id="number" rcf:type="NumberInput" rcf:number="{searchDataModel.number}"></span>
<div rcf:id="searchButton" rcf:type="Button" rcf:value="search">検索</div>
</div>
```

```
<!-- 機能定義部 -->
<div rcf:id="searchDataModel" rcf:type="Model" rcf:object="searchData"></div>
```

作成した外部ファイルは、FragmentContainer部品で使用します。rcf:src属性に外部ファイルを指定してください。
以下に、ViewStack部品で複数のFragmentContainer部品を切り替えて使用する例を示します。
FragmentContainer部品の使用方法の詳細は、「UI部品リファレンス」の「2.2.5 FragmentContainer」を参照してください。

```
<div rcf:id="viewStack" rcf:type="ViewStack" rcf:selectedIndex="0" rcf:height="270px">
  <!-- 入力画面 -->
  <div rcf:id="dataInputContainer" rcf:type="FragmentContainer" rcf:src="dataInputContainer.jsp">
  </div>
  <!-- 一覧画面 -->
  <div rcf:id="listViewContainer" rcf:type="FragmentContainer" rcf:src="listViewContainer.jsp">
  </div>
  <!-- 検索画面 -->
  <div rcf:id="searchContainer" rcf:type="FragmentContainer" rcf:src="searchContainer.jsp">
  </div>
</div>
```

注意

外部ファイルの文字コードは、その外部ファイルを使用するFragmentContainer部品を定義したファイルと同じコードでなければなりません。

2.6.5 UI部品のフォーカス制御

TextInputやCalendarなどのフォーカス設定可能なUI部品間で、フォーカスの移動順序の制御を行うには、以下の2通りの方法があります。

- UI部品のtabIndexプロパティを使用する方法
- FocusManager部品を使用する方法

また、RCF.setFocus関数を使用して、ユーザーロジックから特定の部品にフォーカスを設定することもできます。

なお、フォーカス制御の各機能は、組み合わせて使用することができます。

UI部品のtabIndexプロパティを使用したフォーカス制御

tabIndexプロパティを持つUI部品では、その値を設定することにより、HTMLのtabindex属性と同じ効果を得ることができます。

以下に、tabIndexプロパティを使用したフォーカス制御の例を示します。

```
  <div rcf:type="TextInput" rcf:tabIndex="1"></div>
  <div rcf:type="CheckList" rcf:tabIndex="0" rcf:value="foo;bar"></div>
  <div rcf:type="Calendar" rcf:tabIndex="2"></div>
```

上記の例では、Tabキーを押すと、TextInput部品→Calendar部品→CheckList部品の順でフォーカスが移動します。

FocusManager部品を使用したフォーカス制御

FocusManager部品を利用すると、任意のキーでの部品間のフォーカス制御や、フォーカス制御範囲の切替えが可能になります。
FocusManager部品の詳細は、「UI部品リファレンス」の「4.2.1 FocusManager」を参照してください。

以下に、FocusManager部品を使用したフォーカス制御の例を示します。

```
  <div rcf:id="input1" rcf:type="TextInput"></div>
  <div rcf:id="list1" rcf:type="CheckList" rcf:value="foo;bar"></div>
  <div rcf:id="cal1" rcf:type="Calendar"></div>

  <div rcf:type="FocusManager" rcf:targets="input1;cal1;list1"      ←フォーカス移動順にUI部品のIDを指定
       rcf:tabEnabled="false"></div>                                ←Tabキーでのフォーカス移動を無効化
```

上記の例では、Tabキーでのフォーカス移動を無効化しています。また、Enterキーを押すと、TextInput部品→Calendar部品→CheckList部品の順でフォーカスが移動します。

RCF.setFocus関数を使用したフォーカスの設定

RCF.setFocus関数を使用して、ユーザーロジックからUI部品にフォーカスを設定することができます。

以下に、RCF.setFocus関数の記述形式を示します。

```
RCF.setFocus(id)
```

- id
  フォーカスを設定するUI部品のID文字列を指定します。

以下に、RCF.setFocus関数の使用例を示します。

```
<div rcf:id="button1" rcf:type="Button"
     rcf:onClick="window1.show()">Window表示</div>     // (1)

<div rcf:id="window1" rcf:type="Window"
     rcf:mode="1" rcf:closeButton="true"
     rcf:onShow="RCF.setFocus('input1')"               // (2)
     rcf:onHide="RCF.setFocus('button1')">             // (3)
     <div rcf:id="input1" rcf:type="TextInput"></div>
</div>
```

上記の例では、以下のように動作します。
(1) Button(ID=button1)がクリックされると、Window(ID=window1)を表示します。
(2) window1表示時に、window1内のTextInput(ID=input1)にフォーカスを設定します。
(3) window1を非表示にする際、button1にフォーカスを戻します。

以下に、RCF.setFocus関数に関する注意事項を示します。

- フォーカス設定可能な部品は、FocusManager部品が対象とする部品と同じです。
  - フォーム部品(Textは除く)
  - TabPanel
  - TableView
  - TableEdit
  - DataGrid
  - Calendar
  - CalendarButton
  - TreeView
  - ScrapingView(コンテンツ内は除く)
  - CheckBoxGroup
  - RadioButtonGroup
- 以下の場合、RCF.setFocus関数はフォーカスを設定しないで、例外を通知します。
  ( )内は、例外オブジェクトのメッセージ番号です。
  - 指定されたIDを持つUI部品が存在しない場合
    (メッセージ番号なし、メッセージ本文に[UndeclaredComponentIdError]と設定される)
  - 指定されたUI部品がフォーカスを設定できない部品の場合 (RCF11014)
  - 指定されたUI部品が無効化されている場合 (RCF11015)
  - 指定されたUI部品、またはそのUI部品の親要素が表示されていない場合 (RCF11016)

第3章 通信フレームワーク

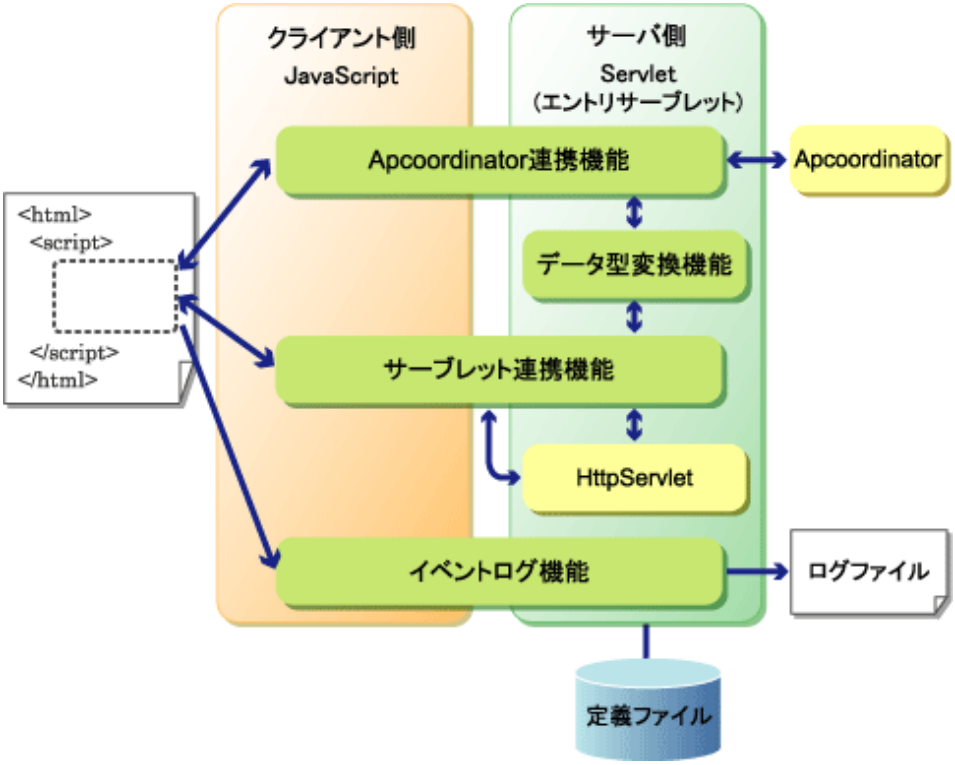
本章では、通信フレームワークの機能および使用方法について説明します。

3.1 通信フレームワークの構成

通信フレームワークは、クライアントJavaScriptアプリケーションとサーバ間の通信を支援する機能の総称です。Webブラウザ上で動作するJavaScriptから、非同期で、サーバ側にあるJavaで記述されたビジネスロジックを呼び出すためのフレームワークを提供します。

以下の図に、通信フレームワークの構成を示します。

図3.1 通信フレームワークの構成



通信フレームワークでは、Apcoordinator連携機能、サーブレット連携機能、データ型変換機能、およびイベントログ機能を提供します。

Apcoordinator連携機能、サーブレット連携機能、およびイベントログ機能は、それぞれ、サーバ側とクライアント側で動作します。データ型変換機能は、サーバ側でだけ動作します。
サーバ側では、エントリサーブレットとして動作します。エントリサーブレットの詳細は、「5.3 エントリサーブレット」を参照してください。
クライアント側では、JavaScriptとして動作します。このJavaScript APIは、例外を通知することがあります。通知される例外オブジェクトの構造は、「3.10 通信フレームワーク内のエラー」を参照してください。

Apcoordinator連携機能

Apcoordinator連携機能は、ブラウザ上のJavaScriptアプリケーションから、Apcoordinatorを利用する機能です。これにより、Apcoordinator上に作成された資産を活用することができます。

Apcoordinator連携機能の詳細は、「3.2 Apcoordinator連携機能」を参照してください。

サーブレット連携機能

サーブレット連携機能は、ブラウザ上のJavaScriptアプリケーションから、サーブレットを利用する機能です。これにより、サーブレット上に作成された資産を活用することができます。

サーブレット連携機能の詳細は、「3.3 サーブレット連携機能」を参照してください。

データ型変換機能

データ型変換機能は、クライアントのJavaScriptとサーバのJava間での双方向オブジェクト変換機能です。通信フレームワークは、メソッド実行前後に、オブジェクト変換を行います。

データ型変換機能の詳細は、「3.6 データ型変換機能」を参照してください。

イベントログ機能

イベントログ機能は、ブラウザ上のJavaScriptアプリケーションから、サーバ上にログを出力する機能です。

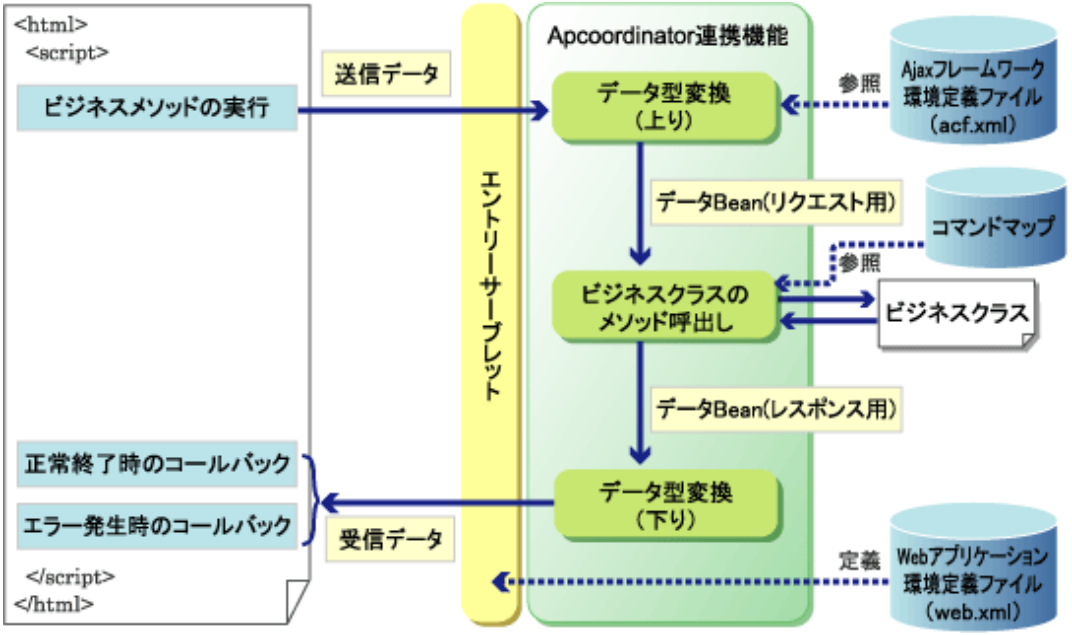
イベントログ機能の詳細は、「3.7 イベントログ機能」を参照してください。

3.2 Apcoordinator連携機能

Apcoordinator連携機能を利用すると、Webブラウザ上のJavaScriptからApcoordinatorのビジネスクラスを呼び出すことができます。

以下の図に、Apcoordinator連携機能の処理の流れを示します。

図3.2 Apcoordinator連携機能の処理の流れ



ビジネスメソッドの実行

JavaScriptからApcoordinatorのビジネスクラスを呼び出します。送信データ、その送信データを格納するデータBean IDとコマンド名、サーバの処理結果を受け取るコールバックなどを指定します。
ビジネスメソッドの実行の詳細は、「3.2.6 ビジネスメソッドの実行」を参照してください。

エントリサーブレットとWebアプリケーション環境定義ファイル(web.xml)

JavaScriptとサーバ上のビジネスクラスとの間でデータを送受信するためには、通信フレームワークが提供するcom.fujitsu.interstage.rcf.AcfServletクラスを継承したユーザー定義エントリサーブレットを作成します。また、Webアプリケーション環境定義ファイル(web.xml)に定義したエントリサーブレットのマッピング情報を指定します。
エントリサーブレットとWebアプリケーション環境定義ファイルの詳細は、「3.2.2 エントリサーブレットの作成」を参照してください。

Ajaxフレームワーク環境定義ファイル(acf.xml)

データBean IDとデータBeanのクラスの対応関係、およびJavaScriptとJavaのデータ型変換について、デフォルトのルールで変換できない場合の変換規則を指定します。
Ajaxフレームワーク環境定義ファイルの詳細は、「3.2.5 Ajaxフレームワーク環境定義ファイルの設定」を参照してください。

データ型変換(上り)

送信データを、指定されたデータBeanに変換します。ビジネスメソッドの実行時に指定されたデータBean IDに基づいて、Ajaxフレームワーク環境定義ファイルから対応するデータBeanのクラスを探してデータを格納します。データの格納の際には、データ型変換を行います。
データ型変換の詳細は、「3.6 データ型変換機能」を参照してください。

コマンドマップ

ビジネスクラスのメソッドを呼び出すためのApcoordinatorの定義ファイルです。送信データが格納されるデータBeanのクラス名とコマンド名から、呼び出すビジネスクラス名、メソッド名を定義したファイルです。
コマンドマップの詳細は、「3.2.4 コマンドマップの定義」を参照してください。

ビジネスクラスのメソッド呼出し

ビジネスメソッドの実行時に指定されたコマンド名と、データBean IDから割り出されたデータBeanのクラス名に基づいて、コマンドマップを参照し、対応するビジネスクラス、メソッドを実行します。ビジネスメソッドの引数には、データ型変換(上り)が行われたデータBeanが渡されます。

ビジネスクラス

ビジネスメソッドの実行によって、呼び出されるビジネスクラスです。引数には、データBeanが渡され、ビジネスメソッドの戻り値には、処理結果が格納されたデータBeanを返却します。
ビジネスクラスの詳細は、「3.2.3 ビジネスクラスの作成」を参照してください。

データ型変換(下り)

ビジネスクラスの戻り値として返却されたデータBeanを、JavaScriptで取り扱うことのできる受信データに変換します。受信データの格納の際には、データ型変換を行います。
データ型変換の詳細は、「3.6 データ型変換機能」を参照してください。

正常終了時のコールバック

ビジネスメソッドの実行が正常に終了した場合に、処理結果を受け取るコールバック関数です。ビジネスメソッドの実行時の引数として指定します。
正常終了時のコールバックの詳細は、「3.2.6 ビジネスメソッドの実行」および「3.2.7 コールバック関数 」を参照してください。

エラー発生時のコールバック

ビジネスメソッドの実行時にエラーが発生した場合に、エラー結果を受け取るコールバック関数です。ビジネスメソッド実行中のタイムアウトについても、このコールバック関数に通知されます。エラー発生時のコールバックは、正常終了時のコールバックと同様に、ビジネスメソッドの実行時の引数として指定します。省略した場合には、エラー結果を受け取ることはできません。
エラー発生時のコールバックの詳細は、「3.2.6 ビジネスメソッドの実行」および「3.2.7 コールバック関数 」を参照してください。

3.2.1 データBeanの作成

ここでは、クライアントからの送信データを受け取るデータBean、クライアントに返却するためのデータを格納するデータBeanの作成方法について説明します。

データBeanは、Apcoordinatorアプリケーションと同様に、com.fujitsu.uji.DataBeanクラスを継承して作成します。データBeanに利用できるプロパティの型は、データ型変換機能で対応する型です。詳細は、「3.6 データ型変換機能」を参照してください。

以下に、データBeanの記述例を示します。

```
package mypkg;
import com.fujitsu.uji.DataBean;

public class BodyBean extends DataBean {
    protected String message;
    public String getMessage() {
        return message;
    }
```

```
    public void setMessage(String message) {
        this.message = message;
    }
}
```

3.2.2 エントリサーブレットの作成

Apcoordinator連携機能を利用して、クライアントとサーバとの間でデータの送受信を行うには、com.fujitsu.interstage.rcf.AcfServletクラスを継承したユーザー定義エントリサーブレットを作成し、Webアプリケーション環境定義ファイル(web.xml)に定義したユーザー定義エントリサーブレットのマッピング情報を指定する必要があります。
詳細は、「5.3 エントリサーブレット」を参照してください。

3.2.3 ビジネスクラスの作成

ここでは、クライアントからの送信データを受け取って処理を行うビジネスクラスの作成方法について説明します。

ビジネスクラスは、Apcoordinatorアプリケーションと同様に、com.fujitsu.uji.GenericHandlerクラスを継承して作成します。ただし、ビジネスメソッドの戻り値は、returnで記述します。返却されたオブジェクトは、クライアントのレスポンスハンドラに渡されます。

なお、ビジネスクラスのインスタンスのライフサイクルは、initメソッドの戻り値で決まります。
また、ビジネスクラスに渡されるDispatchContextオブジェクトの実体は、HttpDispatchContextクラスを継承したAcfHttpDispatchContextオブジェクトです。

以下に、ビジネスクラスの記述例を示します。

```
package mypkg;
import com.fujitsu.uji.DispatchContext;
import com.fujitsu.uji.GenericHandler;

public class MyHandler extends GenericHandler
{
    public boolean init() {
        return true;
    }

    public Object doSomething(DispatchContext context, BodyBean dataBean) {
        // 業務ロジックを記述
        (省略)

        // 戻り値はreturnで記述
        NextBean retBean = new NextBean();
        return retBean;
    }
}
```

参考

Apcoordinatorの表示画面の領域名に設定されたデータBeanをビジネスクラスで受け取ることもできます。この場合のビジネスクラスの作成方法は、「3.2.8 ApcoordinatorとのデータBeanの共有」を参照してください。

3.2.4 コマンドマップの定義

作成したビジネスクラスのメソッドを呼び出すためのコマンドマップの定義方法について説明します。

コマンドマップは、テキストファイルに記述します。

以下に、コマンドマップの記述形式を示します。

```
入力のデータBeanのクラス名:コマンド=ビジネスクラス名.メソッド名
```

- 入力のデータBeanのクラス名
  作成したデータBeanまたは共有するデータBeanのクラス名を設定します。
- コマンド
  Apcoordinator連携機能で指定するコマンド名を設定します。
- ビジネスクラス名.メソッド名
  作成したビジネスクラスのクラス名とメソッド名を設定します。

以下に、コマンドマップの定義例を示します。

```
# commands.map
mypkg.BodyBean:execute=mypkg.MyHandler.doSomething
```

コマンドマップの詳細は、Interstage Application Serverの「Apcoordinator ユーザーズガイド」を参照してください。

3.2.5 Ajaxフレームワーク環境定義ファイルの設定

Ajaxフレームワーク環境定義ファイルでは、データBeanの情報を指定します。

- JavaScriptから指定する際の別名(ID)
- クラス名
- データBeanオブジェクトのライフサイクル

Ajaxフレームワーク環境定義ファイルの指定方法については、「A.5 データBeanの定義(dataBeans)」を参照してください。

以下に、samples.HelloBeanクラスに「HelloBean」というIDを付けて、セションスコープで管理する例を示します。

```
<acfConfig>
  (省略)
  <dataBeans>
    <dataBean>
      <dataBeanId>HelloBean</dataBeanId>
      <className>samples.HelloBean</className>
      <scope>session</scope>
    </dataBean>
  </dataBeans>

</acfConfig>
```

参考

- ビジネスメソッドの実行時に、データBean IDにApcoordinatorの表示画面の領域名を指定した場合、領域名に設定されたデータBeanをビジネスクラスで受け取ることができます。この場合、dataBean要素の指定を省略することができます。詳細は、「3.2.8 Apcoordinatorとのデータの共有」を参照してください。
- ビジネスメソッドの戻り値となるデータBeanには、送信データを受け取るデータBeanとは異なるデータBeanを指定することができます。この際、戻り値となるデータBeanについては、Ajaxフレームワーク環境定義ファイルのデータBeanの定義(dataBeans)は必要ありません。
- データBeanをセションスコープで管理している場合、セションタイムアウトが発生すると処理を継続できません。セションタイムアウト時間は、業務を考慮して適切な時間を設定してください。また、セションタイムアウト発生後も処理を継続させたい場合には、以下のどちらかの対処をしてください。
  - セションスコープで管理されているデータBeanに対して、Ajaxフレームワーク環境定義ファイルのdataBean要素の子要素であるrecreate要素に「true」を指定し、セションタイムアウト発生時にデータBeanを再作成するようにしてください。この場合、アプリケーションでセションタイムアウトを検出することはできません。詳細は、「A.5 データBeanの定義(dataBeans)」を参照してください。
  - Interstage Application Serverの「Apcoordinator ユーザーズガイド」の「セション切断の検出」の「セション切断時に特定の処理を実行する」に記載されている内容に従って、対処してください。なお、入出力ページを作成する際の注意事項については関係ありません。

3.2.6 ビジネスメソッドの実行

JavaScriptでApcoordinatorのビジネスクラスを呼び出すには、UjiRequestオブジェクトのsend関数を使用します。

UjiRequest.send関数の形式

以下に、UjiRequest.send関数の記述形式を示します。

```
UjiRequest.send(dataObj, requestParams, option);
```

- **dataObj**
  サーバ側に送信するデータのオブジェクトです。
  データを送信しない場合は、nullを指定します。

- **requestParams**
  コマンド名やデータBean IDなど、Apcoordinatorの動作を制御するための値を格納したリクエストパラメーターオブジェクトです。
  リクエストパラメーターが不要な場合は、nullを指定します。

  以下に、リクエストパラメーターオブジェクトのプロパティを示します。

プロパティ名	概要	型	省略可否
beanId	送信データを格納するデータBean ID Apcoordinatorアプリケーションとデータbeanを共有しない場合は、Ajaxフレームワーク環境定義ファイルに定義されたデータBeanの名前を指定します。 Apcoordinatorアプリケーションとデータbeanを共有する場合は、データBeanを関連付けたデータBean IDを指定します。(注)	string	dataObjがnullの場合は省略可 それ以外の場合は省略不可
verb	コマンド名 コマンドマップで定義されたコマンドと対応します。	string	可

  注) データBean IDにデータBeanを関連付けるには、DispatchContextクラスのsetResponseBeanメソッドを利用します。詳細は、「3.2.8 Apcoordinatorとのデータbeanの共有」を参照してください。beanIdにそのデータBean IDを指定すると、Apcoordinatorが保持するデータBeanにデータが格納されます。
  Apcoordinatorの画面表示の際、<uji:useBean>タグを利用する場合は、request="true"を指定します。request="true"を指定すると、次のリクエストまで、Apcoordinatorがデータbeanを保持します。

- **option**
  UjiRequestの動作を制御するための値を格納した通信設定オブジェクトです。
  optionにnullが指定された場合、または省略された場合は、通信せずにエラーが発生します。

  以下に、通信設定オブジェクトのプロパティを示します。

プロパティ名	概要	型	省略可否
url	通信先のURL 「acf/apc」を指定します。サブフォルダに配置されているHTML/JSPファイルからビジネスメソッドを実行する場合、urlプロパティで指定する「acf/apc」について、パスの考慮が必要です。アプリケーションフォルダを基点とした「acf/apc」が参照できるように相対パスで指定してください。例えば、アプリケーションフォルダ直下のJSPフォルダにHTML/JSPファイルが存在する場合は、「../acf/apc」を指定してください。HTML/JSPファイルがインクルードされている場合には、インクルード元のHTML/JSPファイルが存在するフォルダを基点とした相対パスで指定してください。 URLには、任意のクエリ文字列やURLリライティングで使用するセションIDを付加することができます。詳細は、「3.9 リクエスト送信時のURLについて」を参照してください。	string	不可

プロパティ名	概要	型	省略可否
callback	レスポンスハンドラ(正常終了時のコールバック) ビジネスメソッドからの戻り値がある場合は、第1引数に渡されます。voidの場合は、パラメーターなしで呼ばれます。	function	不可
errorHandler	エラーハンドラ(エラー発生時のコールバック) 第1引数にエラーオブジェクトが渡されます。 詳細は、「3.10 通信フレームワーク内のエラー」を参照してください。	function	可
timeout	タイムアウト時間(ミリ秒) サーバからのレスポンスが完了するまでの最大待ち時間です。この時間を超えると、エラーハンドラ呼出しとなります。 省略値は、120000(120秒)です。 なお、0を指定することはできません。	number	可
preInvoke	メソッド呼出し直前に呼ばれるコールバック 引数はありません。	function	可
postInvoke	メソッド呼出し完了直後に呼ばれるコールバック 引数はありません。	function	可

以下に、通信設定オブジェクトのプロパティに関する注意事項を示します。

- 各種コールバック関数がnullの場合、コールバック呼出しは行われません。
- プロパティの型がfunctionとなっているものは、JavaScriptの関数を指定します。
  以下に、callbackプロパティにレスポンスハンドラを記述する例を示します。

```
var option = {
  ...,
  // サーバからの戻り値を引数に指定
  // 引数の名前は任意
  callback:function(res){
    // サーバからの戻り値に対する処理を記述
    ...;
  }
};
```

UjiRequest.send関数の記述例

以下に、UjiRequest.send関数の記述例を示します。
以下の例では、Ajaxフレームワーク環境定義ファイルに「databean」の名前を持つIDで対応付けられたデータBeanにデータが格納されます。

```
function clickExecuteButton() {
    // requestParam
    var reqParam = {
        beanId:'databean',  // データBean IDを指定
        verb:'execute'      // コマンド名を指定
    };
    // option
    var option = {
        url:'acf/apc',
        callback:function(res){
        }
    };
    UjiRequest.send(data,reqParam,option);
}
```

UjiRequest.send関数が通知する例外

UjiRequest.send関数が通知する例外については、「J.2.2 通信フレームワーク(JavaScript)に関するメッセージ」を参照してください。

3.2.7 コールバック関数

ここでは、ビジネスメソッドの実行時に指定するコールバック関数について説明します。

callback

callbackには、正常終了時に実行する処理または関数を指定します。
callbackの引数には、サーバからの戻り値が渡されます。サーバからの戻り値がない場合には、引数なしで呼び出されます。
戻り値となるデータBeanには、送信データを受け取るデータBeanとは異なるデータBeanを指定することができます。この際、戻り値となるデータBeanについては、Ajaxフレームワーク環境定義ファイルのデータBeanの定義(dataBeans)は必要ありません。Ajaxフレームワーク環境定義ファイルは、デフォルトの設定以外でデータ型変換を行いたい場合に定義する必要があります。

以下に、callbackで、mypkg.CustomerBeanクラスを戻り値として受け取る例を示します。

- 戻り値となるデータBeanの記述例

```
package mypkg;
import com.fujitsu.uji.DataBean;

public class CustomerBean extends DataBean
{
    protected String customerId;
    protected String customerName;
    public String getCustomerId() {
        return customerId;
    }
    public void setCustomerId(String customerId) {
        this.customerId = customerId;
    }
    public String getCustomerName() {
        return customerName;
    }
    public void setCustmerName(String customerName) {
        this.customerName = customerName;
    }
}
```

- ビジネスクラスの記述例

```
package mypkg;
import com.fujitsu.uji.DispatchContext;
import com.fujitsu.uji.GenericHandler;

public class MyHandler extends GenericHandler
{
    public boolean init() {
        return true;
    }

    public Object doSomething(DispatchContext context, BodyBean dataBean) {
        // 業務ロジックを記述
        (省略)

        // 戻り値はreturnで記述
        CustomerBean retBean = new CustomerBean();

        // customerId、customerNameの設定
        (省略)

        return retBean;
```

```
    }
}
```

- callbackの記述例

```
var option = {
  ...,
  callback:function(res){
    customerModel.setProperty("cid", res.customerId);
    customerModel.setProperty("cname", res.custmerName);
  }
};
```

errorHandler

errorHandlerには、エラー発生時に実行する処理または関数を指定します。
errorHandlerの引数には、エラーオブジェクトが渡されます。詳細は、「3.10 通信フレームワーク内のエラー」を参照してください。

以下に、エラー発生時にエラーコードとエラーメッセージをalertで表示する場合の記述例を示します。

```
var option = {
  ...,
  errorHandler:function(err){
    alert( "Application Error ( " + err.errorCode + " : " + err.message + " )" );
  }
};
```

preInvoke

preInvokeには、メソッド呼出し直前に実行する処理または関数を指定します。

以下に、preInvokeが呼び出される際に、処理時刻をデバッグメッセージに出力する場合の記述例を示します。

```
var option = {
  ...,
  preInvoke:function(){
    var lcDate = new Date();
    RCF.debug( lcDate.toLocaleString() + " : preInvoke " );  }
};
```

postInvoke

postInvokeには、メソッド呼出し直後に実行する処理または関数を指定します。

以下に、postInvokeが呼び出される際に、処理時刻をデバッグメッセージに出力する場合の記述例を示します。

```
var option = {
  ...,
  postInvoke:function(){
    var lcDate = new Date();
    RCF.debug( lcDate.toLocaleString() + " : postInvoke " );  }
};
```

3.2.8 Apcoordinatorとのデータ Beanの共有

AjaxフレームワークアプリケーションとApcoordinatorアプリケーションとの間で、データBeanを共有することができます。データBeanを共有することにより、Apcoordinatorアプリケーションで利用するデータBeanの情報を、Ajaxフレームワークアプリケーションで更新することができます。

ここでは、データBeanを共有するための方法について説明します。

データBeanの作成

データBeanの作成方法は、通常の作成方法と同じです。詳細は、「3.2.1 データBeanの作成」を参照してください。

ビジネスクラスの作成

ビジネスクラスの作成方法は、通常の作成方法と同じです。詳細は、「3.2.3 ビジネスクラスの作成」を参照してください。

データBeanを共有するためには、DispatchContextクラスのsetResponseBeanメソッドを利用します。setResponseBeanメソッドの領域名に指定した名前を、ビジネスメソッド実行時のbeanIdリクエストパラメーターに指定するデータBean IDとして使用することにより、Apcoordinatorが保持するデータBeanにデータが格納されます。

以下に、Apcoordinatorアプリケーションとデータ Beanを共有する場合の、ビジネスクラスの記述例を示します。
以下の例では、Apcoordinatorの表示画面の領域名(body)に設定されたデータBean(BodyBean)を共有します。

```
package mypkg;
import com.fujitsu.uji.DispatchContext;
import com.fujitsu.uji.GenericHandler;

public class MyHandler extends GenericHandler
{
    public boolean init() {
        return true;
    }

    /**
     * Apcoordinatorの初期画面表示の処理
     */
    public void startup(DispatchContext context) {
        BodyBean dataBean = new BodyBean();
        // 初期データの設定
        (省略)
        // 領域名(body)にデータBean(BodyBean)を設定
        context.setResponseBean("body", dataBean);
    }

    /**
     * Apcoordinator連携の処理
     */
    public Object doSomething(DispatchContext context, BodyBean dataBean) {
        // 業務ロジックを記述
        (省略)

        // 戻り値はreturnで記述
        NextBean retBean = new NextBean();
        return retBean;
    }
}
```

コマンドマップの定義

作成したビジネスクラスのメソッドを呼び出すために、コマンドマップを定義します。
コマンドマップの記述形式については、「3.2.4 コマンドマップの定義」を参照してください。

以下に、Apcoordinatorアプリケーションとデータ Beanを共有する場合のコマンドマップの定義例を示します。

```
# commands.map
## Apcoordinator
;=mypkg.MyHandler.startup
## Ajax
mypkg.BodyBean;execute=mypkg.MyHandler.doSomething
```

ビジネスメソッドの実行

Apcoordinatorアプリケーションとデータ Beanを共有する場合には、UjiRequest.send関数のbeanIdリクエストパラメーターにデータBeanを関連付けたデータBean IDを指定します。
ここで指定するデータBean IDは、DispatchContextクラスのsetResponseBeanメソッドに指定された領域名となります。

以下に、Apcoordinatorアプリケーションとデータ Beanを共有する場合の、UjiRequest.send関数の記述例を示します。
以下の例では、Apcoordinatorの表示画面の領域名(body)に関連付けられたデータBeanにデータが格納されます。データBeanの関連付けについては、「3.2.3 ビジネスクラスの作成」を参照してください。

```
function clickExecuteButton() {
    // requestParams
    var reqParam = {
        beanId:'body',   // 領域名を指定
        verb:'execute'   // コマンド名を指定
    };
    // option
    var option = {
        url:'acf/apc',
        callback:function(res){
        }
    };
    UjiRequest.send(data,reqParam,option);
}
```

参考

Apcoordinatorアプリケーションとデータ Beanを共有する場合には、Ajaxフレームワーク環境定義ファイルのdataBean要素の指定を省略できます。

3.2.9 Apcoordinator連携機能の注意事項

ここでは、Apcoordinator連携機能を利用する際の注意事項を説明します。

Apcoordinator連携機能の注意事項

- **HTTPの最大同時接続数**
  HTTPの最大同時接続数の上限を超えて、ビジネスメソッドの実行を行った場合には、先行するビジネスメソッドの実行が完了し、レスポンスが返却されるまで、ブラウザによって待ち合わせが行われます。したがって、HTTPの最大同時接続数の上限を超えてビジネスメソッドの実行を行う場合には、待ち合わせが行われることを考慮してアプリケーションを設計してください。
  なお、Internet ExplorerおよびFirefoxにおけるHTTPの最大同時接続数のデフォルト値は6個です。

データBeanを共有の注意事項

- **データBeanのバックアップ**
  Apcoordinatorには、データBean IDに関連付けられたデータBeanを保存するためのスタックがあり、Apcoordinatorのリクエストごとにデータ Beanが保存されています。
  com.fujitsu.uji.http.HttpDispatchContextクラスのpopBeanBackupメソッドを使用すると、以前に使用したデータBeanをデータBean IDに再割り当てできます。
  Apcoordinatorが管理するスタックには、データBeanのコピーではなく、データBeanそのものが保存されます。したがって、Ajaxフレームワークでデータ Beanを共有し、そのデータBeanのデータを書き換えた場合、popBeanBackupメソッドを使って戻った画面には、書き換えられたデータが表示されます。

- **<uji:useBean>タグのrequestアトリビュート**
  Apcoordinatorアプリケーションの画面表示の際、<uji:useBean>タグのrequestアトリビュートには、trueを指定する必要があります。
  requestアトリビュートがtrueの場合は、データBeanがApcoordinator内で保持されるので、Ajaxフレームワークで共有することができます。
  requestアトリビュートがfalseの場合は、Apcoordinatorアプリケーションの画面表示において、HTTPレスポンス返却後、データBeanへの参照は失われます。このため、UjiRequest.send関数のbeanIdリクエストパラメーターで指定するデータBean IDでは、データBeanを参照(共有)することはできません。UjiRequest.send関数で送信されたデータを受け取るには、Ajaxフレームワーク環境定義ファイルのdataBean要素で、リクエストスコープのデータBeanを定義する必要があります。

- **データBeanをリクエストスコープで使用する場合**
  必ず、Ajaxフレームワーク環境定義ファイルに、データBean IDおよびデータBeanの定義を記述してください。

Ajaxフレームワーク環境定義ファイルとDispatchContext.setResponseBeanに同一のデータBean IDが定義されている場合、Ajaxフレームワーク環境定義ファイルに定義されたデータBeanが有効になります。

- **データBeanをセションスコープで使用する場合**
  Ajaxフレームワーク環境定義ファイルとDispatchContext.setResponseBeanに同一のデータBean IDが定義されている場合、DispatchContext.setResponseBeanに定義されたデータBeanが有効になります。
  Ajaxフレームワーク環境定義ファイルだけに定義されている場合は、UjiRequest.send関数を利用してデータBeanを更新したときに、データBeanのインスタンスが生成されます。

- **Ajaxフレームワーク環境定義ファイルへの定義省略について**
  Ajaxフレームワーク環境定義ファイルのdataBean要素の指定を省略できるのは、DispatchContext.setResponseBeanで定義されたデータBean IDだけです。
  Ajaxフレームワーク環境定義ファイルのdataBean要素の指定を省略した場合、データBeanはセションスコープになります。

参考

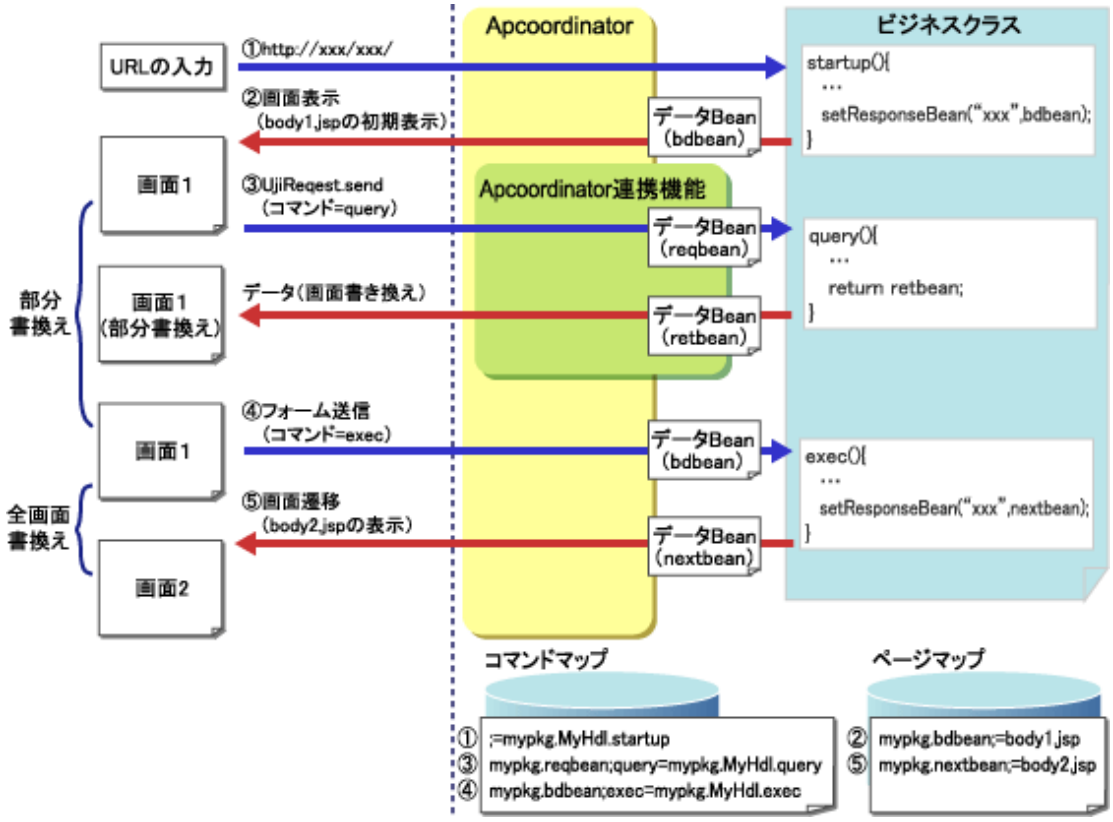
Ajaxフレームワーク環境定義ファイルのデータBeanの定義方法については、「A.5 データBeanの定義(dataBeans)」を参照してください。
ApcoordinatorのデータBeanのバックアップおよび<uji:useBean>タグのrequestアトリビュートについては、Interstage Application ServerのApcoordinatorのマニュアルを参照してください。

3.2.10 初期画面表示と画面遷移

Apcoordinator連携機能を利用する際には、初期画面表示と画面遷移はApcoordinatorの機能を利用して行います。
Ajaxフレームワークアプリケーションは、それぞれの画面の部分書換えを行います。

以下に、画面遷移とアプリケーションの関係を示します。

図3.3 画面遷移とアプリケーションの関係



3.3 サーブレット連携機能

サーブレット連携機能を利用すると、クライアントのJavaScriptアプリケーションからサーバのビジネスロジック(サーブレット)を呼び出すことができます。

サーブレット連携機能では、アプリケーションの自由度と利便性を考慮して、以下の2つの通信方式を想定した機能を提供します。

- **汎用通信方式**
  アプリケーションの自由度を重視した方式です。
  アプリケーションは、クライアントのJavaScriptアプリケーションからサーバのビジネスロジックへ送信するオブジェクトに対し独自の符号化を実施できるなど、JavaScriptアプリケーションとビジネスロジック間の通信を柔軟に制御することができます。
  アプリケーションは、JavaScriptアプリケーションとビジネスロジック間の通信を、Webブラウザ とWebサーバの機能を使用し実装します。JavaScriptアプリケーションのデータであるオブジェクトとビジネスロジックのデータであるJavaBeanを相互変換するために、サーブレット連携機能の提供するAPIを使用します。
  サーブレット連携機能の汎用通信方式の詳細は、「3.4 サーブレット連携機能(汎用通信方式)」を参照してください。

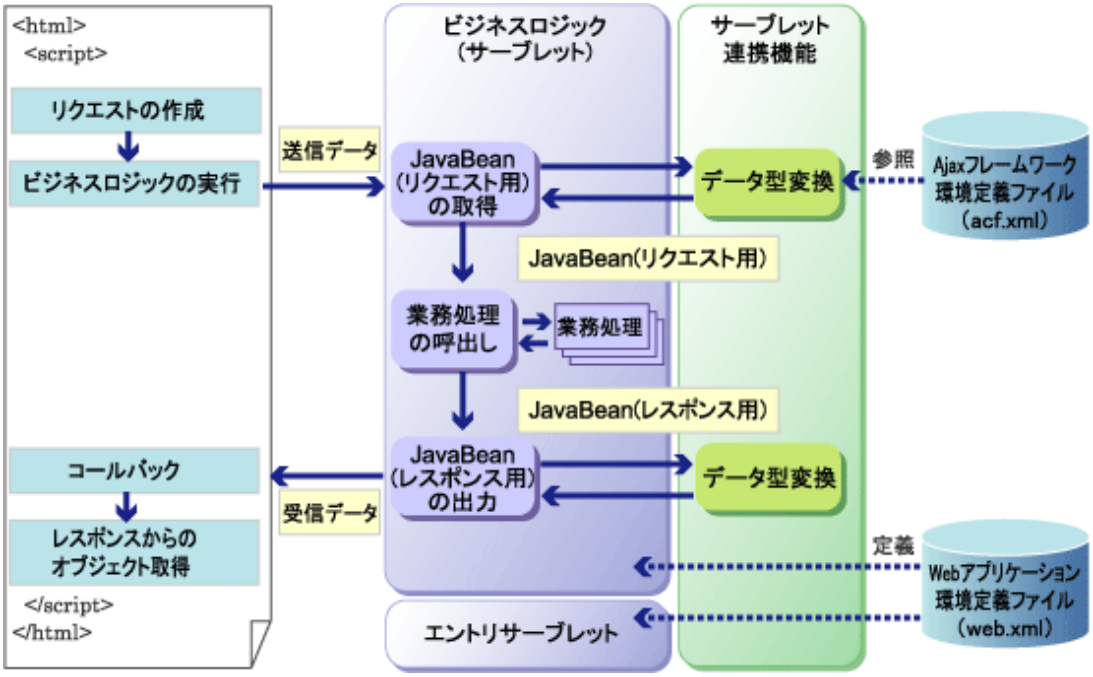
- **簡易通信方式**
  アプリケーションを容易に構築するために利便性を重視した方式です。
  アプリケーションは、使用するWebブラウザの違いに対する考慮や、JavaScriptアプリケーションのオブジェクトとビジネスロジックのJavaBeanを相互変換する必要がなく、アプリケーションを容易に構築することができます。
  アプリケーションは、JavaScriptアプリケーションとビジネスロジック間の通信にサーブレット連携機能の簡易通信機能を使用します。JavaScriptアプリケーションのデータであるオブジェクトとビジネスロジックのデータであるJavaBeanは、サーブレット連携機能が提供する簡易通信機能の中で相互変換します。
  サーブレット連携機能の簡易通信方式の詳細は、「3.5 サーブレット連携機能(簡易通信方式)」を参照してください。

3.4 サーブレット連携機能(汎用通信方式)

ここでは、サーブレット連携機能の汎用通信方式について説明します。

以下に、汎用通信方式における処理の流れを示します。

図3.4 汎用通信方式の処理の流れ



リクエストの作成・ビジネスロジックの実行

JavaScriptオブジェクトを作成してデータを格納します。
作成したJavaScriptオブジェクトをサーブレット連携機能が提供する関数を使用して文字列に変換します。また、XMLHttpRequestオブジェクトを使って、リクエストを送信してビジネスロジックを呼び出します。

ビジネスロジック呼出し時には、変換した文字列をリクエストパラメーターとして送信します。同時に、サーバの処理結果を受け取るコールバックなどを指定します。
ビジネスロジックの実行の詳細は、「3.4.5 ビジネスロジックの実行」を参照してください。

ビジネスロジック

クライアントから要求された処理を実行するサーバ側のアプリケーションです。
ビジネスロジックは、JavaScriptアプリケーションから要求された処理を実行します。また、ビジネスロジックでの処理結果をJavaBeanに格納して、JavaScriptアプリケーションに返却します。ビジネスロジックはサーブレットとして作成します。
詳細は、「3.4.2 ビジネスロジックの作成」を参照してください。

JavaBean(リクエスト用)の取得

ビジネスロジックでは、リクエストの受信後、JavaScriptアプリケーションからの送信データを取り出してJavaBeanに変換します。変換には、サーブレット連携機能が提供するAPIを使用します。変換は、データ型変換機能によって行われます。
データ型変換機能の詳細は、「3.6 データ型変換機能」を参照してください。

業務処理の呼出し

リクエストの内容に従って、適切な業務処理を呼び出します。

JavaBean(レスポンス用)の出力

業務処理の処理結果が格納されているJavaBeanを、JavaScriptアプリケーションで取り扱うことのできる受信データに変換して、レスポンスとして返却します。変換には、サーブレット連携機能が提供するAPIを使用します。変換は、データ型変換機能によって行われます。
データ型変換機能の詳細は、「3.6 データ型変換機能」を参照してください。

コールバック・レスポンスからのオブジェクト取得

ビジネスロジックの実行の処理結果を受け取るコールバック関数です。
レスポンスから受信データを取り出して、JavaScriptのオブジェクトに変換します。
コールバックの詳細は、「3.4.5 ビジネスロジックの実行」を参照してください。

エントリサーブレットとWebアプリケーション環境定義ファイル(web.xml)

JavaScriptアプリケーションとビジネスロジックとの間でデータを送受信するためには、通信フレームワークの提供するcom.fujitsu.interstage.rcf.AcfServletクラスを継承したユーザー定義エントリサーブレットを作成します。また、Webアプリケーション環境定義ファイル(web.xml)にユーザー定義エントリサーブレットのマッピング情報を指定します。
エントリサーブレットとWebアプリケーション環境定義ファイルの詳細は、「3.4.4 エントリサーブレットの作成」を参照してください。
なお、Webアプリケーション環境定義ファイルには、Servletの仕様に従って、ビジネスロジックとして作成したサーブレットなどの定義も記述してください。

Ajaxフレームワーク環境定義ファイル(acf.xml)

JavaScriptとJavaのデータ型変換の変換規則を定義します。
Ajaxフレームワーク環境定義ファイルの詳細は、「3.4.3 Ajaxフレームワーク環境定義ファイルの設定」を参照してください。

JavaScriptアプリケーションとビジネスロジックとの間のデータの送受信方法は、アプリケーションで任意に決めることができます。
以降は、以下の方法でデータを送受信することを前提に説明します。

- JavaScriptオブジェクトを文字列に変換してリクエストパラメーター「obj」に設定し、ビジネスロジックに送信します。
- 業務処理の処理結果であるJavaBeanを文字列に変換してレスポンスのボディに設定し、ビジネスロジックから返却します。

3.4.1 JavaBeanの作成

ここでは、JavaScriptアプリケーションからの送信データを受け取るJavaBean、JavaScriptアプリケーションに返却するためのデータを格納するJavaBeanを作成します。

以下に、JavaBeanの記述例を示します。

・ 送信データを受け取るJavaBean

```
package mypkg;

public class RequestBean {
    private String key;
    public String getKey() {
        return key;
    }

    public void setKey(String key) {
        this.key = key;
    }
}
```

・ データを格納するJavaBean

```
package mypkg;

public class ResponseBean {
    private String message;
    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

3.4.2 ビジネスロジックの作成

ここでは、JavaScriptアプリケーションからの送信データを受け取って処理を行うビジネスロジックの作成方法について説明します。

ビジネスロジックは、サーブレットとして作成します。

JavaBean(リクエスト用)の取得

JavaBean(リクエスト用)の取得は、以下の手順で行います。

1. ServletContextクラスのインスタンスを取得します。
2. com.fujitsu.interstage.rcf.http.RcfServletHelperクラスのgetRcfContextFromメソッドを使用して、ServletContextからcom.fujitsu.interstage.rcf.RcfContextクラスのインスタンスを取得します。
3. com.fujitsu.interstage.rcf.converter.RcfBeanConverterクラスのインスタンスを生成します。コンストラクタの引数にはRcfContextを指定します。なお、性能を考慮する場合は、生成したインスタンスをアプリケーション全体で共用することを推奨します。
4. リクエストから文字列を取り出します。
5. JavaBeanのインスタンスを用意します。
6. リクエストから取り出した文字列を、RcfBeanConverterクラスのdecodeメソッドで変換し、JavaBeanのインスタンスに格納します。

業務処理の呼出し

受け取ったリクエストパラメーターに応じて、各業務処理を振り分けて実行します。
業務処理では、処理結果をJavaBeanに格納します。

JavaBean(レスポンス用)の出力

RcfBeanConverterクラスのencodeメソッドで処理結果であるJavaBeanを文字列(JSON形式)に変換します。
変換した処理結果は、レスポンス(javax.servlet.http.HttpServletResponse)から出力ストリーム(java.io.PrintWriter)を取得し、出力ストリームへ出力します。

以下に、ビジネスロジックの記述例を示します。

```
package mypkg;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.fujitsu.interstage.rcf.RcfContext;
import com.fujitsu.interstage.rcf.converter.RcfBeanConverter;
import com.fujitsu.interstage.rcf.http.RcfServletHelper;

public class MyHandler extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {

        // JavaBean(リクエスト用)の取得
        ServletContext context = getServletContext();
        RcfContext rcfContext = RcfServletHelper.getRcfContextFrom(context);
        RcfBeanConverter conv = new RcfBeanConverter(rcfContext);
        String reqStr = req.getParameter("obj");
        RequestBean reqBean = new RequestBean();
        try {
            conv.decode(reqStr, reqBean);
        } catch (Exception e) {
            // 例外発生時の処理
            ...;
        }
        // 業務ロジックの呼出し
        ResponseBean resBean = null;
        String cmd = req.getParameter("command");
        if ("init".equals(cmd)) {
            // 初期処理の呼出し
            resBean = init(reqBean);
        } else if ("execute".equals(cmd)) {
            // 実行処理の呼出し
            ...;
        }
        // JavaBean(レスポンス用)の出力
        res.setContentType("application/json; charset=UTF-8");
        res.setStatus(HttpServletResponse.SC_OK);
        try {
            String resStr = conv.encode(resBean);
            PrintWriter wrt = res.getWriter();
            wrt.print(resStr);
        } catch (Exception e) {
            // 例外発生時の処理
            ...;
        }
    }
}
```

3.4.3 Ajaxフレームワーク環境定義ファイルの設定

Ajaxフレームワーク環境定義ファイルでは、データ型変換機能に関する定義を指定します。デフォルトの変換規則から変更する必要がない場合は、conversion要素を省略できます。

Ajaxフレームワーク環境定義ファイルの指定方法については、「A.6 コンバータ設定の定義(conversion)」を参照してください。

注意

サーブレット連携機能では、dataBeans要素を使用しません。

3.4.4 エントリサーブレットの作成

サーブレット連携機能を利用して、クライアントとサーバとの間でデータの送受信を行うためには、com.fujitsu.interstage.rcf.AcfServletクラスを継承したユーザー定義エントリサーブレットを作成し、Webアプリケーション環境定義ファイル(web.xml)にユーザー定義エントリサーブレットのマッピング情報を指定する必要があります。

エントリサーブレットは、サーブレット連携機能の初期化とイベントログ機能で使用されます。
詳細は、「5.3 エントリサーブレット」を参照してください。

3.4.5 ビジネスロジックの実行

ここでは、ビジネスロジックを実行して結果を受信するためのJavaScriptアプリケーションの作成方法について説明します。

リクエストの作成

リクエストの作成は、以下の手順で行います。

1. JavaScriptオブジェクトを作成してデータを格納します。
2. 作成したJavaScriptオブジェクトを文字列に変換します。オブジェクトから文字列への変換には、RcfObjectConverter.encode関数を使用します。
3. 変換した文字列をリクエストパラメーターの値として使えるように、encodeURIComponent関数でエンコードします。
4. エンコードした文字列を、ビジネスロジックの実行時に指定するリクエストパラメーターに設定します。

ビジネスロジックの実行

ビジネスロジックを実行するには、XMLHttpRequestクラスオブジェクトを生成し、onreadystatechangeプロパティなどの必要なプロパティを設定してリクエストを送信します。

リクエストの送信は、XMLHttpRequestオブジェクトクラスのsend関数を使用します。

コールバック

onreadystatechangeプロパティに設定した関数で、レスポンスの受信完了を待ちます。

レスポンスからのオブジェクト取得

レスポンスからのオブジェクトの取得は、以下の手順で行います。

1. 受信したレスポンスのボディから文字列を取得します。この文字列は、ビジネスロジックから返却されたJSON形式の文字列です。
2. 文字列がJSON形式として正しいかどうかをチェックします。
3. 文字列をevalでオブジェクトに変換します。

参考

evalでオブジェクトに変換する前に、不当なコードが文字列に含まれていないことをチェックしてください。
JSON形式とそのチェック方法については、RFC 4627を参照してください。

以下に、JavaScriptアプリケーションの記述例を示します。

```
/* イベント処理 */
function onClicked() {
    try {
```

```
        // リクエストの作成
        var obj = new Object();
        obj.key = ...;
        var str = RcfObjectConverter.encode(obj);
        str = encodeURIComponent(str);
        var param = 'obj=' + str + '&command=init';
        // XMLHttpRequestの生成
        var httpReq = new XMLHttpRequest();
        // コールバックの設定
        httpReq.onreadystatechange = function(res) {
            if ((httpReq.readyState == 4) &&
                (httpReq.status == 200)) {
                obj = receive(httpReq.responseText);
                // 受信したオブジェクトの処理
                ...;
            }
        }
        // ビジネスロジックの実行
        httpReq.open('post', url, true);
        httpReq.setRequestHeader('Content-Type',
            'application/x-www-form-urlencoded; charset=UTF-8');
        httpReq.send(param);
    } catch (e) {
        // エラー処理
        ...;
    }
}
/* レスポンス受信 */
function receive(res) {
    // レスポンスからオブジェクト取得
    var obj = !(/[^,:{}¥[¥]0-9.¥-+Eaeflnr-u ¥n¥r¥t]/.test(
        res.replace(/"(¥¥.|[^"¥¥])*"/g, ''))) &&
        eval('(' + res + ')');
    if (!obj) {
        // エラー処理
        ...;
        return null;
    }
    return obj;
}
```

ビジネスロジックの実行で使用するJavaScript用の関数を以下に説明します。

RcfObjectConverter.encode関数

JavaScriptアプリケーションの送信データであるオブジェクトを文字列に変換する関数です。
オブジェクトは、Ajaxフレームワークが規定する独自形式で文字列に変換されますが、アプリケーションがこの形式を意識する必要はありません。変換した文字列は、ビジネスロジックの処理で、com.fujitsu.interstage.rcf.converter.RcfBeanConverterクラスのdecodeメソッドでJavaBeanに変換できます。

RcfObjectConverter.encode関数の記述形式

以下に、RcfObjectConverter.encode関数の記述形式を示します。

```
RcfObjectConverter.encode(dataObj);
```

- **dataObj**
  変換するオブジェクトです。
- **戻り値**
  オブジェクトを変換した結果の文字列です。

RcfObjectConverter.encode関数が通知する例外

RcfObjectConverter.encode関数が通知する例外については、「J.2.2 通信フレームワーク(JavaScript)に関するメッセージ」を参照してください。

3.4.6 サーブレット連携機能(汎用通信方式)の注意事項

ここでは、サーブレット連携機能の汎用通信方式を利用する際の注意事項を説明します。

- **HTTPの最大同時接続数**
  HTTPの最大同時接続数の上限を超えて、ビジネスロジックの実行を行った場合には、先行するビジネスロジックの実行が完了し、レスポンスが返却されるまで、ブラウザによって待ち合わせが行われます。したがって、HTTPの最大同時接続数の上限を超えてビジネスロジックの実行を行う場合には、待ち合わせが行われることを考慮してアプリケーションを設計してください。
  なお、Internet ExplorerおよびFirefoxにおけるHTTPの最大同時接続数のデフォルト値は6個です。

3.5 サーブレット連携機能(簡易通信方式)

ここでは、サーブレット連携機能の簡易通信方式について説明します。

簡易通信方式には、非同期通信と同期通信があります。

- **簡易通信方式(非同期通信)**
  JavaScriptアプリケーションとビジネスロジック間を非同期で通信します。
  リクエストの送信後は、ビジネスロジックの処理が完了しなくても、JavaScriptアプリケーションに制御が戻ります。ビジネスロジックをバックグラウンドで動作させる場合に有効な方式です。
  ビジネスロジックの処理結果は、あらかじめ登録しておくコールバック関数に渡されます。

- **簡易通信方式(同期通信)**
  JavaScriptアプリケーションとビジネスロジック間を同期で通信します。
  リクエストを送信後は、ビジネスロジックの処理が完了するまでJavaScriptアプリケーションに制御が戻りません。ビジネスロジックの処理結果が利用者の操作に影響する場合に有効な方式です。
  ビジネスロジックの処理結果は、ビジネスロジックの呼出しに使用する関数の戻り値として、JavaScriptアプリケーションに渡されます。

  注意

  簡易通信方式(同期通信)は、Internet Explorerのみで利用できます。

以下に、簡易通信方式における処理の流れを示します。

図3.5 簡易通信方式(非同期通信)の処理の流れ

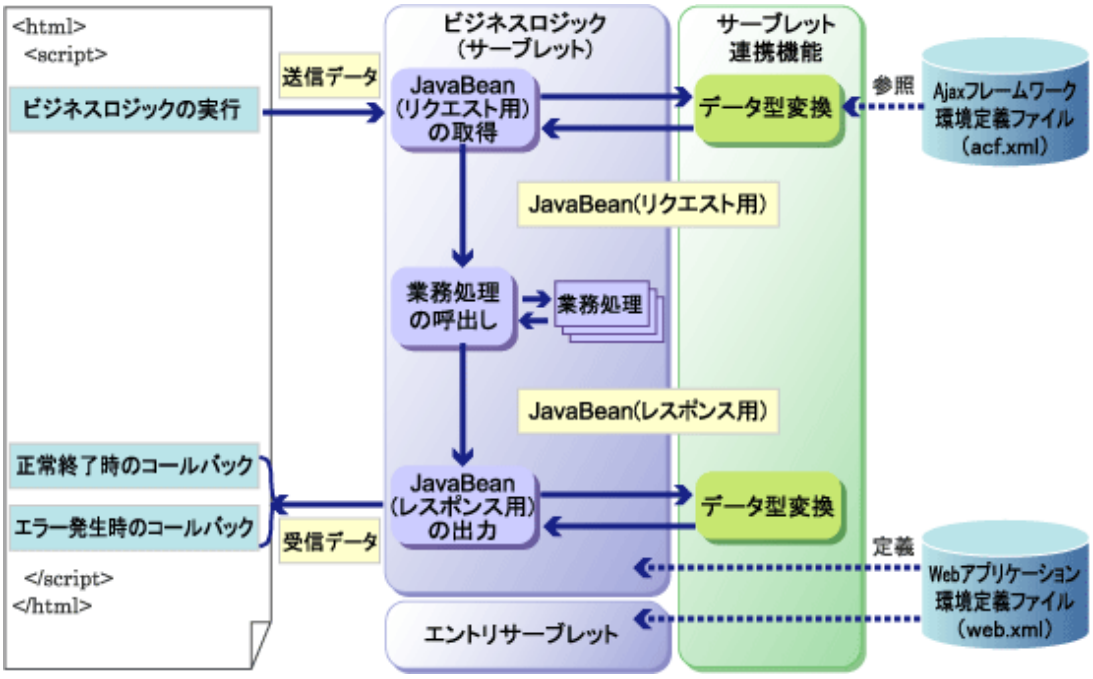
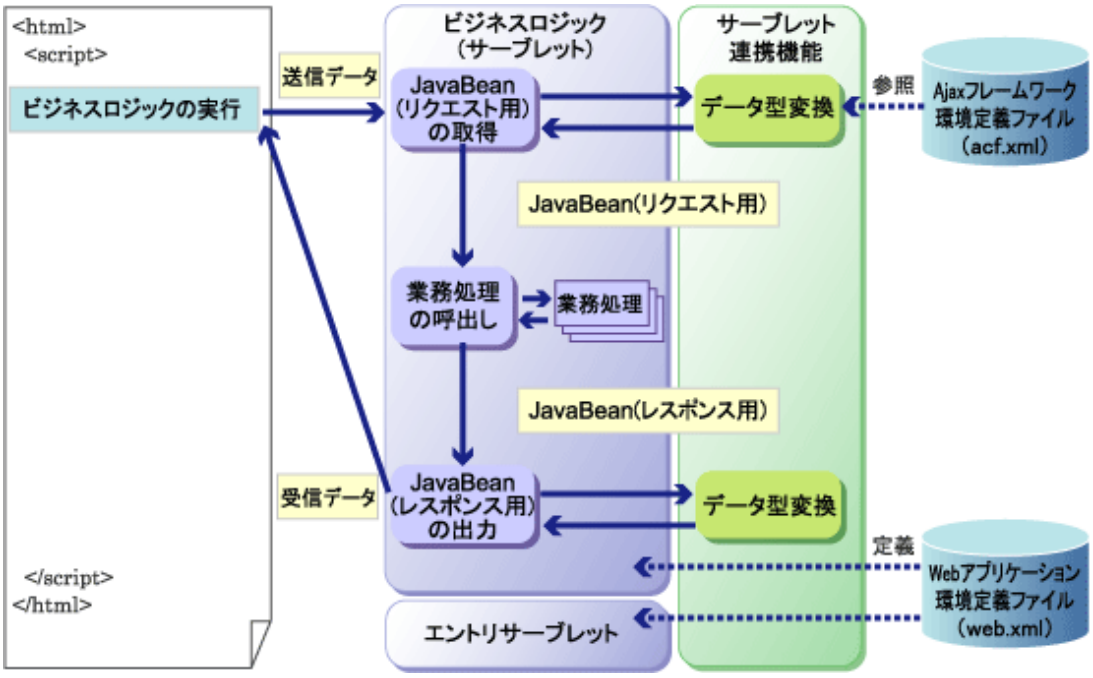


図3.6 簡易通信方式(同期通信)の処理の流れ



ビジネスロジックの実行

JavaScriptアプリケーションからビジネスロジックを呼び出します。送信データ、サーバの処理結果を受け取るコールバックなどを指定します。
ビジネスロジックの実行には、非同期通信と同期通信の2つの方式があります。
ビジネスロジックの実行の詳細は、「3.5.6 ビジネスロジックの実行(非同期通信)」または「3.5.7 ビジネスロジックの実行(同期通信)」を参照してください。

ビジネスロジック

クライアントから要求された処理を実行するサーバ側のアプリケーションです。ビジネスロジックは、JavaScriptアプリケーションから要求された処理を実行します。また、ビジネスロジックでの処理結果をJavaBeanに格納して、JavaScriptアプリケーションに返却します。

ビジネスロジックはサーブレットとして作成します。
詳細は、「3.5.3 ビジネスロジックの作成」を参照してください。

JavaBean(リクエスト用)の取得

ビジネスロジックでは、リクエストの受信後、JavaScriptアプリケーションからの送信データを取り出してJavaBeanに変換します。取得には、サーブレット連携機能が提供するAPIを使用します。変換は、データ型変換機能によって行われます。
データ型変換機能の詳細は、「3.6 データ型変換機能」を参照してください。

業務処理の呼出し

リクエストの内容に従って、適切な業務処理を呼び出します。

JavaBean(レスポンス用)の出力

業務処理の処理結果が格納されているJavaBeanを、JavaScriptアプリケーションで取り扱うことのできる受信データに変換して出力します。出力には、サーブレット連携機能が提供するAPIを使用します。変換は、データ型変換機能によって行われます。
データ型変換機能の詳細は、「3.6 データ型変換機能」を参照してください。

正常終了時のコールバック

ビジネスロジックの実行が正常に終了した場合に、処理結果を受け取るコールバック関数です。ビジネスロジックの実行時の引数として指定します。
正常終了時のコールバックの詳細は、「3.5.6 ビジネスロジックの実行(非同期通信)」および「3.5.8 コールバック関数 」を参照してください。

エラー発生時のコールバック

ビジネスロジックの実行時にエラーが発生した場合に、エラー結果を受け取るコールバック関数です。ビジネスロジック実行中のタイムアウトについても、このコールバック関数に通知されます。エラー発生時のコールバックは、正常終了時のコールバックと同様に、ビジネスロジックの実行時の引数として指定します。省略した場合には、エラー結果を受け取ることはできません。
エラー発生時のコールバックの詳細は、「3.5.6 ビジネスロジックの実行(非同期通信)」および「3.5.8 コールバック関数 」を参照してください。

エントリサーブレットとWebアプリケーション環境定義ファイル(web.xml)

JavaScriptアプリケーションとビジネスロジックとの間でデータを送受信するためには、通信フレームワークの提供するcom.fujitsu.interstage.rcf.AcfServletクラスを継承したユーザー定義エントリサーブレットを作成します。また、Webアプリケーション環境定義ファイル(web.xml)にユーザー定義エントリサーブレットのマッピング情報を指定します。
エントリサーブレットとWebアプリケーション環境定義ファイルの詳細は、「3.5.5 エントリサーブレットの作成」を参照してください。
なお、Webアプリケーション環境定義ファイルには、Servletの仕様に従って、ビジネスロジックとして作成したサーブレットなどの定義も記述してください。

Ajaxフレームワーク環境定義ファイル(acf.xml)

JavaScriptとJavaのデータ型変換の変換規則を定義します。
Ajaxフレームワーク環境定義ファイルの詳細は、「3.5.4 Ajaxフレームワーク環境定義ファイルの設定」を参照してください。

アプリケーションは、JavaBeanにIDを付けて管理する必要があります。
JavaScriptアプリケーションからビジネスロジックを実行するときには、送信データとともに、その送信データの格納先となるJavaBeanのIDを送信します。
データ型変換機能は、JavaScriptアプリケーションから送信されたIDに対応するJavaBeanをアプリケーションから取得して、変換後のデータを格納し、返却します。

3.5.1 JavaBeanの作成

ここでは、JavaScriptアプリケーションからの送信データを受け取るJavaBean、JavaScriptアプリケーションに返却するためのデータを格納するJavaBeanを作成します。

以下に、JavaBeanの記述例を示します。

・ 送信データを受け取るJavaBean

```
package mypkg;

public class RequestBean {
    private String key;
    public String getKey() {
        return key;
    }

    public void setKey(String key) {
        this.key = key;
    }
}
```

・ データを格納するJavaBean

```
package mypkg;

public class ResponseBean {
    private String message;
    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

3.5.2 JavaBeanの取得(コールバック)

データ型変換機能は、JavaBeanのIDに対応するJavaBeanをアプリケーションから取得するために、com.fujitsu.interstage.rcf.http.RcfServletHelperクラスのresolveBeanメソッドを呼び出します。
アプリケーションは、RcfServletHelperクラスを継承し、resolveBeanメソッドを実装する必要があります。

以下に、resolveBeanメソッドの記述例を示します。
この例では、JavaBeanのIDをキーに、JavaBeanを値とし、セションで管理する場合の処理を示します。

```
package mypkg;

import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import com.fujitsu.interstage.rcf.http.RcfServletHelper;

public class MyHelper extends RcfServletHelper {
    public MyHelper(ServletContext context) {
        super(context);
    }

    protected Object resolveBean (
        HttpServletRequest request, String beanId) {

        HttpSession session = request.getSession();
        Object bean = null;
        synchronized (session) {
            bean = session.getAttribute(beanId);
            if (bean == null) {
                // JavaBeanのIDに対応するJavaBeanを作成し、セションに登録する
                bean = new ...;
                session.setAttribute(beanId, bean);
            }
```

```
        }
        return bean;
    }
}
```

resolveBeanメソッドは、複数のスレッドから同時に呼び出されることがありますので、スレッドセーフとなるように作成してください。

3.5.3 ビジネスロジックの作成

ここでは、JavaScriptアプリケーションからの送信データを受け取って処理を行うビジネスロジックの作成方法について説明します。

ビジネスロジックは、サーブレットとして作成します。

JavaBean(リクエスト用)の取得

JavaBean(リクエスト用)の取得は、以下の手順で行います。

1. ServletContextクラスのインスタンスを取得します。
2. 「3.5.2 JavaBeanの取得(コールバック)」で作成したcom.fujitsu.interstage.rcf.http.RcfServletHelperクラスを継承したクラスのインスタンスを生成します。コンストラクタの引数には、ServletContextを指定します。なお、性能を考慮する場合は、生成したインスタンスをアプリケーション全体で共用することを推奨します。
3. 生成したインスタンスのgetRequestBeanメソッドを呼び出し、JavaBeanを取得します。このとき、引数にはjavax.servlet.http.HttpServletRequestクラスのインスタンスを指定します。ビジネスロジック呼出し時に送信データがない場合は、getRequestBeanメソッドからnullが返却されます。

業務処理の呼出し

受け取ったリクエストパラメーターに応じて、各業務処理を振り分けて実行します。
業務処理では、処理結果をJavaBeanに格納します。

JavaBean(レスポンス用)の出力

JavaBean(リクエスト用)の取得で作成したインスタンスのsetResponseBeanメソッドを呼び出し、処理結果のJavaBeanをレスポンスに出力します。
このとき、引数には以下の情報を指定します。

- javax.servlet.http.HttpServletRequestクラスのインスタンス
- javax.servlet.http.HttpServletResponseクラスのインスタンス
- 処理結果のJavaBean

JavaBeanがnullだった場合は、JavaScriptアプリケーションにnullが返却されます。

エラー情報の出力

業務処理でエラーが発生した場合などで、クライアントに例外を返却したい場合は、JavaBean(リクエスト用)の取得で作成したインスタンスのsetExceptionメソッドを呼び出し、レスポンスに例外の情報を出力します。
このとき、引数には以下の情報を指定します。

- javax.servlet.http.HttpServletRequestクラスのインスタンス
- javax.servlet.http.HttpServletResponseクラスのインスタンス
- 例外のインスタンス

例外の情報を出力すると、JavaScriptアプリケーションでは、errorHandlerコールバック関数が呼び出されます。errorHandlerの引数にはエラーコードRCF0907のエラーオブジェクトが渡されます。エラーオブジェクトのcauseプロパティには、setExceptionの引数に指定した例外の情報が格納されています。

なお、setResponseBeanメソッドでJavaBeanを返却しない場合は、setExceptionメソッドで例外を返却してください。また、setResponseBeanメソッドを実行して正常終了した場合は、setExceptionメソッドを呼び出さないでください。setExceptionメソッドを実行した場合は、setResponseBeanを呼び出さないでください。

以下に、ビジネスロジックの記述例を示します。

```
package mypkg;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.fujitsu.interstage.rcf.http.RcfServletHelper;

public class MyHandler extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException {

        // JavaBean(リクエスト用)の取得
        ServletContext context = getServletContext();
        RcfServletHelper helper = new MyHelper(context);
        RequestBean reqBean = null;
        try {
              reqBean = (RequestBean) helper.getRequestBean(req);
        } catch (Exception e) {
            // 例外発生時の処理
            ...;
        }
        // 業務ロジックの呼出し
        ResponseBean resBean = null;
        String cmd = req.getParameter("command");
        if ("init".equals(cmd)) {
            // 初期処理の呼出し
            resBean = init(reqBean);
        } else if ("execute".equals(cmd)) {
            // 実行処理の呼出し
            ...;
        }
        // JavaBean(レスポンス用)の出力
        try {
            helper.setResponseBean(req, res, resBean);
        } catch (Exception e) {
            // 例外発生時の処理
            ...;
            try {
                helper.setException(req, res, e);
            } catch (Exception ie) {
                // 例外発生時の処理
                ...;
            }
        }
    }
}
```

3.5.4 Ajaxフレームワーク環境定義ファイルの設定

Ajaxフレームワーク環境定義ファイルでは、データ型変換機能に関する定義を指定します。デフォルトの変換規則から変更する必要がない場合は、conversion要素を省略できます。

Ajaxフレームワーク環境定義ファイルの指定方法については、「A.6 コンバータ設定の定義(conversion)」を参照してください。

注意

サーブレット連携機能では、dataBeans要素を使用しません。

3.5.5 エントリサーブレットの作成

サーブレット連携機能を利用して、クライアントとサーバとの間でデータの送受信を行うためには、com.fujitsu.interstage.rcf.AcfServletクラスを継承したユーザー定義エントリサーブレットを作成し、Webアプリケーション環境定義ファイル(web.xml)にユーザー定義エントリサーブレットのマッピング情報を指定する必要があります。

エントリサーブレットは、サーブレット連携機能の初期化とイベントログ機能で使用されます。
詳細は、「5.3 エントリサーブレット」を参照してください。

3.5.6 ビジネスロジックの実行(非同期通信)

JavaScriptアプリケーションからビジネスロジックを非同期通信で呼び出すには、RcfRequestオブジェクトのsend関数を使用します。

RcfRequest.send関数

JavaScriptアプリケーションとビジネスロジック間を非同期で通信するための関数です。
JavaScriptアプリケーションの送信データであるオブジェクトを、非同期通信を使用して送信(POST)します。また、ビジネスロジックの処理結果をJavaScriptのオブジェクトとして受け取り、JavaScriptアプリケーションのコールバック関数にパラメーターとして渡します。

RcfRequest.send関数の記述形式

以下に、RcfRequest.send関数の記述形式を示します。

```
RcfRequest.send(dataObj, requestParams, option);
```

- **dataObj**
  サーバ側に送信するデータのオブジェクトです。
  データを送信しない場合は、nullを指定します。

- **requestParams**
  JavaBeanのIDや任意のリクエストパラメーターなど、ビジネスロジックの動作を制御するための値を格納したリクエストパラメーターオブジェクトです。
  リクエストパラメーターが不要な場合は、nullを指定します。

  以下に、リクエストパラメーターオブジェクトのプロパティを示します。

プロパティ名	概要	型	省略可否
beanId	送信データを格納するJavaBeanの識別子 アプリケーションがJavaBeanを特定するために使用します。	string	dataObjがnullの場合は省略可 それ以外の場合は省略不可
任意	アプリケーションが任意に設定できるリクエストパラメーター リクエストパラメーターの名前には、任意の名前を指定できます。ただし、以下の予約名を除きます。 予約名: beanId、call-id、version、「uji.」または「rcf.」で始まる名前	string	可

- **option**
  RcfRequestオブジェクトの動作を制御するための値を格納した通信設定オブジェクトです。
  optionにnullが指定された場合、または省略された場合は、通信せずにエラーが発生します。

  以下に、通信設定オブジェクトのプロパティを示します。

プロパティ名	概要	型	省略可否
url	サーブレットのURL 任意のクエリ文字列やURLリライティングで使用するセッションIDをURLに付加する方法については、「3.9 リクエスト送信時のURLについて」を参照してください。	string	不可

プロパティ名	概要	型	省略可否
	urlプロパティに指定するURLには、クエリ文字列やセションIDを含めないでください。		
callback	レスポンスハンドラ(正常終了時のコールバック) ビジネスロジックからの戻り値は、第1引数に渡されます。	function	不可
errorHandler	エラーハンドラ(エラー発生時のコールバック) 第1引数にエラーオブジェクトが渡されます。 詳細は、「3.10 通信フレームワーク内のエラー」を参照してください。	function	可
timeout	タイムアウト時間(ミリ秒) サーバからのレスポンスが完了するまでの最大待ち時間です。この時間を超えると、エラーハンドラ呼出しとなります。 省略値は、120000(120秒)です。 なお、0を指定することはできません。	number	可
preInvoke	メソッド呼出し直前に呼ばれるコールバック 引数はありません。	function	可
postInvoke	メソッド呼出し完了直後に呼ばれるコールバック 引数はありません。	function	可

以下に、通信設定オブジェクトのプロパティに関する注意事項を示します。

- 各種コールバック関数がnullの場合、コールバック呼出しは行われません。
- プロパティの型がfunctionとなっているものは、JavaScriptの関数を指定します。
  以下に、callbackプロパティにレスポンスハンドラを記述する例を示します。

```
var option = {
  ...,
  // サーバからの戻り値を引数に指定
  // 引数の名前は任意
  callback:function(res){
    // サーバからの戻り値に対する処理を記述
    ...;
  }
};
```

RcfRequest.send関数の記述例

以下に、RcfRequest.send関数の記述例を示します。
以下の例では、「databean」をIDとするJavaBeanにデータが格納されます。

```
function clickExecuteButton() {
    // dataObjの作成
    var dataObj = new Object();
    dataObj.key = ...;
    // requestParamsの作成
    var reqParam = {
        beanId :'databean',  // JavaBeanのIDを指定
        command:'init'       // コマンド名を指定(任意)
    };
    // optionの作成
    var option = {
        url:'entry',
        callback:function(res){
           // 受信したオブジェクト(res)の処理
           ...;
        }
    };
```

```
    try {
        RcfRequest.send(dataObj, reqParam, option);
    } catch (e) {
        alert("Application Error (" + e.message + ")");
    }
}
```

RcfRequest.send関数が通知する例外

RcfRequest.send関数が通知する例外については「J.2.2 通信フレームワーク(JavaScript)に関するメッセージ」および「J.3.9 サーブレット連携機能に関するメッセージ」を参照してください。

3.5.7 ビジネスロジックの実行(同期通信)

JavaScriptアプリケーションからビジネスロジックを同期通信で呼び出すには、RcfRequestオブジェクトのinvoke関数を使用します。

注意

ビジネスロジックの実行(同期通信)は、Internet Explorerのみで利用できます。

RcfRequest.invoke関数

JavaScriptアプリケーションとビジネスロジック間を同期で通信するための関数です。
JavaScriptアプリケーションの送信データであるオブジェクトを、同期通信を使用して送信(POST)します。また、ビジネスロジックの処理結果をJavaScriptのオブジェクトとして受け取り、関数の戻り値として返却します。

RcfRequest.invoke関数の記述形式

以下に、RcfRequest.invoke関数の記述形式を示します。

```
RcfRequest.invoke(dataObj, requestParams, option);
```

- **dataObj**
  サーバ側に送信するデータのオブジェクトです。
  データを送信しない場合は、nullを指定します。

- **requestParams**
  JavaBeanのIDや任意のリクエストパラメーターなど、ビジネスロジックの動作を制御するための値を格納したリクエストパラメーターオブジェクトです。
  リクエストパラメーターが不要な場合は、nullを指定します。

  以下に、リクエストパラメーターオブジェクトのプロパティを示します。

プロパティ名	概要	型	省略可否
beanId	画面データを格納するJavaBeanの識別子 アプリケーションがJavaBeanを特定するために使用します。	string	dataObjがnullの場合は省略可 それ以外の場合は省略不可
任意	アプリケーションが任意に設定できるリクエストパラメーター リクエストパラメーターの名前には、任意の名前を指定できます。ただし、以下の予約名を除きます。 予約名: beanId、call-id、version、「uji.」または「rcf.」で始まる名前	string	可

- **option**
  RcfRequestオブジェクトの動作を制御するための値を格納した通信設定オブジェクトです。
  optionにnullが指定された場合、または省略された場合は、通信せずにエラーが発生します。

  以下に、通信設定オブジェクトのプロパティを示します。

プロパティ名	概要	型	省略可否
url	サーブレットのURL 任意のクエリ文字列やURLリライティングで使用するセションIDをURLに付加する方法については、「3.9 リクエスト送信時のURLについて」を参照してください。 urlプロパティに指定するURLには、クエリ文字列やセションIDを含めないでください。	string	不可
preInvoke	メソッド呼出し直前に呼ばれるコールバック 引数はありません。	function	可
postInvoke	メソッド呼出し完了直後に呼ばれるコールバック 引数はありません。	function	可

  以下に、通信設定オブジェクトのプロパティに関する注意事項を示します。

  ー 各種コールバック関数がnullの場合、コールバック呼出しは行われません。

  ー プロパティの型がfunctionとなっているものは、JavaScriptの関数を指定します。

- **戻り値**
  ビジネスロジックの処理結果がある場合には、処理結果(オブジェクト)を返却します。
  ビジネスロジックの処理結果がnullの場合には、nullを返却します。

RcfRequest.invoke関数の記述例

以下に、RcfRequest.invoke関数の記述例を示します。
以下の例では、「databean」をIDとするJavaBeanにデータが格納されます。

```
function clickExecuteButton() {
    // dataObjの作成
    var dataObj = new Object();
    dataObj.key = ...;
    // requestParamsの作成
    var reqParam = {
        beanId :'databean',  // JavaBeanのIDを指定
        command:'init'       // コマンド名を指定(任意)
    };
    // optionの作成
    var option = {
        url:'entry'
    };
    try {
        var res = RcfRequest.invoke(dataObj, reqParam, option);
        // 受信したオブジェクト(res)の処理
        ...;
    } catch (e) {
        alert("Application Error (" + e.message + ")");
    }
}
```

RcfRequest.invoke関数が通知する例外

RcfRequest.invoke関数が通知する例外については、「J.2.2 通信フレームワーク(JavaScript)に関するメッセージ」および「J.3.9 サーブレット連携機能に関するメッセージ」を参照してください。

3.5.8 コールバック関数

ここでは、ビジネスメソッドの実行時に指定するコールバック関数について説明します。

callback

callbackには、正常終了時に実行する処理または関数を指定します。
callbackの引数には、サーバからの戻り値が渡されます。サーバからの戻り値がnullの場合は、nullが渡されます。

以下に、mypkg.CustomerBeanクラスを戻り値として受け取る例を示します。

- 戻り値となるJavaBeanの記述例

```
package mypkg;

public class CustomerBean
{
    private String customerId;
    private String customerName;
    public String getCustomerId() {
        return customerId;
    }
    public void setCustomerId(String customerId) {
        this.customerId = customerId;
    }
    public String getCustomerName() {
        return customerName;
    }
    public void setCustmerName(String customerName) {
        this.customerName = customerName;
    }
}
```

- callbackの記述例

```
var option = {
  ...,
  callback:function(res){
    customerModel.setProperty("cid", res.customerId);
    customerModel.setProperty("cname", res.custmerName);
  }
};
```

errorHandler

errorHandlerには、エラー発生時に実行する処理または関数を指定します。
errorHandlerの引数には、エラーオブジェクトが渡されます。詳細は、「3.10 通信フレームワーク内のエラー」を参照してください。

以下に、エラー発生時にエラーコードとエラーメッセージをalertで表示する場合の記述例を示します。

```
var option = {
  ...,
  errorHandler:function(err){
    alert( "Application Error ( " + err.errorCode + " : " + err.message + " )" );
  }
};
```

preInvoke

preInvokeには、リクエストを送信する直前に実行する処理または関数を指定します。

以下に、処理時刻をデバッグメッセージに出力する場合の記述例を示します。

```
var option = {
  ...,
  preInvoke:function(){
    var lcDate = new Date();
    RCF.debug( lcDate.toLocaleString() + " : preInvoke " );  }
};
```

postInvoke

postInvokeには、リクエストを送信後に実行する処理または関数を指定します。

postInvokeは、非同期通信と同期通信で呼び出されるタイミングが異なります。

- 非同期通信の場合:リクエストを送信した直後に呼び出されます。
- 同期通信の場合:レスポンスを受信した直後に呼び出されます。

postInvokeに指定した関数は、callbackに指定した関数より前に呼び出されます。

以下に、処理時刻をデバッグメッセージに出力する場合の記述例を示します。

```
var option = {
  ...,
  postInvoke:function(){
    var lcDate = new Date();
    RCF.debug( lcDate.toLocaleString() + " : postInvoke " );  }
};
```

3.5.9 サーブレット連携機能(簡易通信方式)の注意事項

ここでは、サーブレット連携機能の簡易通信方式を利用する際の注意事項を説明します。

- **HTTPの最大同時接続数**
  HTTPの最大同時接続数の上限を超えて、ビジネスロジックの実行を行った場合には、先行するビジネスロジックの実行が完了し、レスポンスが返却されるまで、ブラウザによって待ち合わせが行われます。したがって、HTTPの最大同時接続数の上限を超えてビジネスロジックの実行を行う場合には、待ち合わせが行われることを考慮してアプリケーションを設計してください。
  なお、Internet ExplorerおよびFirefoxにおけるHTTPの最大同時接続数のデフォルト値は6個です。

3.6 データ型変換機能

データ型変換機能は、JavaScriptからサーバ上のJavaのメソッドを実行する際に、引数のデータ(上り)と戻り値のデータ(下り)を、適切に利用できるように変換する機能です。

3.6.1 データ型変換機能の概要

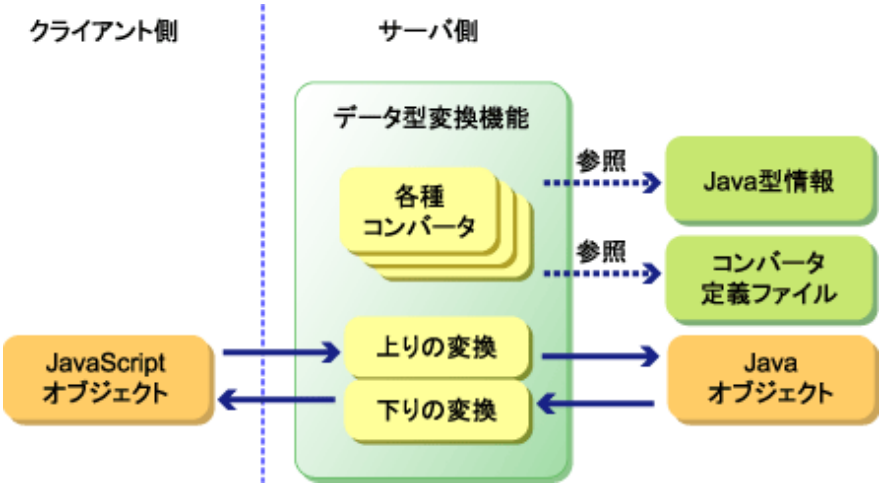
ここでは、データ型変換機能の概要として、以下の項目を説明します。

- データ型変換機能の構成要素
- データ型変換の例
- データ型変換の範囲
- データ型変換の例外

データ型変換機能の構成要素

以下の図に、データ型変換機能の構成要素を示します。

図3.7 データ型変換機能の構成要素



- データ型変換機能
  JavaScriptとJavaのデータ型変換を行います。変換には、引数用(上り)と戻り値用(下り)があります。
- 各種コンバータ
  1つのJavaScriptオブジェクト/プリミティブを、1つのJavaのオブジェクト/プリミティブに変換するコンポーネントです。Ajaxフレームワークでは、多数のコンバータが用意されており、コンバータごとに変換の方式が異なります。コンバータの詳細は、「3.6.2 コンバータ一覧」を参照してください。
- Java型情報
  用意されている各種コンバータから、1つのコンバータを選ぶ際に利用されます。Java型情報の詳細は、「3.6.3 コンバータ選択のロジック」を参照してください。
- コンバータ定義ファイル
  コンバータ定義ファイルの記述内容によって、コンバータをカスタマイズすることができます。コンバータ定義ファイルの詳細は、「A.6 コンバータ設定の定義(conversion)」を参照してください。

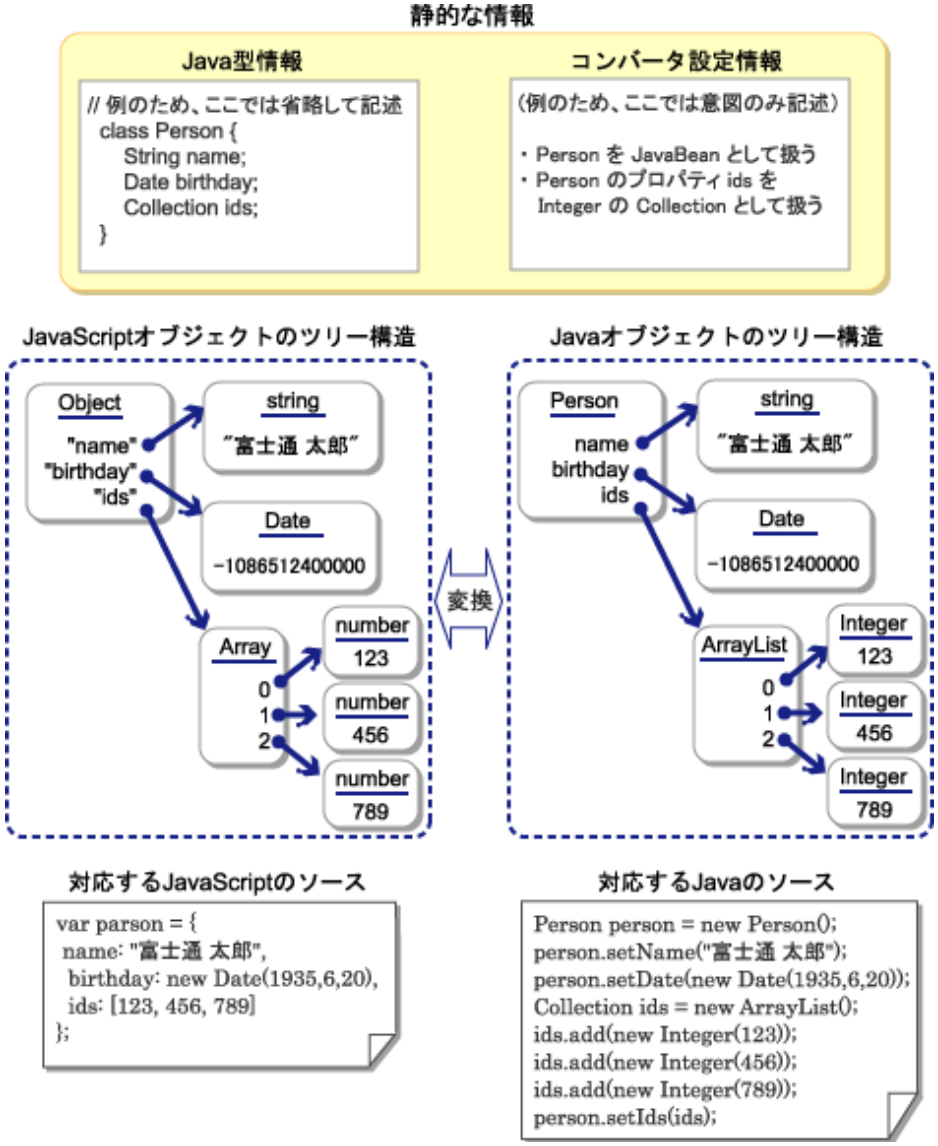

注意

データ型変換機能では、JavaScriptとJavaの間の自然なマッピングの範囲のデータ型変換に限定しています。より複雑な変換は、アプリケーションのJavaScript側またはJava側で行ってください。

データ型変換の例

以下の図に、データ型変換の例を示します。

図3.8 データ型変換の例

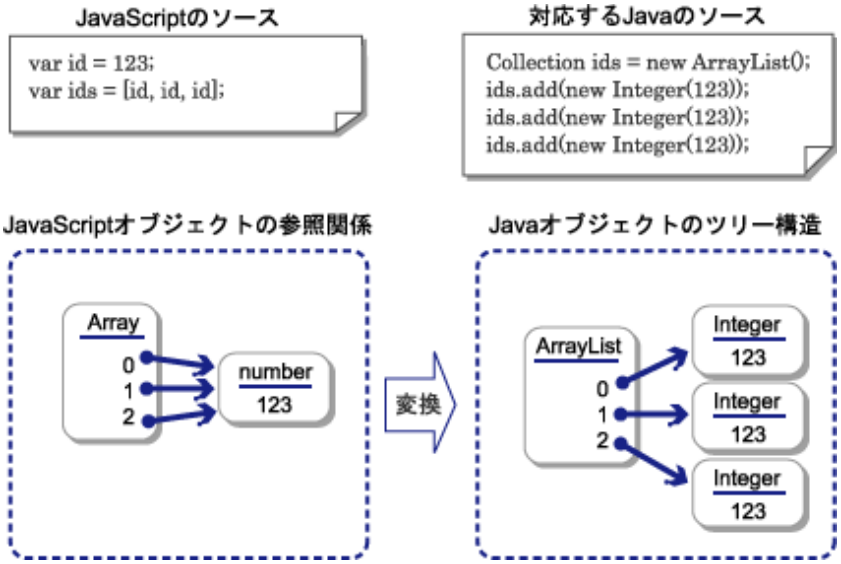


データ型変換の範囲

以下に、データ型変換機能がサポートするデータ型変換の範囲を説明します。

- 変換対象となるJavaScriptとJavaのオブジェクト(プリミティブ値を含む)は、一対一に変換されます。
- 変換対象オブジェクトの参照関係の同一性は保存されません。すなわち、あるオブジェクトが複数のオブジェクトから参照されている場合、変換後は、オブジェクトはその参照ごとに生成されます。詳細は、「図3.9 オブジェクトの複数回参照の例」を参照してください。
- オブジェクトのツリー構造の階層は、上限が20です。それを超える場合はエラーとなります。循環参照がある場合も同一のエラーとなります。

図3.9 オブジェクトの複数回参照の例



データ型変換機能では、上記の変換の範囲で、以下のようなカスタマイズが可能です。

- コンバータの選択は、基本的に、Java側の型情報に基づいて行われます。また、ユーザー(開発者)がカスタマイズすることもできます。
- デフォルトの設定により、ユーザー(開発者)は明示的にコンバータを選ばなくても、データ型変換機能を利用できます。
- コンバータの選択方法や、コンバータの細かい動作は、定義ファイルに記述することにより変更できます。

データ型変換の例外

データ型変換機能が通知する例外については、「J.3.6 データ型変換機能に関するメッセージ」を参照してください。

3.6.2 コンバーター一覧

コンバータは、変換対象のデータ型に基づいて、単体型用と複合型用とに分類されます。

- 単体型データ
  プリミティブ型やそのラッパーオブジェクトのように、1つのデータで完結するものを指します。
- 複合型データ
  配列、JavaBeans、コレクション、マップなどのように、ほかのいくつかのデータを参照するものを指します。

単体型データ用のコンバータ

以下の表に、単体型データ用のコンバータの一覧を示します。

コンバータ名	Java型	JavaScript型	備考
BooleanConverter	boolean, Boolean	boolean, Boolean	
ByteConverter	byte, Byte	number, Number	範囲外はエラー
ShortConverter	short, Short	number, Number	範囲外はエラー
IntegerConverter	int, Integer	number, Number	範囲外はエラー
LongConverter	long, Long	number, Number	範囲外はエラー
FloatConverter	float, Float	number, Number	範囲外はエラー 値がNaNまたはInfinityの場合、下りではString("NaN"、"Infinity"、"-Infinity")に変換されます。

コンバータ名	Java型	JavaScript型	備考
DoubleConverter	double, Double	number, Number	範囲外はエラー 値がNaNまたはInfinityの場合、下りではString("NaN"、"Infinity"、"-Infinity")に変換されます。
CharacterConverter	char, Character	string, String	文字列が1文字以外のときはエラー
StringConverter	String	string, String	
DateConverter	java.util.Dateのサブクラス	Date	Servlet連携の場合は、1970年1月1日午前0時からのミリ秒を表すNumber型になります。
BigIntegerConverter	java.math.BigInteger	string, String	整数の書式でない場合はエラー
BigDecimalConverter	java.math.BigDecimal	string, String	小数点の書式でない場合はエラー

複合型データ用のコンバータ

以下の表に、複合型データ用のコンバータの一覧を示します。

コンバータ名	Java型	JavaScript型	備考
ArrayConverter	Javaの任意の配列	Array	
CollectionConverter	java.util.Collectionのサブクラス	Array	上りでは、elementの型はすべて、コンバータ定義ファイルのchild要素のtype要素で指定された型に変換されます。(注) なお、child要素のtype要素に対応するコンバータが定義されていない場合は、例外が通知されます。
MapConverter	java.util.Mapのサブクラス	Object	上りでは、keyの型はすべてStringに変換されます。valueの型はすべて、コンバータ定義ファイルのchild要素のtype要素で指定された型に変換されます。(注) なお、child要素のtype要素に対応するコンバータが定義されていない場合は、例外が通知されます。
BeanConverter	ユーザー指定のJavaBeanクラス	Object	JavaScriptのObjectのプロパティと、JavaBeanのプロパティを対応させます。 両者のプロパティ全体が一致しない場合、下りでは、Javaのプロパティが自動的にJavaScriptのプロパティに変換されます。 上りでは、JavaScript側のプロパティが少ない場合、JavaScript側のプロパティに対応するデータBeanのプロパティだけが設定されます。 JavaScript側にないプロパティについては、データBeanのプロパティは変更されません。
ExceptionConverter	java.lang.Throwableのサブクラス	Object	下りメッセージ作成時に、内部的に利用されます。ユーザーが意識する必要はありません。 内容は以下のとおりです。

コンバータ名	Java型	JavaScript型	備考
			{ "error_code": エラーコード, "name": クラス名, "message": getMessage()の結果, "cause": getCause()の結果 }

注) コンバータ定義ファイルの詳細は、「A.6 コンバータ設定の定義(conversion)」を参照してください。

3.6.3 コンバータ選択のロジック

コンバータの選択は、実行時に、以下のロジック(アルゴリズム)によって決まります。

a. ユーザーが、親オブジェクトのコンバータによって子オブジェクトのコンバータを指定した場合、それに従います。

b. a.以外で、ユーザーが、Javaの型(上りは入れ物の型、下りは入っているオブジェクトの型)に対してコンバータを指定した場合、それに従います。

c. a.およびb.に当てはまらない場合は、Javaの型に対するデフォルトのコンバータを使用します。詳細は、「デフォルトのルール」を参照してください。

デフォルトのルール

以下の表に、Java型に対応するデフォルトのコンバータの一覧を示します。

表3.1 Java型に対応するデフォルトコンバーター覧

Java型	コンバータ名
boolean, Boolean	BooleanConverter
byte, Byte	ByteConverter
short, Short	ShortConverter
int, Integer	IntegerConverter
long, Long	LongConverter
float, Float	FloatConverter
double, Double	DoubleConverter
char, Character	CharacterConverter
String	StringConverter
java.util.Date	DateConverter
java.math.BigInteger	BigIntegerConverter
java.math.BigDecimal	BigDecimalConverter
任意の配列	ArrayConverter
java.util.Collectionのサブクラス	CollectionConverter 詳細は、「Collection(Map)関連のデフォルトコンバータ」を参照してください。
java.util.Map のサブクラス	MapConverter 詳細は、「Collection(Map)関連のデフォルトコンバータ」を参照してください。
java.lang.Throwableのサブクラス	ExceptionConverter
上記に当てはまらないObject	例外

Collection(Map)関連のデフォルトコンバータ

Collection(Map)関連のデフォルトコンバータは、Java型に応じたconcreteTypeを設定したCollectionConverter(MapConverter)です。以下の表に、Collection(Map)関連のデフォルトコンバータの一覧を示します。

表3.2 Collection(Map)関連のデフォルトコンバーター覧

Java型	concreteType	コンバータ名
Collection	ArrayList	CollectionConverter
List	ArrayList	CollectionConverter
Map	LinkedHashMap	MapConverter
Set	LinkedHashSet	CollectionConverter
SortedMap	TreeMap	MapConverter
SortedSet	TreeSet	CollectionConverter
AbstractCollection	ArrayList	CollectionConverter
AbstractList	ArrayList	CollectionConverter
AbstractMap	LinkedHashMap	MapConverter
AbstractSequentialList	LinkedList	CollectionConverter
AbstractSet	LinkedHashSet	CollectionConverter
ArrayList	ArrayList	CollectionConverter
HashMap	HashMap	MapConverter
HashSet	HashSet	CollectionConverter
Hashtable	Hashtable	MapConverter
IdentityHashMap	IdentityHashMap	MapConverter
LinkedHashMap	LinkedHashMap	MapConverter
LinkedHashSet	LinkedHashSet	CollectionConverter
LinkedList	LinkedList	CollectionConverter
Properties	Properties	MapConverter
Stack	Stack	CollectionConverter
TreeMap	TreeMap	MapConverter
TreeSet	TreeSet	CollectionConverter
Vector	Vector	CollectionConverter
WeakHashMap	WeakHashMap	MapConverter

3.7 イベントログ機能

イベントログ機能は、通信フレームワーククライアントのJavaScriptアプリケーションから、簡単な記述でサーバ側のログを出力する機能です。

3.7.1 イベントログAPI

イベントログAPIには、以下の関数があります。

- AcfEventLog.trace([string メッセージ]);
- AcfEventLog.info([string メッセージ]);
- AcfEventLog.warn([string メッセージ]);
- AcfEventLog.error([string メッセージ]);

注意

string型以外を受け取った場合は、何も出力されません。

3.7.2 ログ出力レベルの設定

ログのレベルは、ログの詳細度を表します。特定のレベル以下のログだけを出力するように設定して、不要なログを抑止することができます。

クライアント側

クライアント側で以下のように設定すると、設定されたログレベル以下のログ情報がサーバに送られます。

```
RCF_config = {
  eventLogLevel: "[ログレベル]"
}
```

- ログレベル
  出力するログのレベルを、以下の文字列で設定します。

    TRACE | INFO | WARN | ERROR

  レベルは、左が高く、右が低くなります。
  例えば、「WARN」を指定した場合、WARNとERRORのログが出力されます。

  サーバ側の設定値(logLevel)との対応関係は、以下の表のとおりです。

クライアント設定文字列	サーバ設定数値
TRACE	20
INFO	10
WARN	7
ERROR	3

注意

eventLogLevelプロパティを宣言しない場合、または、eventLogLevelプロパティにnullを指定した場合、ログ出力の関数呼出しがあっても、ログはサーバに送られません。

サーバ側

イベントログの出力先や出力レベルは、Apcoordinatorのログ定義ファイルで制御します。
ログ定義ファイルは、WEB-INFフォルダ直下に配置します。デフォルトのファイル名は、logConf.xmlです。また、イベントログで出力するログに対するApcoordinatorログライブラリの管理名は、「acf_client」です。

以下に、イベントログを出力する場合のログ定義ファイルの記述例を示します。
以下の記述例では、ログレベルがWARNおよびERRORのログが出力されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<logConfig
    xmlns=http://interstage.fujitsu.com/schemas/uji/logConf
    xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xsi:schemaLocation="http://interstage.fujitsu.com/schemas/uji/logConf
                        http://interstage.fujitsu.com/schemas/uji/logconf.xsd">

  <config>
    <version>7.0</version>
  </config>
```

```
  <logComposer name="acf_client">
    <level>9</level>
    <output name="clientOut" type="stdout"/>
  </logComposer>

</logConfig>
```

ログ定義ファイルについては、Interstage Application Server の「Apcoordinator ユーザーズガイド」を参照してください。

ポイント

イベントログAPIを利用したサンプル(eventLog)については、「付録D サンプルアプリケーション」を参照してください。

イベントログの例外

イベントログ機能が通知する例外については、「J.3.4 イベントログ機能に関するメッセージ」を参照してください。

3.7.3 サーバ出力情報

イベントログ機能では、クライアントアプリケーションで指定したメッセージなどの情報に加え、クライアントJavaScriptエンジンやServletコンテナで取得可能な情報も、ログ情報として出力します。

以下の表に、サーバに出力されるログ情報を示します。

表3.3 サーバに出力されるログ情報

出力情報	情報の出所
サーバ側での出力時間	Servletで取得
ログレベル	ユーザー指定
メッセージ	ユーザー指定
クライアントでのログ出力要求時間	クライアントで取得
クライアントURL	クライアントで取得
クライアントwindow名	クライアントで取得
クライアントIPアドレス	Servletで取得
User-Agent	Servletで取得

3.8 セキュリティ機能

ここでは、通信フレームワークが利用するセキュリティ機能について説明します。

3.8.1 SSL

メソッド呼出しなどのブラウザからの非同期通信で、セキュアな通信を行う場合は、SSLを利用します。

ページを読み込んだプロトコルと非同期通信で行うプロトコルは、同じである必要があります。このため、非同期通信をHTTPSで行う場合は、ページ自体の読込みもHTTPSで行います。

以下の表に、ページを読み込んだプロトコルと非同期通信のプロトコルの関係を示します。

表3.4 ページを読み込んだプロトコルと非同期通信のプロトコルの関係

		ページを読み込んだプロトコル	
		HTTP	HTTPS
非同期通信のプロトコル	HTTP	○	×
	HTTPS	×	○

○: 使用可、×: 使用不可

3.8.2 ユーザー認証

ユーザー認証は、ユーザーIDとパスワードによって、正当なユーザーであるかをチェックする機能です。これにより、不当なユーザーからのアクセスを防止することができます。

通信フレームワークでは、web.xmlに記述することにより、ユーザー認証を行います。
web.xmlの設定方法の詳細は、ご利用のアプリケーションサーバのマニュアルを参照してください。

フォームベース認証を行う場合、Ajaxを利用したページを表示する前に認証を行っておく必要があります。認証を行っていない状態で、非同期通信によるサーバ呼出しはできません。

3.8.3 アクセス制限

通信フレームワークでは、以下の単位でアクセス制限を設定することができます。

- セキュリティロール
- 転送方法

セキュリティロールによるアクセス制限

セキュリティロールを利用して、エントリサーブレットへのアクセスを制限することができます。

通信フレームワークでは、web.xmlに記述することにより、エントリサーブレットをadminロールだけにアクセス制限します。
web.xmlの設定方法の詳細は、ご利用のアプリケーションサーバのマニュアルを参照してください。

転送方法

クライアントとエントリサーブレットの間の転送方法(通信方法)に対してアクセスを制限することができます。

転送方法に対するアクセス制限の詳細は、ご利用のアプリケーションサーバのマニュアルを参照してください。

なお、ページを読み込んだプロトコルと非同期通信で行うプロトコルは、同じである必要があります。詳細は、「3.8.1 SSL」を参照してください。

3.9 リクエスト送信時のURLについて

Ajaxフレームワークを利用した通信、および画面部品内での通信において、リクエスト送信時のURLに任意のクエリ文字列やURLリライティングで使用するセションIDを付加することができます。

ここでは、リクエスト送信時のURLにクエリ文字列やセションIDを付加する方法について説明します。

3.9.1 クエリ文字列の追加

Ajaxフレームワークを利用した通信、および画面部品内での通信において、リクエスト送信時のURLに任意のクエリ文字列を付加することができます。
ロードバランサなどの中継機器やサーバ側の処理において、URLにクエリ文字列を付加する必要がある場合は、本機能を使用してください。

クエリ文字列を付加するには、動作定義によって付加する方法と、JavaScript APIによって付加する方法があります。

ここでは、以下の項目について説明します。

- 動作定義によってクエリ文字列を付加する方法
- JavaScript APIによってクエリ文字列を付加する方法
- クエリ文字列を利用する項目

- JavaScript APIによって任意のURLにクエリ文字列を付加する方法
- サーバ側でクエリ文字列を利用する方法

動作定義によってクエリ文字列を付加する方法

URLに付加するクエリ文字列は、Ajaxフレームワークの動作定義に追加することによって、設定することができます。

以下に、key1=value1&key2を送信する場合の設定例を示します。

```
RCF_config = {
  // key1=value1&key2を送信する場合
  queryString: {key1:'value1', key2:''}
};
```

クエリ文字列がキーだけで値を持たない場合は、上記の例(key2:'')のように、空文字を指定します。
RCF_configオブジェクトのqueryStringプロパティについては、「2.3.2 Ajaxフレームワークの動作定義」を参照してください。

以下に、クエリ文字列指定時の注意事項について説明します。

- URLに設定するホスト名の文字列長によっては、クエリ文字列に設定可能な文字列長の考慮が必要です。URLとして有効な文字列長は、最大で2083バイトです。
- キーと値を区切るための「=」、クエリ文字列を複数指定した場合のキーとキーを区切るための「&」は、設定文字数として計上されます。
- URLエンコードの対象となる文字列が設定された場合、URLエンコード処理後の文字列数が計上されます。例えば、文字コード「UTF-8」の実行環境において、文字列「あ」が設定された場合は、9バイト(%E3%81%82)として処理されます。
- URLの文字列長が2083バイトを超える文字列が設定された場合は、エラーメッセージ(RCF20005)が出力されます。
- クエリ文字列に設定された値に異常がある場合は、エラーメッセージ(RCF20006)が出力されます。

JavaScript APIによってクエリ文字列を付加する方法

JavaScript APIによってクエリ文字列を付加する場合は、以下のAPIを利用します。

JavaScript API	説明
RCF.setQueryString(Object queryString)	パラメーターには、URLに付加するクエリ文字列を連想配列で指定します。設定済みのクエリ文字列は、パラメーターで指定したクエリ文字列に置き換わります。 クエリ文字列指定時の注意事項については、「動作定義によってクエリ文字列を付加する方法」を参照してください。 パラメーターの設定値に異常がある場合は、エラーメッセージ(RCF20009)が出力されます。
RCF.addQueryString(String key, String value)	パラメーターには、URLに付加するクエリ文字列のキー名と値を指定します。設定済みのクエリ文字列に、パラメーターで指定したキーが追加設定されます。指定したキー名がすでに設定されている場合、古い値は指定した値に置き換わります。 クエリ文字列指定時の注意事項については、「動作定義によってクエリ文字列を付加する方法」を参照してください。 keyパラメーターには、nullを指定することはできません。valueパラメーターには、nullを指定することができます。valueパラメーターにnullを指定した場合、keyだけがクエリ文字列として作成されます。 パラメーターの設定値に異常がある場合は、エラーメッセージ(RCF20007)が出力されます。

クエリ文字列を利用する項目

動作定義やJavaScript APIによって設定されたクエリ文字列は、以下の項目でURLに付加されます。

- UI部品
  - ー ComboBox部品
    - buttonImageプロパティ
  - ー FragmentContainer部品
    - srcプロパティ
    - statusIconImageプロパティ
  - ー Window部品、PopupCalendar部品
    - closeButtonImageプロパティ
  - ー ViewColumnImage部品
    - モデルに指定した配列データに含まれる表示画像のURL
  - ー TreeView部品、ContextMenu部品
    - headImageプロパティ
- 通信フレームワーク
  - ー Apcoordinator連携機能、サーブレット連携機能
    - optionオブジェクトに設定するurlプロパティ
- マッシュアップフレームワーク
  - ー クライアント通信API
    - optionオブジェクトに設定するurlプロパティ
- その他
  - ー Ajaxフレームワークが内部処理で利用するURL指定
    - イベントログ機能
    - UI部品を初期化するためのJavaScriptファイルの取得、など

注意

以下に示すHTMLタグによって記述されたリソースの取得では、Ajaxフレームワークで設定されたクエリ文字列は付加されません。
アプリケーションで付加する必要があります。

- <script>タグ
  Ajaxフレームワークの動作定義、Ajaxフレームワークの初期化処理、ユーザー定義のJavaScriptファイルの取得も対象です。
- <img>タグ
  Button部品の子要素として記述した<img>タグも対象です。
- <a>タグ
- <link>タグ

JavaScript APIによって任意のURLにクエリ文字列を付加する方法

以下のAPIを利用すると、任意のURLに対して、動作定義やJavaScript APIによって設定されたクエリ文字列を付加することができます。

JavaScript API	説明
RCF.createQueryString()	RCF_configオブジェクトに設定されたクエリ文字列が、URLに付加可能な形式に変換されて返されます。

JavaScript API	説明
	返却されるクエリ文字列は、以下のような形式になります。 ?SVID01&key01=val01 クエリ文字列指定時の注意事項については、「動作定義によってクエリ文字列を付加する方法」を参照してください。

サーバ側でクエリ文字列を利用する方法

Apcoordinator連携機能、サーブレット連携機能でクライアントから送信されたクエリ文字列を取得する場合は、サーブレットのAPIを利用します。
以下に、ビジネスクラスからサーブレットのAPIを利用する方法を示します。

Apcoordinator連携機能を利用する場合

```
import javax.servlet.http.HttpServletRequest;
import com.fujitsu.uji.http.HttpDispatchContext;

  (省略)
public void login(DispatchContext context, mypkg.MyDataBean dataBean) {
  ...
  HttpServletRequest request = ((HttpDispatchContext)context).getServletRequest();
  // クエリ文字列全体を取得
  String query = request.getQueryString();
  // paramというキーに対する値を取得
  String value = request.getParameter("param");
  ...

}
```

サーブレット連携機能を利用する場合

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

  (省略)
public void doPost (HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
  ...
  // クエリ文字列全体を取得
  String query = request.getQueryString();
  // paramというキーに対する値を取得
  String value = request.getParameter("param");
  ...

}
```

3.9.2 URLリライティング

Ajaxフレームワークを利用した通信、および画面部品内での通信において、URLリライティングを使用する場合は、動作定義にセションIDを設定します。

ここでは、以下の項目について説明します。

- セションIDを付加する方法
- ApcoordinatorのURLリライティング機能を使用する方法
- セションIDを利用する項目
- JavaScript APIによって任意のURLにセションIDを付加する方法

注意

URLリライティングは、Cookieによるセション管理よりも、セションIDが搾取される危険性が高くなります。できる限り、Cookieによるセション管理を行ってください。URLリライティングを使用しない場合は、Cookieによるセション管理となります。

セションIDを付加する方法

URLに付加するセションIDは、Ajaxフレームワークの動作定義に追加することによって、設定することができます。

以下に、Ajaxフレームワークの動作定義に設定する例を示します。

```
<script type="text/javascript" charset="UTF-8" src="rcf_config.js"></script>

<script type="text/javascript">
//<![CDATA[
  // Ajaxフレームワークの動作定義にセションIDを設定
  RCF_config.jsessionid = "<%=session.getId()%>";
//]]>
</script>

<script type="text/javascript" src="acf/file/rcf/rcf.js"></script>
```

RCF_configオブジェクトのjsessionidプロパティについては、「2.3.2 Ajaxフレームワークの動作定義」を参照してください。

注意

セションIDの設定は、Ajaxフレームワークの初期化処理(rcf.js)よりも先に記述しなければなりません。

ApcoordinatorのURLリライティング機能を使用する方法

Apcoordinatorを利用する場合は、ApcoordinatorのURLリライティング機能も使用する必要があります。
ApcoordinatorのURLリライティング機能を使用するには、com.fujitsu.uji.http.HttpControlStateProfileクラスを継承して作成したセションクラスで、getSessionMode()メソッドをオーバーライドして、HttpControlStateProfile.URLREWRITINGを返却するようにします。詳細は、Interstage Application Serverの「Apcoordinator ユーザーズガイド」を参照してください。
以下に記述例を示します。

```
import com.fujitsu.uji.http.HttpControlStateProfile;

public class Session extends HttpControlStateProfile {
  public Session() {
  }

  public int getSessionMode() {
    return HttpControlStateProfile.URLREWRITING;
  }

}
```

セションIDを利用する項目

動作定義に設定されたセションIDは、以下の項目でURLに付加されます。

- UI部品
  - ComboBox部品
    - buttonImageプロパティ
  - FragmentContainer部品
    - srcプロパティ

- statusIconImageプロパティ

ー Window部品、PopupCalendar部品

- closeButtonImageプロパティ

ー ViewColumnImage部品

- モデルに指定した配列データに含まれる表示画像のURL

ー TreeView部品、ContextMenu部品

- headImageプロパティ

- 通信フレームワーク

ー Apcoordinator連携機能、サーブレット連携機能

- optionオブジェクトに設定するurlプロパティ

- マッシュアップフレームワーク

ー クライアント通信API

- optionオブジェクトに設定するurlプロパティ

- その他

ー Ajaxフレームワークが内部処理で利用するURL指定

- イベントログ機能

- UI部品を初期化するためのJavaScriptファイルの取得、など

注意

以下に示すHTMLタグによって記述されたリソースの取得では、Ajaxフレームワークで設定されたセションIDは付加されません。
アプリケーションで付加する必要があります。

- <script>タグ
  Ajaxフレームワークの動作定義、Ajaxフレームワークの初期化処理、ユーザー定義のJavaScriptファイルの取得も対象です。
- <img>タグ
  Button部品の子要素として記述した<img>タグも対象です。
- <a>タグ
- <link>タグ

JavaScript APIによって任意のURLにセションIDを付加する方法

以下のAPIを利用すると、任意のURLに対して、動作定義に設定されたセションIDを付加することができます。

JavaScript API	説明
RCF.encodeURL(String url)	パラメーターで指定したURLに、RCF_configオブジェクトのjsessionidで指定されたセションIDが付加されて返されます。 RCF_configオブジェクトのjsessionidが設定されていない場合は、パラメーターで指定したURLがそのまま返されます。 返却されるURLは、以下のような形式となります。 url;jsessionid=123ABC パラメーターの設定値に異常がある場合は、エラーメッセージ(RCF20008)が出力されます。

3.10 通信フレームワーク内のエラー

通信フレームワークでエラーが発生した場合、エラーオブジェクトが作られます。

以下に、エラーオブジェクトの構造を示します。
なお、以下に記載されている以外のプロパティは、アプリケーションで使用しないでください。

```
{
  errorCode: [エラーコード],
  message: [メッセージ],              // サーバ例外の場合 Exception#getMessage()の値
  cause: [内包されるエラーオブジェクト]  // サーバ例外の場合 Exception#getCause()の値
}
```

- errorCodeには、エラーメッセージ番号の文字列から先頭の文字列「RCF」が取り除かれた値が、数値で通知されます。
- causeには、エラーオブジェクトが内包されています。
- causeで内包されるエラーオブジェクトの階層は、データ型変換機能でサポートできる、オブジェクトのツリー構造の階層と同様に規定されます。詳細は、「データ型変換の範囲」を参照してください。
- 各プロパティ値には、以下のような注意点があります。
  - 内包されるエラーオブジェクトを除いて、errorCodeは必ず設定されています。
  - 内包されるエラーオブジェクトは、errorCodeがnullの場合があります。
  - サーバ例外のException#getMessage()がnullまたは空文字列の場合、messageは""です。
  - 内包されるエラーオブジェクトが存在しない場合、causeはnullです。
- RCF0700のエラーが発生した場合は、サーバからのレスポンスデータがそのままcauseに格納されます。messageの内容は、「J.3.7 通信に関するメッセージ」の「RCF0700」を参照してください。

なお、通信フレームワークが通知する例外については、「J.3 サーバのエラーメッセージ」を参照してください。

3.11 通信フレームワーク定義ファイル

通信フレームワークでは、以下の3種類の定義ファイルがあります。

- エントリサーブレットのweb.xml
  ファイル: WEB-INF/web.xml
  詳細は、「5.3.2 web.xmlの設定」を参照してください。
- Ajaxフレームワーク環境定義ファイル
  ファイル: WEB-INF/acf.xml
  詳細は、「付録A 環境定義ファイル」を参照してください。
- ログ出力の定義ファイル(Apcoordinatorのログ定義ファイル)
  ファイル: WEB-INF/logConf.xml(デフォルト)
  通信フレームワークのログ出力は、Apcoordinatorのログ出力ライブラリを利用しています。定義ファイルの詳細は、Interstage Application Server の「Apcoordinator ユーザーズガイド」を参照してください。

  以下の表に、通信フレームワークが出力するログの種類と管理名を示します。

出力するログの種類	Apcoordinatorログライブラリの管理名
通信フレームワークサーバ自身のログ	acf_server
イベントログ機能のログ	acf_client

第4章 マッシュアップフレームワーク

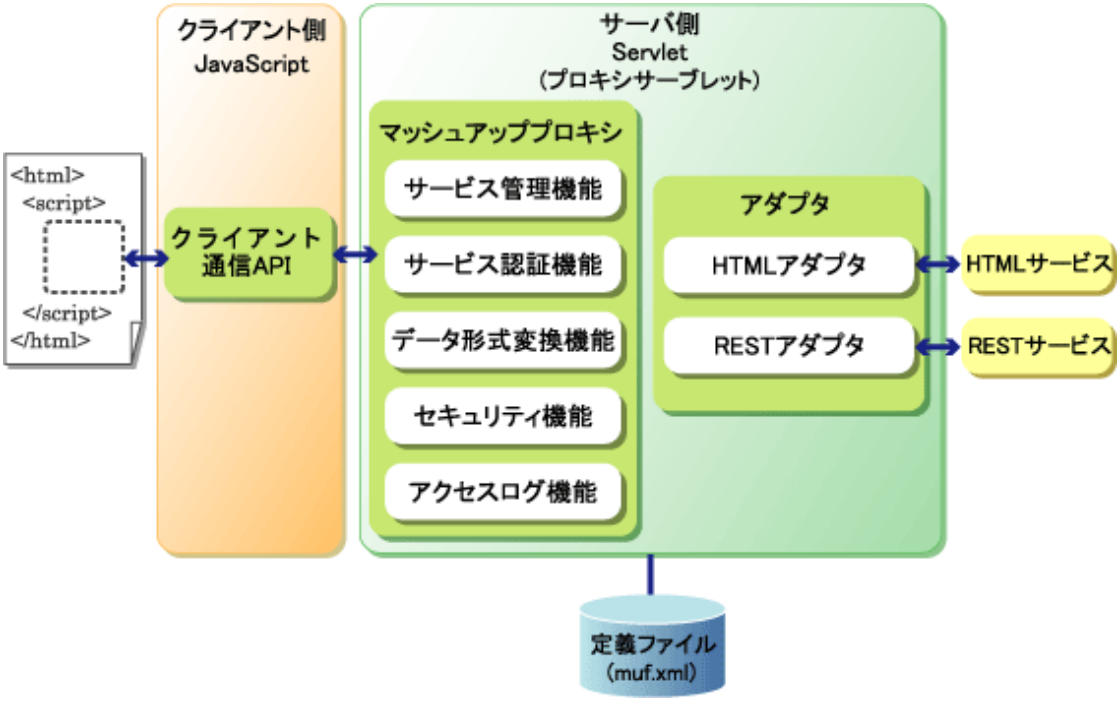
本章では、マッシュアップフレームワークの機能および使用方法について説明します。

4.1 マッシュアップフレームワークの構成

マッシュアップフレームワークは、WebブラウザとWebサービスの通信を中継するフレームワークを提供します。Ajaxフレームワークアプリケーションに対して、様々な社外/社内のサービスを取り込んで、それらを組み合わせたWebアプリケーションを作成することができます。

以下の図に、マッシュアップフレームワークの構成を示します。

図4.1 マッシュアップフレームワークの構成



マッシュアップフレームワークで提供する機能のうち、クライアント通信APIは、クライアント側で動作します。これ以外の機能は、サーバ側で動作します。

サーバ側では、プロキシサーブレットとして動作します。プロキシサーブレットの詳細は、「5.4 プロキシサーブレット」を参照してください。
クライアント側では、JavaScriptとして動作します。このJavaScript APIは、例外を通知することがあります。通知される例外オブジェクトの構造は、「4.8 マッシュアップフレームワーク内のエラー」を参照してください。

クライアント通信API

クライアント通信APIは、ブラウザ上のJavaScriptアプリケーションから、Webサービスを呼び出す機能です。これにより、インターネット上にあるWebサービスや企業内の既存システムを活用することができます。
クライアント通信APIの詳細は、「4.2 クライアント通信API」を参照してください。

マッシュアッププロキシ

マッシュアッププロキシは、クライアント通信APIからの呼出しに応じて、様々なWebサービスから情報を取得します。取得した情報は、Ajaxフレームワーク上で使用できるデータ形式(XML/JSON形式)に変換してからクライアントに返却します。
マッシュアッププロキシでは、Webサービスを呼び出すために以下の機能を提供します。

- **サービス管理機能**
  サービス管理機能は、ブラウザ上のJavaScriptアプリケーションからアクセスするWebサービスを管理する機能です。
  サービス管理機能の詳細は、「4.3 サービス管理機能」を参照してください。

- **サービス認証機能**
  サービス認証機能は、Webサービスにアクセスする際に、認証情報を付加する機能です。
  サービス認証機能の詳細は、「4.4 サービス認証機能」を参照してください。

- **データ形式変換機能**
  データ形式変換機能は、アダプタから受け取ったXML情報をクライアントのJavaScriptで使えるように、JSON形式に変換する機能です。
  データ形式変換機能の詳細は、「4.5 データ形式変換機能」を参照してください。

- **セキュリティ機能**
  セキュリティ機能は、リクエストの正当性を確認する機能です。リクエストのセション情報をチェックすることで、マッシュアッププロキシへのほかのアプリケーションからのアクセスを制限します。
  なお、SSL、ユーザー認証、アクセス制限については、通信フレームワークと同じセキュリティ機能を利用します。詳細は、「3.8 セキュリティ機能」を参照してください。

- **アクセスログ機能**
  アクセスログ機能は、ブラウザ上のJavaScriptアプリケーションからWebサービスへのアクセスログを収集し、出力する機能です。
  アクセスログ機能の詳細は、「4.6 アクセスログ機能」を参照してください。

アダプタ

アダプタは、様々なWebサービスとのアクセスを制御する部品の総称です。マッシュアップフレームワークでは、RESTサービスやHTMLサービスのためのアダプタを提供しています。
アダプタの詳細は、「4.7 アダプタ」を参照してください。

4.2 クライアント通信API

クライアント通信APIを利用すると、Webブラウザ上のJavaScriptからWebサービスを呼び出すことができます。

マッシュアップフレームワークでは、マッシュアッププロキシを経由して、Webサービスにアクセスします。クライアント通信APIは、ブラウザとマッシュアッププロキシ間の非同期通信を行うJavaScript API(MuRequest.send)を提供します。このAPIを使用することにより、クロスドメイン制約に影響されることなく、指定したWebサービスに任意のデータを送信し、結果を非同期で受信することができます。

以下に、MuRequest.send関数について説明します。

MuRequest.send関数の記述形式

以下に、MuRequest.send関数の記述形式を示します。

```
MuRequest.send (dataObj, requestParams, option);
```

- **dataObj**
  アダプタがWebサービスに送信するクエリ文字列を表すオブジェクトです。
  string、number、boolean型のプロパティを持ちます。
  クエリ文字列を送信しない場合は、nullを指定します。

  以下の形式のオブジェクトを指定します。この形式以外のオブジェクトは、クエリ文字列として送信されません。

```
{key1:<string>, key2:<string>}
```

- **requestParams**
  アクセスするサービスを一意に決定するためのID(以降、サービスIDと呼びます)や、マッシュアッププロキシの動作を制御するための値を格納したリクエストパラメーターオブジェクトです。
  リクエストパラメーターが不要な場合は、nullを指定します。

  以下に、リクエストパラメーターオブジェクトのプロパティを示します。

プロパティ名	概要	型	省略可否
serviceId	サービスID サービス管理機能で定義したサービスIDを指定します。	string	不可

プロパティ名	概要	型	省略可否
type	マッシュアッププロキシからのレスポンス形式 以下のどちらかの形式を指定します。 ・ xml XML形式の文字列 ・ json JavaScriptオブジェクト 大文字/小文字の区別はなく、省略値は、jsonです。	string	可

- **option**
MuRequestの動作を制御するための値を格納した通信設定オブジェクトです。
optionにnullが指定された場合、または省略された場合は、通信せずにエラーが発生します。

以下に、通信設定オブジェクトのプロパティを示します。

プロパティ名	概要	型	省略可否
url	通信先のURL 「muf/proxy」を指定します。サブフォルダに配置されているHTML/JSPファイルからマッシュアッププロキシを呼び出す場合、urlプロパティで指定する「muf/proxy」について、パスの考慮が必要です。アプリケーションフォルダを基点とした「muf/proxy」が参照できるように相対パスで指定してください。例えば、アプリケーションフォルダ直下のJSPフォルダにHTML/JSPファイルが存在する場合は、「../ muf/proxy」を指定してください。HTML/JSPファイルがインクルードされている場合には、インクルード元のHTML/JSPファイルが存在するフォルダを基点とした相対パスで指定してください。 URLには、任意のクエリ文字列やURLリライティングで使用するセションIDを付加することができます。詳細は、「3.9 リクエスト送信時のURLについて」を参照してください。	string	不可
callback	レスポンスハンドラ(正常終了時のコールバック) 第1引数にサーバから返却されたJavaScriptオブジェクトが渡されます。 リクエストパラメーターオブジェクトのtypeがxmlの場合はStringオブジェクト、typeがjsonの場合はマッシュアッププロキシ側で規定したJavaScriptオブジェクトです。	function	不可
errorHandler	エラーハンドラ(エラー発生時のコールバック) 第1引数にエラーオブジェクトが渡されます。 詳細は、「4.8 マッシュアップフレームワーク内のエラー」を参照してください。	function	可
timeout	タイムアウト時間(ミリ秒) サーバからのレスポンスが完了するまでの最大待ち時間です。この時間を超えると、エラーハンドラ呼出しとなります。 省略値は、120000(120秒)です。 なお、0を指定することはできません。	number	可
preInvoke	メソッド呼出し直前に呼ばれるコールバック 引数はありません。	function	可
postInvoke	メソッド呼出し完了直後に呼ばれるコールバック 引数はありません。	function	可

MuRequest.send関数の記述例

以下に、Webサービスにアクセスし、結果を表示する場合の、MuRequest.send関数の記述例を示します。

```
function clickExecuteButton(){
    dataObj={key1:'value1',key2:'value2'};
    var requestParams = {
        serviceId:'serviceA',//サービスIDを指定
        type:'json'//レスポンス形式を指定
    };
    var option = {
        url:'muf/proxy',
        callback:function(data){/* ここに処理を記述 */},
        errorHandler:function(err){/* ここに失敗時の処理を記述 */}
    };
    MuRequest.send(dataObj,requestParams,option);
}
```

MuRequest.send関数が通知する例外

MuRequest.send関数が通知する例外については、「J.2.2 通信フレームワーク(JavaScript)に関するメッセージ」を参照してください。

4.3 サービス管理機能

サービス管理機能では、クライアント通信APIからサービスIDを受け取り、そのサービスIDに基づいて、使用するアダプタの情報やアダプタに受け渡す情報をマッシュアップ定義ファイルから取得します。サービス管理機能で定義されていないサービスは、セキュリティ確保のため、アクセスできません。

サービスIDとアダプタの関連付けの設定方法については「5.7.11 マッシュアップ定義ファイルの編集」を、設定内容については「B.4 サービスの定義(muf_services)」を参照してください。

サービス管理機能が通知する例外

サービス管理機能が通知する例外については、「J.3.8 マッシュアップに関するメッセージ」を参照してください。

4.4 サービス認証機能

サービス認証機能は、認証が必要なWebサービスにアクセスする際に、認証情報を付加する機能です。

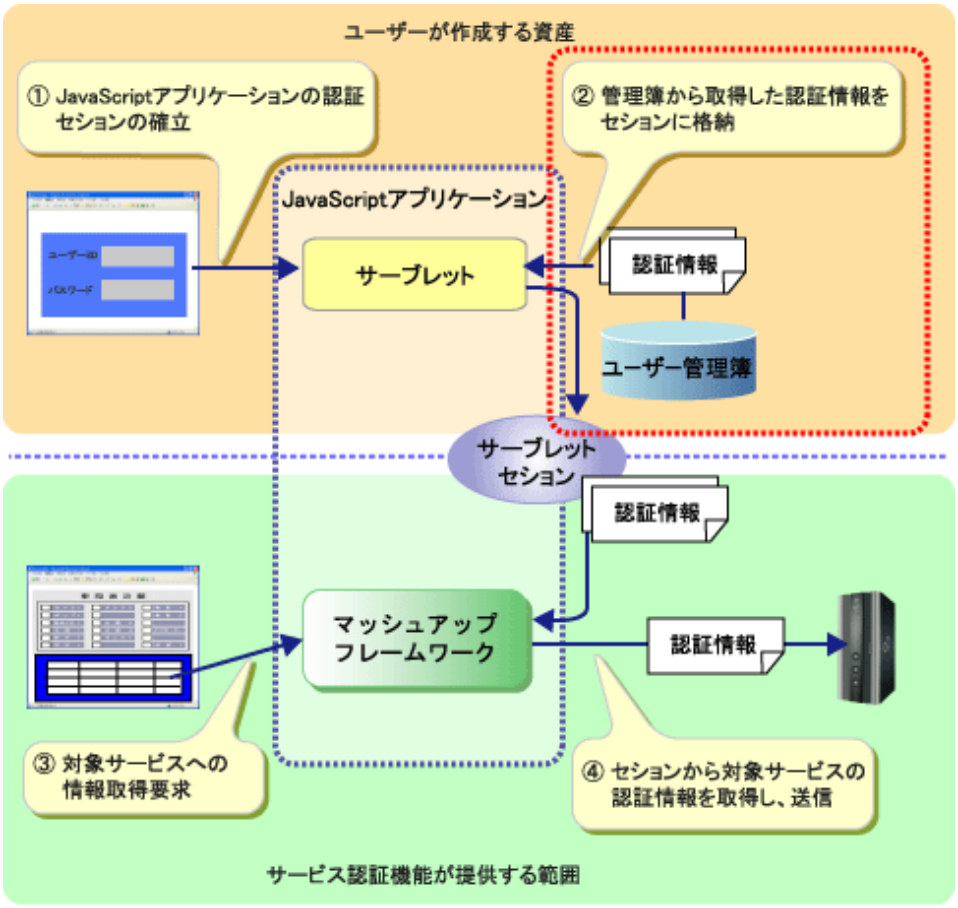
利用するHTMLサービスやRESTサービスの中には、アクセスする際に認証が必要なものがあります。サービス認証機能では、JavaScriptアプリケーションから渡された認証情報をサービスのアクセス時に付加することによって、サービスへの認証を実現します。
JavaScriptアプリケーションからマッシュアップフレームワークに認証情報を受け渡すには、セション方式を利用します。

4.4.1 セション方式

セション方式では、サーブレットセションを介して、JavaScriptアプリケーションからマッシュアップフレームワークに認証情報を受け渡します。

以下の図に、セション方式の動作概要を示します。

図4.2 セション方式の動作概要



サービス認証機能でユーザーが作成する資産は、以下のとおりです。

- 認証情報
  Webサービスごとに必要な認証情報をサーブレットセションに格納します。認証情報の内容については、「4.4.2 認証情報の内容」を参照してください。認証情報を格納する方法については、「4.4.3 認証情報の設定」を参照してください。
- ユーザー管理簿
  Webサービスへの認証情報の管理は、Ajaxフレームワークアプリケーションで行います。そのため、認証情報を格納する管理簿は、RDBやLDAPなど、任意のものを使用することができます。ユーザーIDと各Webサービスへの認証情報の対応付けを管理しておく必要があります。

4.4.2 認証情報の内容

サービス認証機能では、サービスの認証方式として、以下の2つの方式をサポートしています。

- 基本認証
  HTTPプロトコルで規定されたWebサーバでのユーザー認証方式です。
- Proxy認証
  HTTPプロトコルで規定されたProxyサーバでのユーザー認証方式です。

セションに格納する認証情報には、各認証に共通の情報と認証ごとに異なる情報があります。

共通の認証情報

以下に、各認証に共通の認証情報の内容を示します。

属性	意味	値	省略可否
globalID	アプリケーションを利用するエンドユーザを一意に識別するID 例) アプリケーションのユーザーID	ASCII文字	不可

属性	意味	値	省略可否
serviceID	サービス管理機能で定義したWebサービスのサービスID	ASCII文字	不可

基本認証固有の認証情報

以下に、基本認証固有の認証情報の内容を示します。

属性	意味	値	省略可否
userID	サービスに対するユーザーID セション上では暗号化することを推奨します。	ASCII文字	不可
Password	サービスに対するパスワード セション上では暗号化することを推奨します。省略された場合は空とみなします。	ASCII文字	可

Proxy認証固有の認証情報

以下に、Proxy認証固有の認証情報の内容を示します。

属性	意味	値	省略可否
userID	プロキシに対するユーザーID セション上では暗号化することを推奨します。	ASCII文字	不可
password	プロキシに対するパスワード セション上では暗号化することを推奨します。省略された場合は空とみなします。	ASCII文字	可

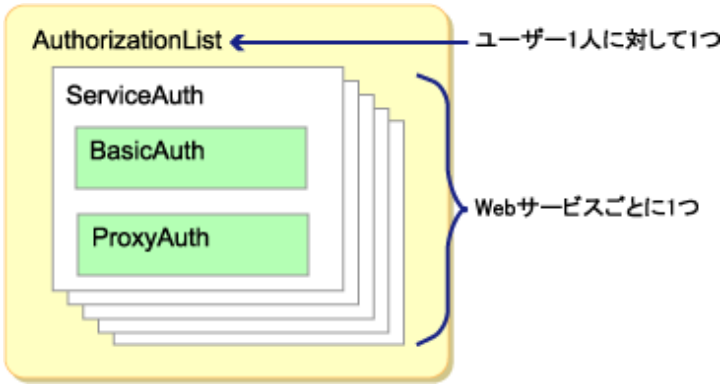
4.4.3 認証情報の設定

認証情報は、サービス認証機能のAPIを使用して、サーブレットセションに格納します。サービス認証機能のAPIの詳細は、「Ajaxフレームワーク(マッシュアップ) APIリファレンス」を参照してください。

認証情報の格納形式

認証情報は、AuthorizationListクラスに設定します。
AuthorizationListには、以下の形式で認証情報を設定します。

図4.3 認証情報の格納形式



サーブレットセションへの格納方法

サーブレットセションへは、HttpSession#setAttributeメソッドを利用して、以下のセションキーに格納します。

- セションキー

```
com.fujitsu.interstage.rcf.mashup.authinfo.service
```

セションキーは、AuthorizationクラスのSESSION_AUTHORIZATION_KEY定数に定義されています。

使用例

サービス認証機能のAPIを利用したプログラム例については、「5.10.9 ビジネスクラスの作成(マッシュアップ開発例)」の「loginメソッドの記述内容」を参照してください。

4.4.4 サービス認証機能のセキュリティ

ここでは、サービス認証機能が提供するセキュリティ機能について説明します。

認証情報の暗号化

セキュリティ強化のため、セションに格納する認証情報を暗号化します。したがって、セションから取り出した認証情報は、復号化してから使用する必要があります。復号化のインターフェースはマッシュアップフレームワークで隠蔽しており、セションに格納された認証情報は、マッシュアップフレームワークでだけ利用することができます。

なお、JavaVMのオプションに「-Dcom.fujitsu.interstage.rcf.mashup.auth.crypt=false」を設定した場合は、暗号化しません。

監査ログ

サービスアクセス時に認証を行ったかどうかを監査するため、監査ログを採取します。監査ログは、アクセス時の認証に関するものであるため、アクセスログ内に出力されます。認証を行った場合は、アクセスログに認証方式の情報が追加されます。

アクセスログについては、「4.6 アクセスログ機能」を参照してください。

4.5 データ形式変換機能

データ形式変換機能は、アダプタから受け取ったXML情報をクライアントのJavaScriptで使えるように、JSON形式に変換します。アダプタから返却されるデータがJSON形式の場合は、変換処理は行われません。

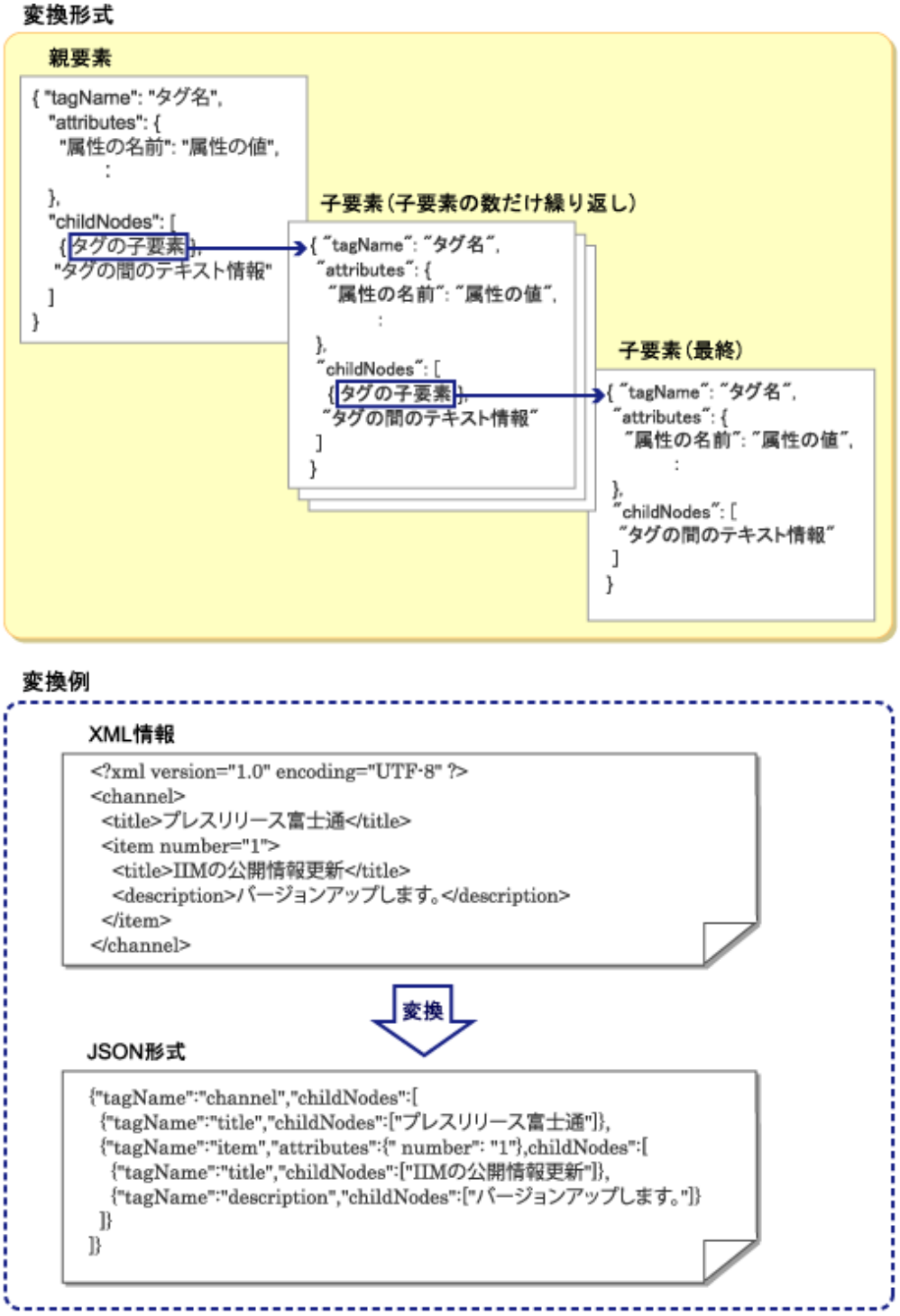
注意

クライアント通信APIでマッシュアッププロキシからのレスポンス形式(typeリクエストパラメーター)にxmlを指定した場合は、XML情報がそのまま出力されます。

データ形式変換の例

以下の図に、データ形式変換の例を示します。

図4.4 データ形式変換の例



データ形式変換機能が通知する例外

データ形式変換機能が通知する例外については、「J.3.8 マッシュアップに関するメッセージ」を参照してください。

4.6 アクセスログ機能

アクセスログ機能は、アクセスログを出力するための情報を収集し、ログファイルに出力します。

以下に、アクセスログの形式を示します。

```
[Date] [UserID] [接続元IPアドレス] [サービスID] [認証] [メッセージ]
```

以下に、出力される項目について説明します。

項目	説明
Date	ログの出力日時です。 「YYYY/MM/DD HH:MM:SS JST」の形式で出力されます。
UserID	サーブレットセションIDです。
接続元IPアドレス	接続元のIPアドレスです。
サービスID	アクセスしたサービスのIDです。
認証	認証情報を使用したかどうかを示します。 以下の文字列のどれかが設定されます。 ・ Null 使用していない ・ PROXY Proxy認証を使用 ・ BASIC 基本認証を使用 ・ PROXY,BASIC Proxy認証と基本認証を使用
メッセージ	アクセス先のURLやレスポンスコードなど、アダプタからの情報です。

アクセスログの出力先に関する情報は、マッシュアップ定義ファイルに設定します。設定内容については、「B.5.3 ログの定義(log)」を参照してください。

4.7 アダプタ

アダプタは、様々なWebサービスとのアクセスを制御する部品の総称です。マッシュアップフレームワークでは、RESTサービスやHTMLサービスのためのアダプタを提供しています。

アダプタには、以下の2種類があります。

- HTMLアダプタ
- RESTアダプタ

4.7.1 HTMLアダプタ

HTMLアダプタは、既存のWebアプリケーションからHTMLを取得します。取得したHTMLは、HTMLフィルタ機能を利用して正規化し、XML情報としてマッシュアッププロキシに返却します。また、スクレイピング機能を利用して、データを切り出すこともできます。

HTMLアダプタの機能

HTMLアダプタでは、以下の機能を提供します。

- **通信機能**
  通信機能は、各サービスとHTTP通信またはHTTPS通信を行います。タイムアウトや同時接続数の管理など、アプリケーション単位で一括して管理します。
- **HTMLフィルタ機能**
  HTMLフィルタ機能は、HTMLをXHTMLに変換する機能です。HTMLフィルタ機能を利用して、各サービスから取得したHTMLを正規化することができます。

  以下の表に、タグを正規化する例を示します。

変換前	変換後
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<img src="/pwimage/logon.gif" >	
閉じられていないtable	</table>を補完する

なお、スクリプトとして動作する以下のタグは、すべて削除されます。

- script
- style
- object
- embed
- link
- applet
- iframe
- frame
- frameset
- layer
- ilayer
- meta

また、タグに以下の属性を含む場合、属性だけが削除されます。

- onで始まる属性
- id
- class
- tabindex

- **スクレイピング機能**
  スクレイピング機能は、HTMLフィルタ機能で変換したXHTMLを、XSLTプロセッサを使用して、XMLとして部分切り出しする機能です。スクレイピング機能を利用して、各サービスから取得したデータのうち、実際に必要なデータだけを切り出すことができます。アダプタ管理では、スクレイピング機能をユーティリティ(スクレイピングツール)として提供します。スクレイピングツールの使用方法については、「付録G スクレイピングツール」を参照してください。

パラメーターの設定

HTMLアダプタには、以下のパラメーターが設定できます。

キー	値	省略	記述例
URL	HTMLを返却するWebサービスのURLを指定します。	×	http://jp.fujitsu.com
XSL	スクレイピングツールで作成したXSLファイル名を指定します。(注)	○	rss.xsl

○: 可能、×: 不可能
注) XSLファイルは[コンテキストルート]/WEB-INF/xslフォルダ配下に格納してください。

パラメーターは、マッシュアップ定義ファイルに設定します。設定方法については「5.7.11 マッシュアップ定義ファイルの編集」を、設定内容については「B.4 サービスの定義(muf_services)」を参照してください。

4.7.2 RESTアダプタ

RESTアダプタは、REST形式のWebアプリケーションからXML形式で情報を取得します。

RESTアダプタの機能

RESTアダプタでは、以下の機能を提供します。

- **通信機能**
  通信機能は、各サービスとHTTP通信またはHTTPS通信を行います。タイムアウトや同時接続数の管理など、アプリケーション単位で一括して管理します。

パラメーターの設定

RESTアダプタには、以下のパラメーターが設定できます。

キー	値	省略	記述例
URL	XMLを返却するWebサービスのURLを指定します。	×	http://restserver/listrest.cgi?product=A

×: 不可能

パラメーターは、マッシュアップ定義ファイルに設定します。設定方法については「5.7.11 マッシュアップ定義ファイルの編集」を、設定内容については「B.4 サービスの定義(muf_services)」を参照してください。

4.8 マッシュアップフレームワーク内のエラー

マッシュアップフレームワークでエラーが発生した場合、エラーオブジェクトが作られます。

以下に、エラーオブジェクトの構造を示します。
なお、以下に記載されている以外のプロパティは、アプリケーションで使用しないでください。

```
{
  errorCode: [エラーコード],
  message: [メッセージ],              // サーバ例外の場合 Exception#getMessage()の値
  cause: [内包されるエラーオブジェクト]  // サーバ例外の場合 Exception#getCause()の値
}
```

- errorCodeには、エラーメッセージ番号の文字列から先頭の文字列「RCF」が取り除かれた値が、数値で通知されます。
- causeには、エラーオブジェクトが内包されています。
- causeで内包されるエラーオブジェクトの階層は、データ型変換機能でサポートできる、オブジェクトのツリー構造の階層と同様に規定されます。詳細は、「データ型変換の範囲」を参照してください。
- 各プロパティ値には、以下のような注意点があります。
  - 内包されるエラーオブジェクトを除いて、errorCodeは必ず設定されています。
  - 内包されるエラーオブジェクトは、errorCodeがnullの場合があります。
  - サーバ例外のException#getMessage()がnullまたは空文字列の場合、messageは""です。
  - 内包されるエラーオブジェクトが存在しない場合、causeはnullです。
- RCF0700のエラーが発生した場合は、サーバからのレスポンスデータがそのままcauseに格納されます。messageの内容は、「J.3.7 通信に関するメッセージ」の「RCF0700」を参照してください。

なお、マッシュアップフレームワークが通知する例外については、「J.3.8 マッシュアップに関するメッセージ」を参照してください。

4.9 マッシュアップフレームワーク定義ファイル

マッシュアップフレームワークでは、以下の2種類の定義ファイルがあります。

- プロキシサーブレットのweb.xml
  ファイル: WEB-INF/web.xml
  詳細は、「5.4.2 web.xmlの設定」を参照してください。
- マッシュアップ定義ファイル
  ファイル: WEB-INF/conf/muf.xml
  詳細は、「付録B マッシュアップ定義ファイル」を参照してください。

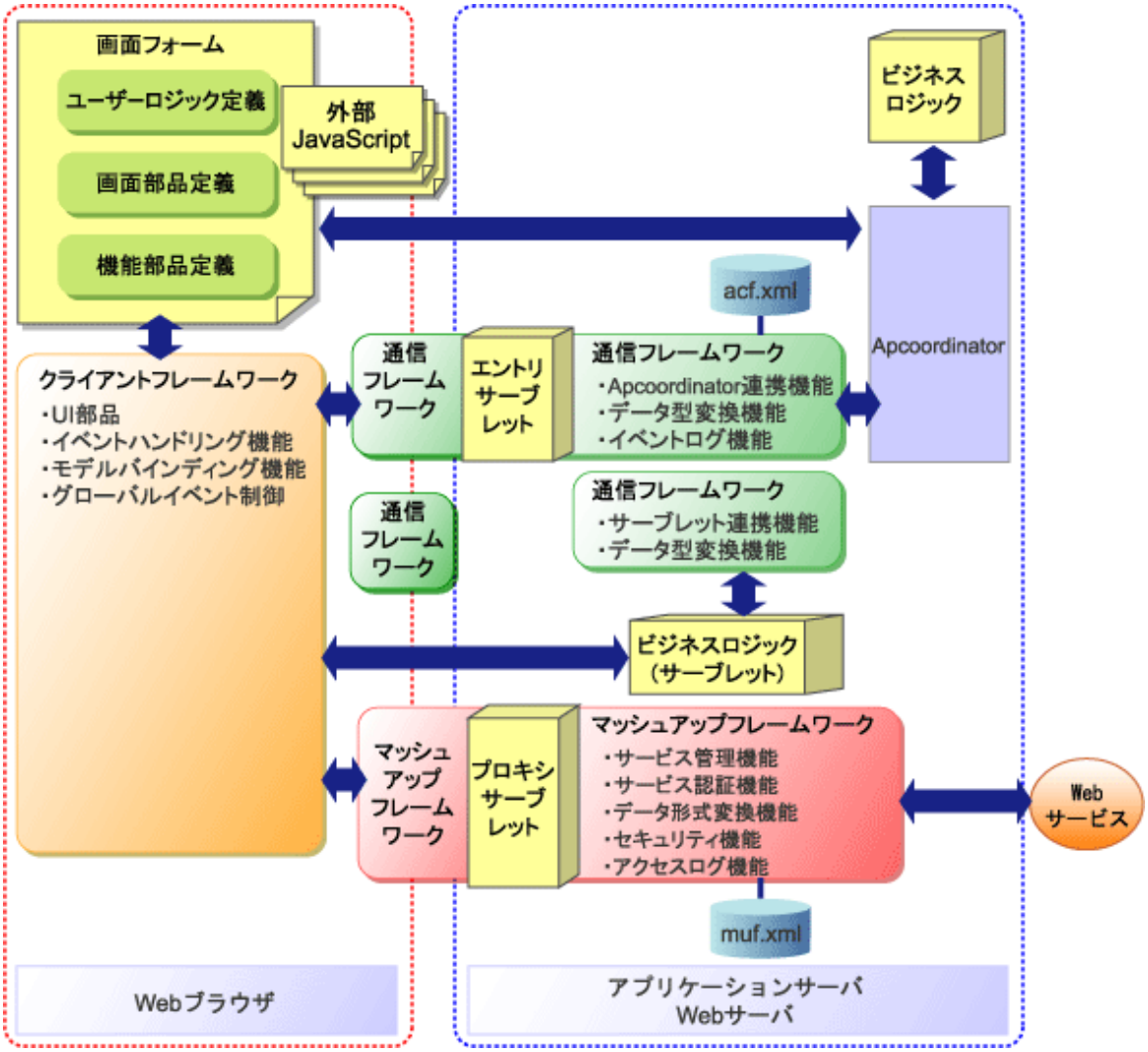
第5章 アプリケーションの開発

本章では、Ajaxフレームワークアプリケーションの開発方法について説明します。

5.1 アプリケーションの概要

以下の図に、Ajaxフレームワークアプリケーションの概要を示します。

図5.1 Ajaxフレームワークアプリケーションの概要



上図のうち、ユーザーが作成する資産は、以下のとおりです。

- 画面フォームおよび画面フォームに組み込まれる外部JavaScript
  クライアントで表示される画面、実行されるJavaScriptを作成します。画面フォームの機能構成については、「5.2 画面フォームの機能構成」を参照してください。
- エントリサーブレット
  通信フレームワークのサーバ機能の入り口となるサーブレットを作成します。エントリサーブレットの詳細は、「5.3 エントリサーブレット」を参照してください。
- ビジネスロジック
  Apcoordinator連携機能を利用する場合に、Apcoordinator連携で使用するデータBeanおよびビジネスクラスを作成します。ビジネスクラスには、ビジネスロジックを記述します。作成したデータBeanには、クライアントから送信されたデータが格納されます。データBeanおよびビジネスクラスの詳細は、「3.2.1 データBeanの作成」および「3.2.3 ビジネスクラスの作成」を参照してください。

- ビジネスロジック(サーブレット)
  サーブレット連携機能を利用する場合に、サーブレット連携で使用するJavaBeanおよびビジネスロジック(サーブレット)を作成します。作成したJavaBeanには、サーブレット連携機能のAPIを使用して、クライアントから送信されたデータを格納します。JavaBeanおよびビジネスロジックの詳細は、サーブレット連携機能で使用している通信方式に応じて、以下の項を参照してください。

  ー 汎用通信方式を利用している場合
    「3.4.1 JavaBeanの作成」および「3.4.2 ビジネスロジックの作成」

  ー 簡易通信方式を利用している場合
    「3.5.1 JavaBeanの作成」および「3.5.3 ビジネスロジックの作成」

  また、作成したビジネスロジック(サーブレット)を呼び出すためには、Servletの仕様に従ってweb.xmlを設定する必要があります。web.xmlの設定方法の詳細は、ご利用のアプリケーションサーバのマニュアルを参照してください。

- Ajaxフレームワーク環境定義ファイル(acf.xml)
  通信フレームワークの動作を定義するAjaxフレームワーク環境定義ファイルを作成します。Ajaxフレームワーク環境定義ファイルの詳細は、「付録A 環境定義ファイル」を参照してください。

- プロキシサーブレット
  マッシュアップフレームワークのサーバ機能の入り口となるサーブレットを設定します。プロキシサーブレットの詳細は、「5.4 プロキシサーブレット」を参照してください。

- マッシュアップ定義ファイル(muf.xml)
  マッシュアップフレームワークの動作を定義するマッシュアップ定義ファイルを作成します。マッシュアップ定義ファイルの詳細は、「付録B マッシュアップ定義ファイル」を参照してください。

Ajaxフレームワークアプリケーションを開発するのに必要な環境設定については、「5.5 開発環境の設定」を参照してください。

Ajaxフレームワークアプリケーションを実行するのに必要なファイルと、その構成・配置については、「5.6 実行環境の設定」を参照してください。

Ajaxフレームワークアプリケーションは、通常、Interstage StudioまたはEclipseを利用して開発します。アプリケーションの開発手順については、「5.7 Interstage Studioワークベンチを利用した開発」を参照してください。

5.2 画面フォームの機能構成

Ajaxフレームワークの画面フォームは、以下の機能から構成されます。

表5.1 画面フォームの機能構成

機能	形式	記述場所(注)		詳細
Ajaxフレームワーク宣言	htmlタグ	ー		「2.3 Ajaxフレームワークの宣言」を参照してください。
ユーザーデータ定義	JavaScript	画面固有	html内(任意)	「2.4.1 ユーザーデータの定義」を参照してください。
		画面間で共有	外部ファイル	
イベント定義	JavaScript	画面固有	html内(任意)	「2.4.2 ユーザーロジックの定義(イベント)」を参照してください。
		画面間で共有	外部ファイル	
イベントの登録	JavaScript	html内		「2.4.2 ユーザーロジックの定義(イベント)」を参照してください。
ユーザーロジック定義	JavaScript	画面固有	html内(任意)	「2.4.2 ユーザーロジックの定義(イベント)」を参照してください。
		画面間で共有	外部ファイル	
Apcoordinator連携	JavaScript	画面固有	html内(任意)	「3.2.6 ビジネスメソッドの実行」を参照してください。
		画面間で共有	外部ファイル	
サーブレット連携	JavaScript	画面固有	html内(任意)	汎用通信を利用している場合は「3.4.5 ビジネスロジックの実行」を参照してください。

機能	形式	記述場所(注)		詳細
				簡易通信を利用している場合は「3.5.6 ビジネスロジックの実行(非同期通信)」、「3.5.7 ビジネスロジックの実行(同期通信)」を参照してください。
イベント情報で定義した関数	JavaScript	画面固有	html内(任意)	「2.4.2 ユーザーロジックの定義(イベント)」を参照してください。
		画面間で共有	外部ファイル	
グローバルイベント制御を行う関数の定義	JavaScript	画面固有	html内(任意)	「2.6.2 グローバルイベント制御」を参照してください。
		画面間で共有	外部ファイル	
画面部品定義	htmlタグ	―		「2.5.1 画面情報定義部」を参照してください。
機能部品定義	htmlタグ	―		「2.5.2 機能定義部」を参照してください。
マッシュアップ	JavaScript	画面固有	html内(任意)	「4.2 クライアント通信API」を参照してください。
		画面間で共有	外部ファイル	

注) JavaScriptの推奨される記述場所を示します。
「画面固有」は、その画面だけで使用する機能であることを示します。
「画面間で共有」は、複数画面で共有する機能として設計し、実際に複数の画面で使用することを示します。

画面フォームで使用するコンバータおよびデータBeanについては、Ajaxフレームワーク環境定義ファイルへの定義が必要です。
データBeanの定義内容については「A.5 データBeanの定義(dataBeans)」、エディタによる定義方法については「データBeanに関する定義」を参照してください。
コンバータの定義内容については「A.6 コンバータ設定の定義(conversion)」、エディタによる定義方法については「コンバータに関する定義」を参照してください。

DOCTYPE宣言について

画面フォームをAjaxページエディタで編集する場合は、DOCTYPEを宣言する必要があります。
DOCTYPE宣言は、表示される画面フォームで1つだけ指定できます。このため、Apcoordinator連携を使用したアプリケーションの制御ページから、複数の画面フォームをインクルードすることはできません。制御ページの領域名は、複数個指定しないでください。

5.3 エントリサーブレット

エントリサーブレットは、通信フレームワークが提供する機能の入り口となるサーブレットです。

5.3.1 ユーザー定義エントリサーブレット

エントリサーブレットを利用する場合、通信フレームワークが提供する以下のAcfServletクラスを継承したユーザー定義エントリサーブレットを作成する必要があります。

クラス名:

AcfServlet

パッケージ名:

com.fujitsu.interstage.rcf

格納されているjarファイル:

Windows32/64

C:¥InteractionManager¥rcf¥lib¥ujircf.jar

C:¥InteractionManagerは標準のインストールフォルダです。製品のインストールフォルダを変更した場合は、読み替えてください。

Solaris32/64

/opt/FJSVrcf/lib/ujircf.jar

/optは標準のインストールディレクトリです。製品のインストールディレクトリを変更した場合は、読み替えてください。

Linux32/64

/opt/FJSVrcf/lib/ujircf.jar

エントリサーブレットのひな形は、Ajaxフレームワークプロジェクトウィザードで[エントリサーブレット情報]を指定することによって作成されます。

基本的な動作を担うエントリサーブレットは、AcfServletを継承したサーブレットを定義するだけです。特定のメソッドを実装する必要はありません。

5.3.2 web.xmlの設定

web.xmlに、エントリサーブレットのマッピング情報を記述します。

ユーザー定義エントリサーブレットクラスをservlet-class要素に登録し、servlet-mappingを利用して、/acf/*として参照できるようにしてください。

以下に、web.xmlの記述例を示します。

```
<servlet>
   <!--ユーザー定義エントリサーブレット -->
   <servlet-name>my-acf-servlet</servlet-name>
   <servlet-class>mypkg.MyAcfServlet</servlet-class>
   <load-on-startup>3</load-on-startup>
</servlet>

<servlet-mapping>
   <!--エントリサーブレットにマップ-->
   <servlet-name>my-acf-servlet</servlet-name>
   <url-pattern>/acf/*</url-pattern>
</servlet-mapping>
```

5.3.3 起動直後の動作

エントリサーブレットは、コンテナが起動した段階で初期化処理が開始され、環境定義ファイルをチェックします。問題を検出した場合、ログにエラーメッセージを出力し、初期化処理を中止します。

初期化処理が中止されたあとにリクエストが送信された場合は、サーブレットコンテナにエラーが通知されます。

以下にチェック対象と、チェック内容の概要を示します。

順番	チェック対象	概要
1	WEB-INF/ web.xml	Servletコンテナに任せ、独自のチェックは行いません。
2	WEB-INF/acf.xml	詳細なチェックを行います。 問題が検出された場合のエラーメッセージについては、「J.3.2 環境定義ファイルに関するメッセージ」を参照してください。

5.4 プロキシサーブレット

プロキシサーブレットは、マッシュアップフレームワークが提供する機能の入り口となるサーブレットです。

5.4.1 サービス・運用情報の設定

プロキシサーブレットを利用する場合、マッシュアップ定義ファイルに以下の情報を設定する必要があります。

- サービスの情報(サービスのID、アダプタ名など)

・ 運用情報(プロキシの設定、アクセスログの出力情報など)

マッシュアップ定義ファイルの設定方法については「5.7.11 マッシュアップ定義ファイルの編集」を、マッシュアップ定義ファイルの設定内容については「付録B マッシュアップ定義ファイル」を参照してください。

5.4.2 web.xmlの設定

web.xmlに、プロキシサーブレットのマッピング情報を記述します。

プロキシサーブレットクラスをservlet-class要素に登録し、servlet-mappingを利用して、/muf/proxyとして参照できるようにしてください。

以下に、web.xmlの記述例を示します。

```
<servlet>
   <servlet-name>MUProxyServlet</servlet-name>
   <servlet-class>com.fujitsu.interstage.rcf.mashup.proxy.MUProxyServlet</servlet-class>
   <load-on-startup>3</load-on-startup>
</servlet>

<servlet-mapping>
   <servlet-name>MUProxyServlet</servlet-name>
   <url-pattern>/muf/proxy</url-pattern>
</servlet-mapping>
```

5.4.3 起動直後の動作

プロキシサーブレットは、コンテナが起動した段階で初期化処理が開始され、マッシュアップフレームワークの定義ファイルをチェックします。問題を検出した場合、ログにエラーメッセージを出力し、初期化処理を中止します。

初期化処理が中止されたあとにリクエストが送信された場合は、サーブレットコンテナにエラーが通知されます。

以下にチェック対象と、チェック内容の概要を示します。

順番	チェック対象	概要
1	WEB-INF/ web.xml	Servletコンテナに任せ、独自のチェックは行いません。
2	WEB-INF/conf/ muf.xml	詳細なチェックを行います。 問題が検出された場合のエラーメッセージについては、「J.3.2 環境定義ファイルに関するメッセージ」を参照してください。

5.5 開発環境の設定

ここでは、Ajaxフレームワークアプリケーションの開発環境の設定について説明します。

ブラウザの設定

Ajaxフレームワークアプリケーションの開発で以下の機能を使用する場合は、ブラウザの設定が必要です。

・ ブラウザを使用した実行・デバッグ

・ Ajaxページエディタによる画面フォームの編集

・ モックアップ開発

ブラウザの設定に関して、以下の点に注意してください。

・ Internet Explorerを使用する場合
以下の手順で、オプションを確認してください。

1. [ツール]メニューから[インターネットオプション]ダイアログボックスを開き、[セキュリティ]タブをクリックします。

2. Ajaxフレームワークアプリケーションが動作するゾーンを選択し、[レベルのカスタマイズ]ボタンをクリックします。
   ここで、以下のオプションが[ダイアログを表示する]または[有効にする]となっている必要があります。(注)

   a. [ActiveXコントロールとプラグインの実行]

   b. [スクリプトを実行しても安全だとマークされているActiveXコントロールのスクリプトの実行]

   c. [アクティブスクリプト]

   注) [ダイアログを表示する]を選択した場合は、画面表示時に確認のためのダイアログボックスが表示されます。メッセージの内容を確認して問題がなければ、[はい]ボタンをクリックしてください。

- Firefoxを使用する場合
  以下の手順で、オプションを確認してください。

  1. アドレスバーにabout:configと入力し、Enterキーを押します。
  2. 設定値のjavascript.enabledの値がtrueになっていることを確認します。
     trueになっていない場合は、javascript.enabledの値をfalseからtrueに変更して、Webページをリロードします。

注意

ブラウザの設定を変更する場合は、ブラウザのセキュリティレベルが下がる可能性があるため、企業内のセキュリティポリシーなどを考慮したうえで変更してください。また、システムを利用したあとは、ブラウザの設定を元に戻しておいてください。

5.6 実行環境の設定

ここでは、Ajaxフレームワークアプリケーションの実行環境の設定について説明します。

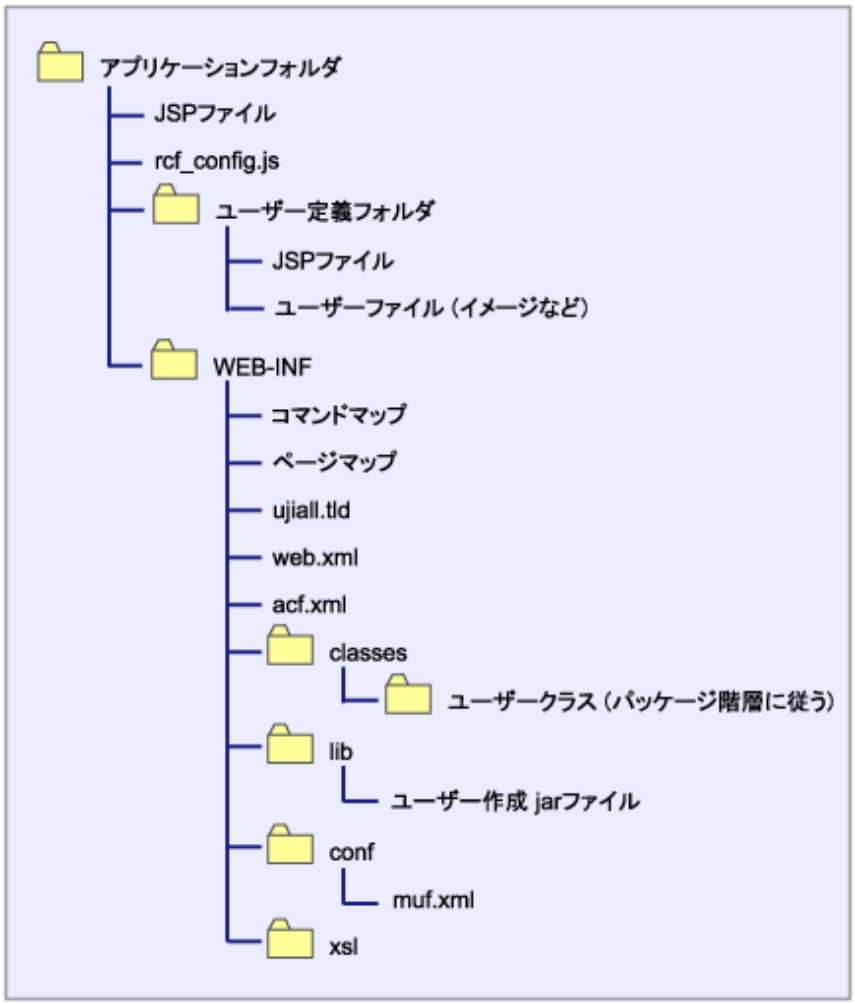
参考

- Java EE 5のアプリケーションを配備する手順については、「Interstage Application Server/Interstage Web Server Express Java EE運用ガイド」の「第4章 Java EEアプリケーションの運用」を参照してください。
- Java EE 6のアプリケーションを配備する手順については、「Interstage Application Server/Interstage Web Server Express Java EE運用ガイド(Java EE 6編)」の「第4章 Java EEアプリケーションの運用」を参照してください。

5.6.1 実行時のファイル配置

以下の図に、Ajaxフレームワークアプリケーションの一般的なファイル配置を示します。この構成のファイル、または、この構成のwarファイルを配置してください。

図5.2 Ajaxフレームワークアプリケーションのファイル配置



- JSPファイル
  通常は、アプリケーションフォルダ直下に配置します。
  サブフォルダに配置することもできますが、その場合は、以下に示す定義についてパスの考慮が必要です。
  - Ajaxフレームワークの動作定義で指定するrcf_config.jsのパス
  - Ajaxフレームワークの初期化処理で指定するacf/file/rcf/rcf.jsのパス
  - ページマップで指定するJSPファイルのパス
  - JSPファイルからほかのファイルを参照するときのパス
- rcf_config.jsファイル
  Ajaxフレームワークの動作オプションを定義するファイルです。詳細は、「2.3.2 Ajaxフレームワークの動作定義」を参照してください。
- ユーザーファイル
  イメージなどの静的ドキュメントは、アプリケーションフォルダ直下にも、サブフォルダにも配置できます。
- Apcoordinator関連のファイル
  Apcoordinator連携機能を利用する場合に必要なファイルです。
  関係定義ファイル(コマンドマップ/ページマップ)およびタグライブラリファイル(ujiall.tld)は、WEB-INFフォルダに配置します。WEB-INFフォルダは、Webアプリケーションの情報を配置するための特別なフォルダで、クライアントから直接参照することが禁止されています。
- Webアプリケーション環境定義ファイル(web.xml)
  web.xmlは、WEB-INFフォルダの直下に配置します。

- Ajaxフレームワーク環境定義ファイル(acf.xml)
  サーバ側で動作する通信フレームワークの動作定義ファイルです。詳細は、「付録A 環境定義ファイル」を参照してください。
- ユーザークラスファイル
  ユーザー定義のクラスファイルは、WEB-INFのclassesフォルダの下に、パッケージ階層に従って配置します。例えば、sampleパッケージのSampleHandler.classの配置場所は、WEB-INF¥classes¥sampleの下になります。
- ユーザー作成jarファイル
  ユーザー定義のクラスファイルをjarファイルに結合した場合は、WEB-INFのlibフォルダの下に配置します。
- マッシュアップ定義ファイル(muf.xml)
  マッシュアップフレームワークの動作定義ファイルです。WEB-INFのconfフォルダの下に配置します。詳細は、「付録B マッシュアップ定義ファイル」を参照してください。
- xslフォルダ
  スクレイピングツールで作成されたXSLファイルが格納されます。スクレイピングツールの詳細は、「付録G スクレイピングツール」を参照してください。

5.6.2 実行時に必要なファイル

ここでは、アプリケーションの実行時に必要なファイルについて説明します。

Ajaxフレームワークを利用したアプリケーションの場合

Ajaxフレームワークを利用したアプリケーションの実行時には、以下のライブラリをクラスパスで指定してください。

Windows32/64

C:¥InteractionManager¥rcf¥lib¥ujircf.jar
C:¥InteractionManager¥APC¥lib¥uji.jar

または

C:¥InteractionManager¥rcf¥lib¥ujircf.jar
C:¥Interstage¥APC¥lib¥uji.jar

C:¥InteractionManagerおよびC:¥Interstageは標準のインストールフォルダです。製品のインストールフォルダを変更した場合は、読み替えてください。

Solaris32/64

/opt/FJSVrcf/lib/ujircf.jar
/opt/FJSVwebc/lib/uji.jar

/optは標準のインストールディレクトリです。製品のインストールディレクトリを変更した場合は、読み替えてください。

Linux32/64

/opt/FJSVrcf/lib/ujircf.jar
/opt/FJSVwebc/lib/uji.jar

マッシュアップフレームワークを利用したアプリケーションの場合

マッシュアップフレームワークを利用したアプリケーションの実行時には、以下のライブラリをクラスパスで指定してください。

Windows32/64

C:¥InteractionManager¥rcf¥lib¥ujircf.jar
C:¥InteractionManager¥rcf¥lib¥muf.jar
C:¥InteractionManager¥rcf¥lib¥Tidy.jar
C:¥InteractionManager¥APC¥lib¥uji.jar

または

C:¥InteractionManager¥rcf¥lib¥ujircf.jar
C:¥InteractionManager¥rcf¥lib¥muf.jar
C:¥InteractionManager¥rcf¥lib¥Tidy.jar
C:¥Interstage¥APC¥lib¥uji.jar

C:¥InteractionManagerおよびC:¥Interstageは標準のインストールフォルダです。製品のインストールフォルダを変更した場合は、読み替えてください。

Solaris32/64

/opt/FJSVrcf/lib/ujircf.jar
/opt/FJSVrcf/lib/muf.jar
/opt/FJSVrcf/lib/Tidy.jar
/opt/FJSVwebc/lib/uji.jar

/optは標準のインストールディレクトリです。製品のインストールディレクトリを変更した場合は、読み替えてください。

Linux32/64

/opt/FJSVrcf/lib/ujircf.jar
/opt/FJSVrcf/lib/muf.jar
/opt/FJSVrcf/lib/Tidy.jar
/opt/FJSVwebc/lib/uji.jar

参考

クラスパスの指定方法については、「スタートガイド」の「第2章 サンプルアプリケーションの利用」を参照してください。

5.6.3 ブラウザの設定

アプリケーション実行時には、開発環境と同様にブラウザを設定します。また、アプリケーションを利用するエンドユーザーのブラウザも、同様に設定する必要があります。
詳細は、「5.5 開発環境の設定」を参照してください。

5.7 Interstage Studioワークベンチを利用した開発

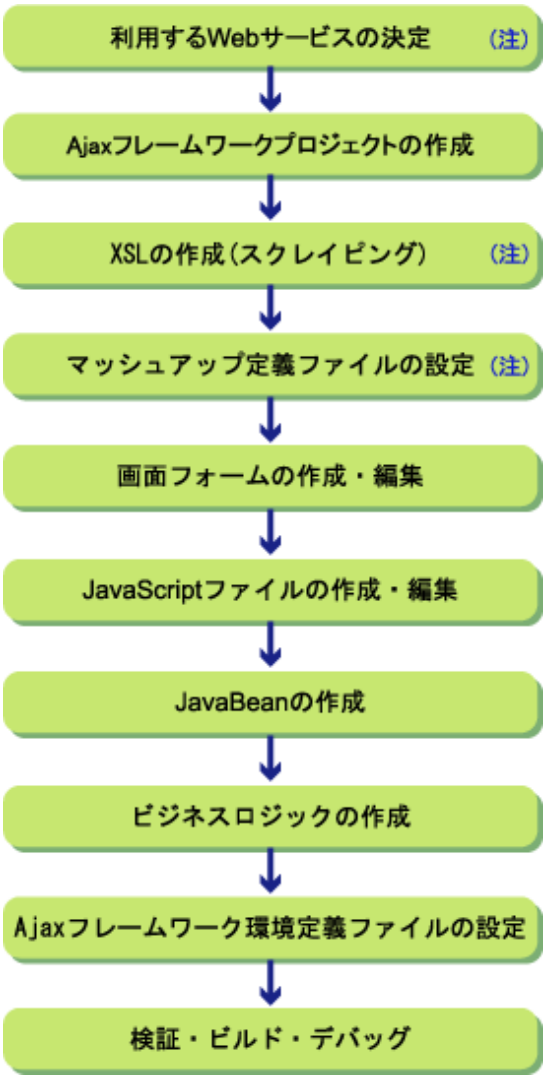
Interstage Studio ワークベンチを利用して、Ajaxフレームワークアプリケーションを開発する場合、以下の手順で開発します。

注意

Interstage Studioワークベンチを使用して、Ajaxフレームワークを利用したアプリケーションを開発する場合は、プロジェクトのInterstage Java EE検証の設定について、以下の値となるように指定してください。

- [Interstage Java EE検証を自動的に実行する]が選択されている場合は、選択を解除して指定を取り外します。
- [JSPをチェックする]が選択されている場合は、選択を解除して指定を取り外します。
- [配備前にチェックする]が選択されている場合は、選択を解除して指定を取り外します。
- [差分ビルドではチェックしない]が選択されていない場合は、選択します。

図5.3 Interstage Studioワークベンチを利用した開発手順



注) マッシュアップフレームワークを利用する場合にだけ必要です。

1. 利用するWebサービスの決定
   Ajaxフレームワークアプリケーションで利用するWebサービスを決定します。
   この作業は、マッシュアップフレームワークを利用する場合にだけ必要です。

2. Ajaxフレームワークプロジェクトの作成
   Ajaxフレームワークアプリケーションを開発するためのプロジェクトを作成します。
   詳細は、「5.7.1 Ajaxフレームワークプロジェクトの作成」を参照してください。

3. XSLの作成(スクレイピング)
   スクレイピングツールを利用して、Webアプリケーションをスクレイピングし、XSLファイルを作成します。
   この作業は、マッシュアップフレームワークを利用する場合にだけ必要です。
   詳細は、「付録G スクレイピングツール」を参照してください。

4. マッシュアップ定義ファイルの設定
   マッシュアップ定義ファイルに必要な設定を行います。
   この作業は、マッシュアップフレームワークを利用する場合にだけ必要です。
   マッシュアップ定義ファイルの設定の詳細は、「5.7.11 マッシュアップ定義ファイルの編集」を参照してください。

5. 画面フォームの作成・編集
   Ajaxフレームワークアプリケーションで使用する画面フォームを作成します。
   画面フォーム(ひな形)の作成の詳細は、「5.7.2 画面フォーム(ひな形)の作成」を参照してください。画面フォームの編集の詳細は、「5.7.3 画面フォームの編集」を参照してください。

ユーザーデータ/ユーザーロジックを定義する外部ファイルの作成については、「5.7.4 ユーザーロジック定義の作成」を参照してください。

6. JavaScriptファイルの作成・編集
   Ajaxフレームワークアプリケーションで使用するJavaScriptファイルを作成します。
   JavaScriptファイルの作成の詳細は、「5.7.5 JavaScriptファイルの作成」を参照してください。JavaScriptファイルの編集の詳細は、「5.7.6 JavaScriptファイルの編集」を参照してください。

7. JavaBeanの作成
   JavaBeanを作成します。
   詳細は、「5.7.7 JavaBeanの作成」を参照してください。

8. ビジネスロジックの作成
   ビジネスロジックを作成します。
   詳細は、「5.7.8 ビジネスロジックの作成」を参照してください。

9. Ajaxフレームワーク環境定義ファイルの設定
   Ajaxフレームワーク環境定義ファイルに必要な設定を行います。
   Ajaxフレームワーク環境定義ファイルの設定の詳細は、「5.7.9 Ajaxフレームワーク環境定義ファイルの作成」および「5.7.10 Ajaxフレームワーク環境定義ファイルの編集」を参照してください。

10. 検証・ビルド・デバッグ
    環境定義の検証、実行環境のビルド、アプリケーションのデバッグを行います。
    詳細は、それぞれ、「5.7.12 検証」、「5.7.13 ビルド」、「5.7.14 実行・デバッグ」を参照してください。

ポイント

画面フォームの文字コードは、UTF-8を推奨します。UTF-8以外の文字コードを使用する場合は、「5.11 文字コードについて」を参照してください。

参考

- Java EE 5のアプリケーションを開発する手順については、「Interstage Studio ユーザーズガイド」の「第6章 Java EE 5アプリケーション共通事項」を参照してください。
- Java EE 6のアプリケーションを開発する手順については、「Interstage Studio ユーザーズガイド」の「第9章 Java EE 6アプリケーションを開発する」を参照してください。

5.7.1 Ajaxフレームワークプロジェクトの作成

Ajaxフレームワークアプリケーションを開発するためのプロジェクトは、Ajaxフレームワークプロジェクトウィザードで作成します。

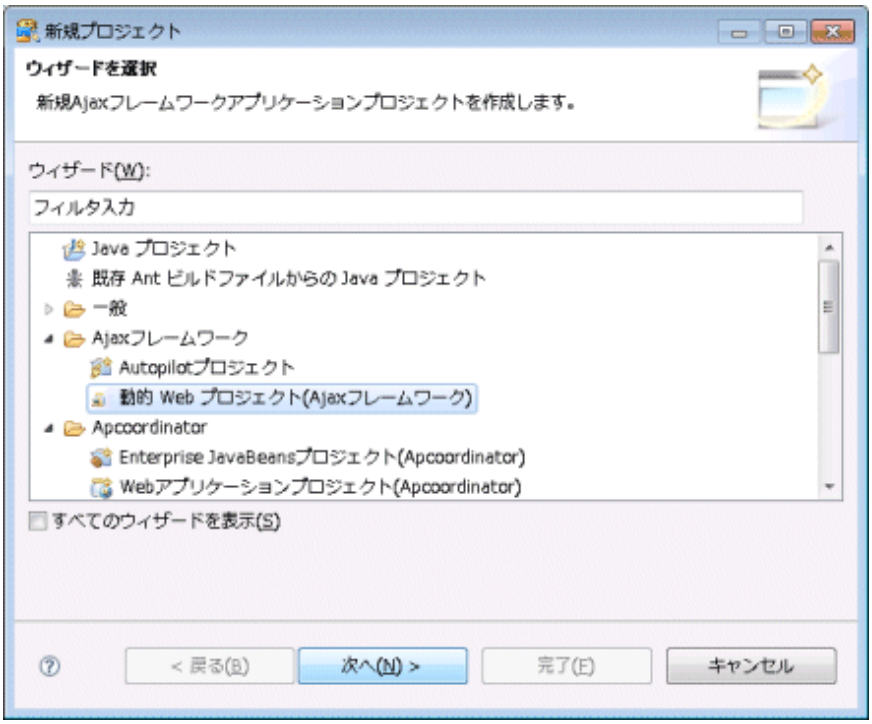
Ajaxフレームワークプロジェクトは、以下の手順で作成します。

1. [新規プロジェクト]ウィザードで、「動的Webプロジェクト(Ajaxフレームワーク)」を選択します。
2. [新規動的Webプロジェクト]ウィザードで、以下を設定します。
   - [動的Webプロジェクト]ページで、プロジェクトの基本情報を設定します。
   - [Webモジュール]ページで、Webアプリケーションの情報を設定します。
   - [エントリサーブレット情報]ページで、エントリサーブレットの情報を設定します。

以下に、それぞれのウィザードおよびページについて説明します。

新規プロジェクトウィザード

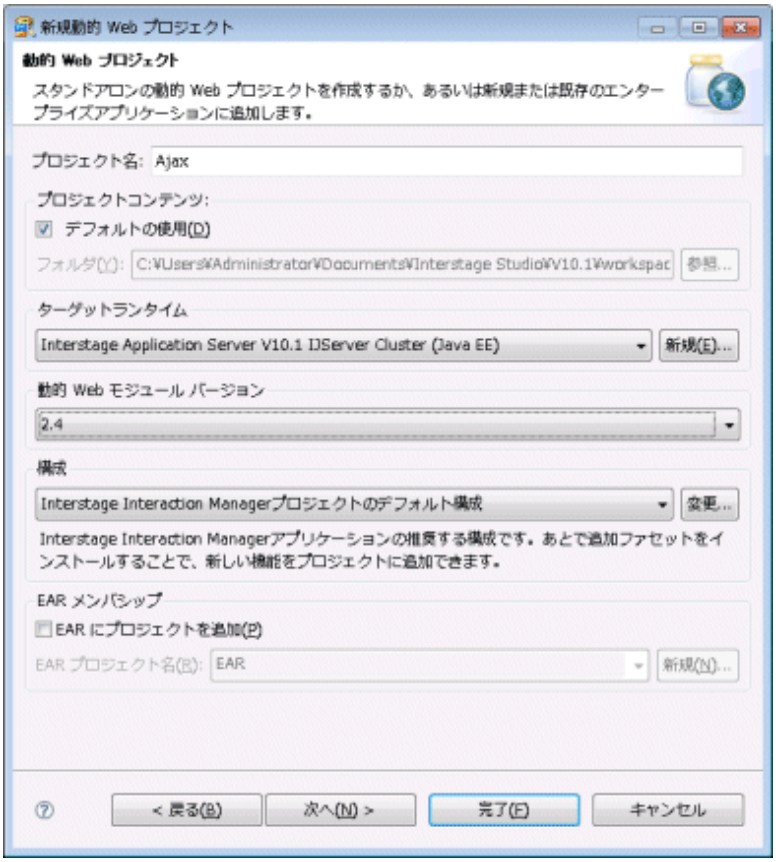
[ファイル]メニューの[新規] > [プロジェクト]で起動します。



「動的Webプロジェクト(Ajaxフレームワーク)」を選択し、[次へ]ボタンをクリックします。

動的Webプロジェクト

プロジェクトの基本情報を指定します。



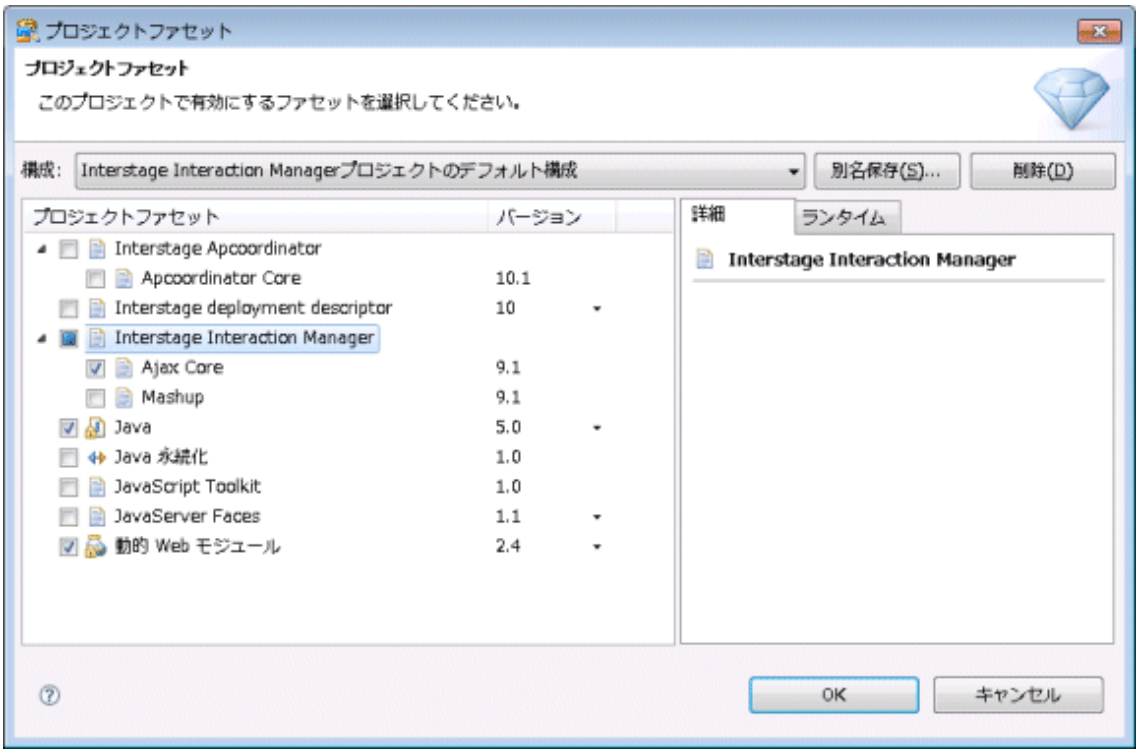
指定する項目は、以下のとおりです。

項目		説明
プロジェクト名		作成するプロジェクトの名前を指定します。
プロジェクトコンテンツ		プロジェクトコンテンツの格納フォルダを指定します。
	デフォルトの使用	デフォルトのフォルダに格納されます。
	フォルダ	デフォルトを使用しない場合に、フォルダを指定します。
ターゲットランタイム		Java EEのアプリケーションを動作させるランタイムを選択します。
動的Webモジュールバージョン		Webモジュールのバージョンを指定します。デフォルトは「2.4」です。
構成		プロジェクトの構成を選択します。ここでは、「Interstage Interaction Managerプロジェクトのデフォルト構成」を選択します。 以下の場合、[変更]ボタンをクリックして[プロジェクトファセット]ダイアログボックスで、構成を変更してください。 ・ マッシュアップフレームワーク機能を利用する 詳細は「プロジェクトファセットダイアログボックス」を参照してください。
EARメンバシップ		モジュールやライブラリをEARファイルにまとめる場合は、エンタープライズアプリケーションのプロジェクトを選択します。ここでは、チェックしません。
	EARにプロジェクトを追加	
	EARプロジェクト名	

指定後、[次へ]ボタンをクリックすると、[Webモジュール]ページが表示されます。

プロジェクトファセットダイアログボックス

プロジェクトの機能構成を指定します。



利用する機能に応じて、以下のファセットを指定してください。

項目	説明
Interstage Interaction Manager	Interstage Interaction Managerを利用するプロジェクトの場合、チェックします。

	Ajax Core	Ajaxフレームワークを利用するプロジェクトの場合、チェックします。
	Mashup	マッシュアップフレームワークを利用するプロジェクトの場合、チェックします。

Webモジュール

Webモジュールの情報を指定します。

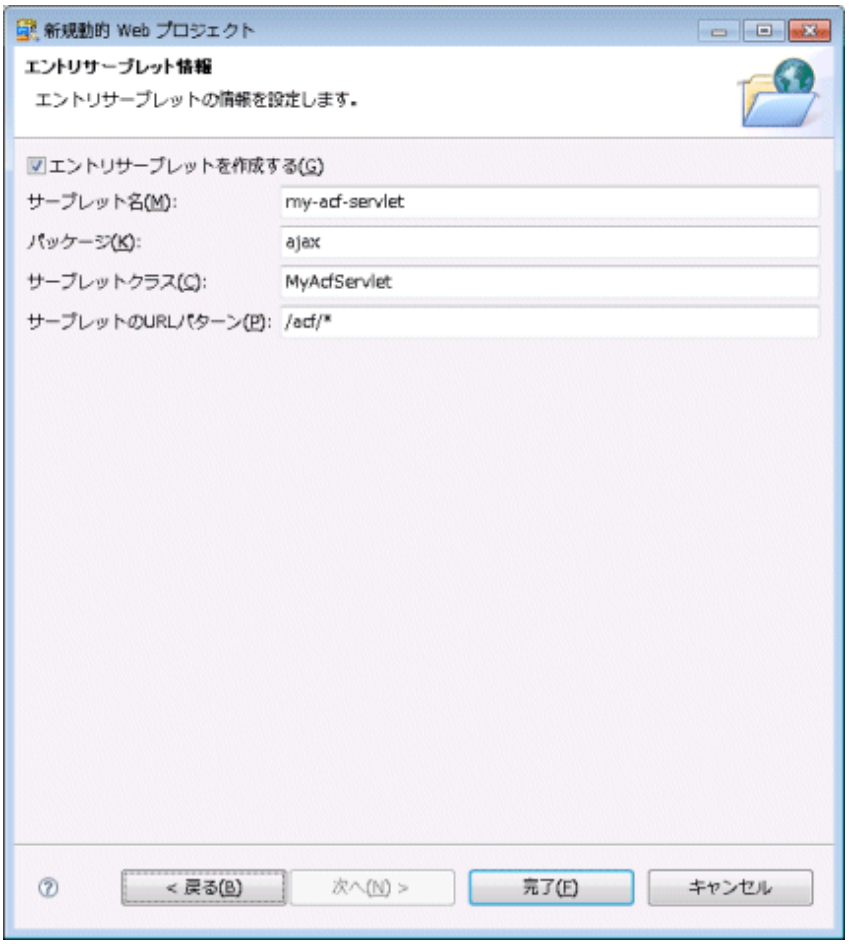
指定する項目は、以下のとおりです。

項目	説明
コンテキストルート	Webサーバに配備されるときの最上位フォルダを指定します。任意のコンテキストルート名を入力してください。
コンテンツフォルダ	Webアプリケーションとして配備するすべてのリソースを格納するフォルダを指定します。ここでは、任意のコンテンツフォルダ名を入力してください。
Javaソースフォルダ	プロジェクトのソースを格納するフォルダを指定します。ここでは、任意のJavaソースフォルダ名を入力してください。

項目を指定し、[次へ]ボタンをクリックします。

エントリサーブレット情報

エントリサーブレットの情報を指定します。



指定する項目は、以下のとおりです。

項目	説明
エントリサーブレットを作成する	エントリサーブレットを作成する場合に指定します。
サーブレット名	作成するサーブレットの名前を指定します。
パッケージ	サーブレットのパッケージ名を指定します。
サーブレットクラス	サーブレットのクラス名を指定します。
サーブレットのURLパターン	サーブレットのURLパターンを指定します。

項目を指定し、[完了]ボタンをクリックします。

5.7.2 画面フォーム(ひな形)の作成

Ajaxフレームワークアプリケーションで使用する画面フォーム(JSPファイル)のひな形は、以下のどちらかのウィザードを使用して作成します。

- Ajax JSPウィザード
- JSPウィザード

注意

JSPウィザードは、文字コードにUTF-8以外を利用したい場合に使用します。文字コードをUTF-8以外にする方法は、「5.11 文字コードについて」を参照してください。

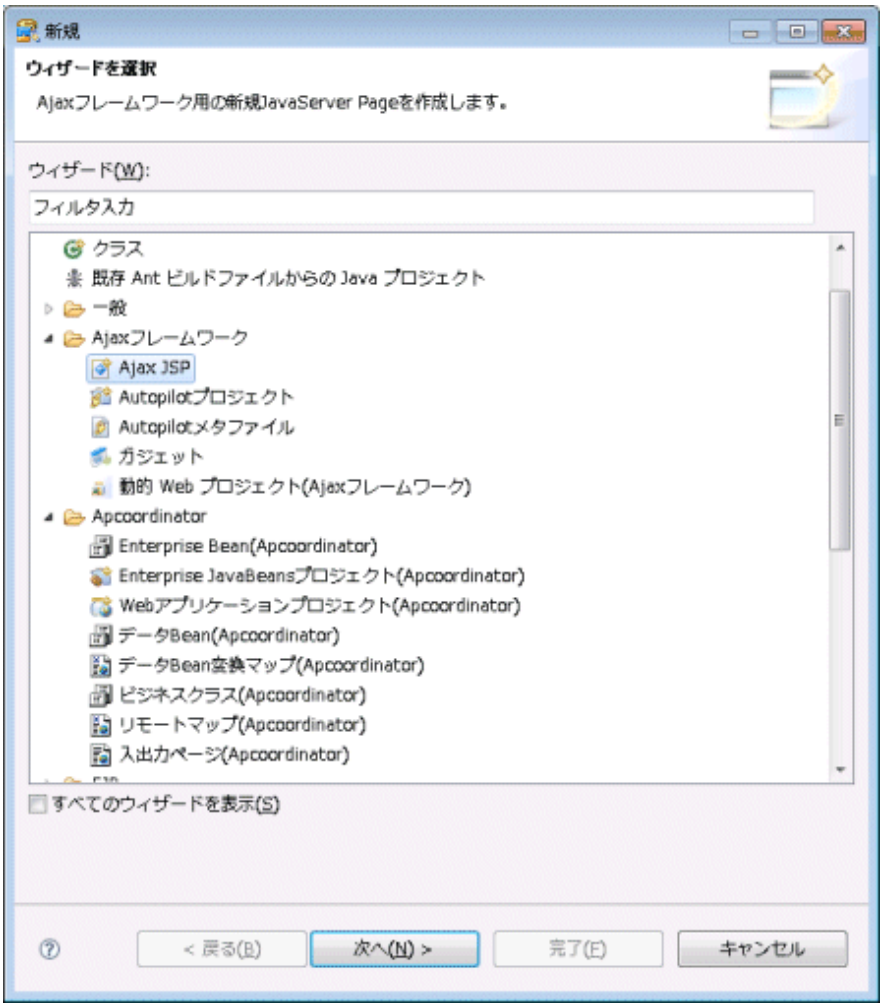
Ajax JSPウィザードで作成する場合

以下の手順で、Ajaxフレームワークの画面フォームのひな形を作成します。

1. [新規]ウィザードで、[Ajax JSP]を選択します。または、[ファイル]メニューから[新規] > [Ajax JSP]を選択します。
2. [Ajax JavaServer Page]ウィザードで、ファイル名と作成先を指定します。

新規ウィザード

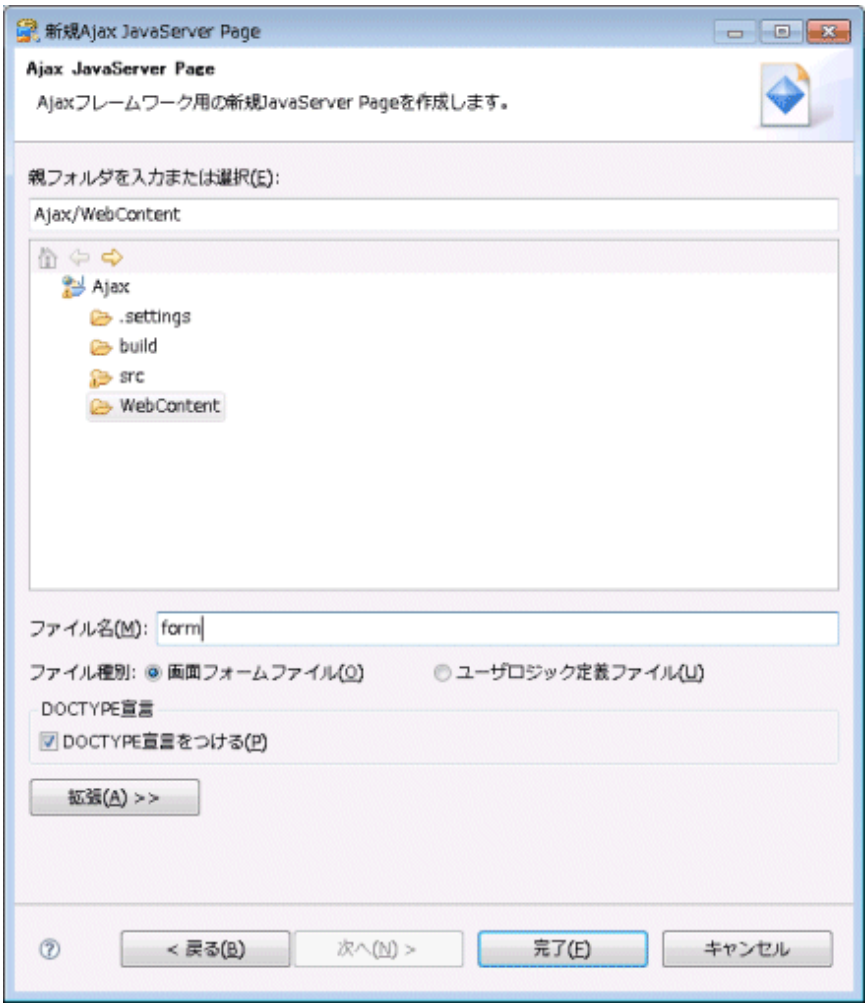
[ファイル]メニューの[新規] > [その他]で起動します。



「Ajax JSP」を選択し、[次へ]ボタンをクリックします。

Ajax JavaServer Pageウィザード

作成するJSPのファイル名と格納先を指定します。



指定する項目は、以下のとおりです。

項目	説明
親フォルダを入力または選択	作成先フォルダを入力またはリストから選択します。
ファイル名	作成するJSPのファイル名を指定します。
ファイル種別	画面フォームファイルとユーザーロジック定義ファイルのどちらを作成するかを指定します。ここでは、[画面フォームファイル]を選択してください。
DOCTYPE宣言をつける	作成する画面フォームでDOCTYPEを宣言するかどうかを指定します。(注)

注) 画面フォームをAjaxページエディタで編集する場合は、画面フォームでDOCTYPEを宣言してください。

項目を指定し、[完了]ボタンをクリックします。

JSPウィザードで作成する場合

JSPウィザードでAjaxフレームワーク画面フォームのテンプレートを選択しても、画面フォームのひな形を作成できます。

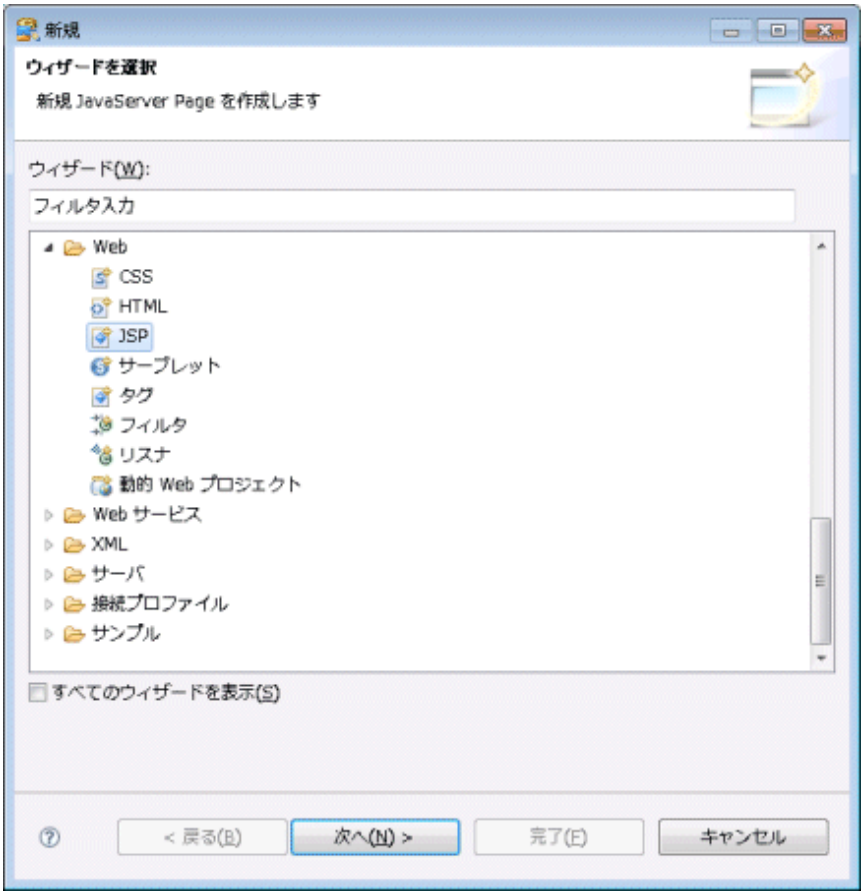
以下に、画面フォームの作成手順を示します。

1. [新規]ウィザードで、[Web] > [JSP]を選択します。
2. [JavaServer Page]ウィザードで、以下を設定します。
   - [JavaServer ページ]ページで、ファイル名と作成先を指定します。

ー [JSPテンプレートの選択]ページで、「新規JSPファイル(Ajaxフレームワーク サーブレット連携 画面フォーム定義)」を選択します。

新規ウィザード

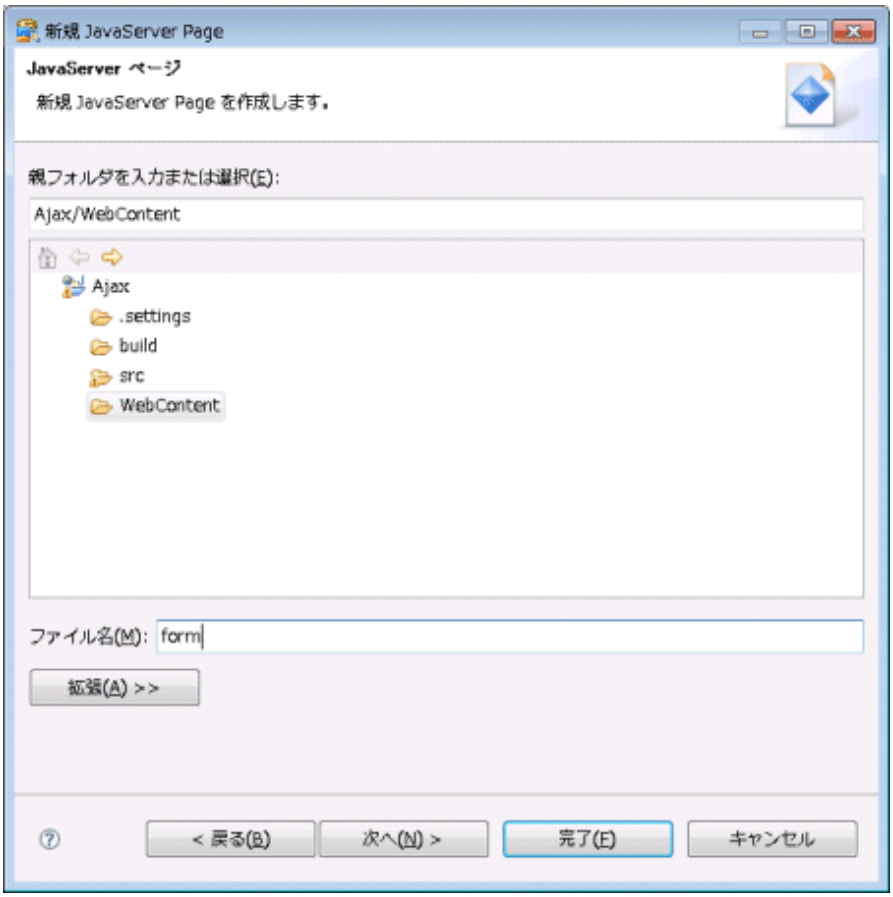
[ファイル]メニューの[新規] > [その他]で起動します。



「JSP」を選択し、[次へ]ボタンをクリックします。

JavaServer ページ

作成するJSPのファイル名と格納先を指定します。



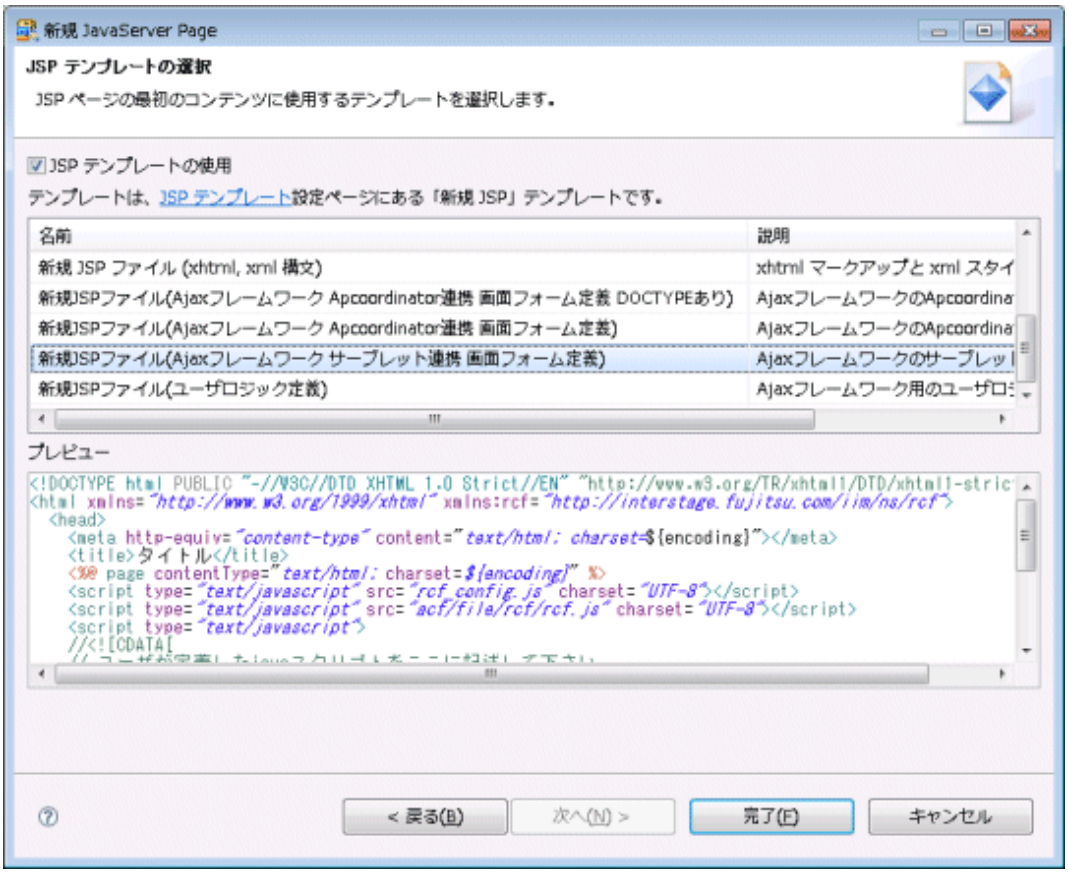
指定する項目は、以下のとおりです。

項目	説明
親フォルダを入力または選択	作成先フォルダを入力またはリストから選択します。
ファイル名	作成するJSPのファイル名を指定します。

項目を指定し、[次へ]ボタンをクリックします。

JSPテンプレートの選択

JSPの作成に使用するテンプレートを選択します。



「新規JSPファイル(Ajaxフレームワーク サーブレット連携 画面フォーム定義)」を選択し、[完了]ボタンをクリックします。

ここで生成されるJSPファイルの文字コードは、Interstage Studioワークベンチで JSPファイルに対して設定されたコード系になります。文字コードの確認および変更方法は、以下のとおりです。

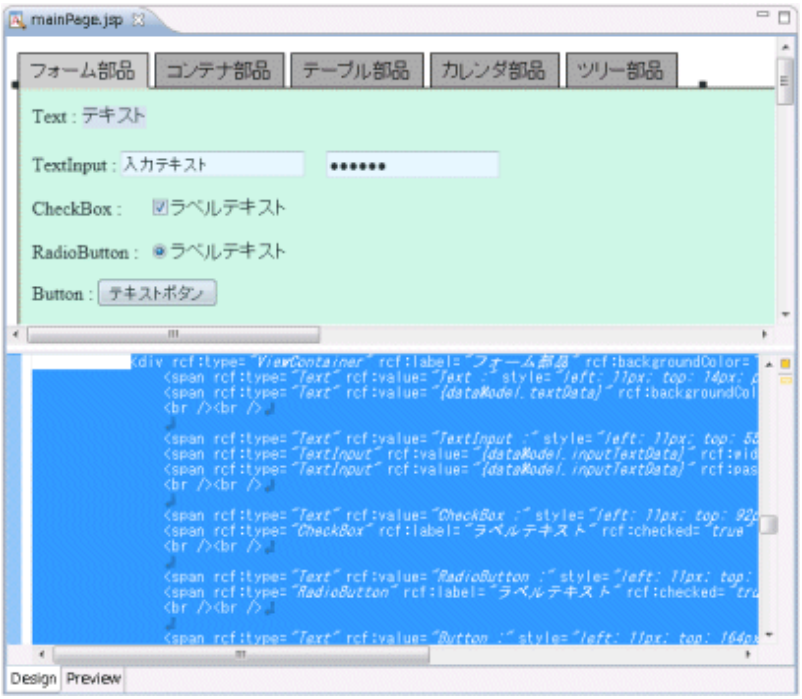
1. [ウィンドウ]メニューから[設定]を選択します。
2. [設定]ダイアログボックスの設定項目で、[Web] > [JSPファイル]を選択します。
3. JSPファイルの作成時のエンコードを確認し、必要に応じて修正してください。

5.7.3 画面フォームの編集

Ajaxフレームワークの画面フォーム(JSPファイル)は、Ajaxページエディタで編集します。

Ajaxページエディタは、プロジェクトエクスプローラからAjaxフレームワークの画面フォームのJavaServer Pageファイルをダブルクリックするか、コンテキストメニューの[開く]を選択すると起動します。

以下に、Ajaxページエディタのビューを示します。



Ajaxページエディタは、以下の機能を提供します。

- ビジュアル編集機能
  実行イメージを確認しながら、WYSIWYGで業務画面を編集することができます。
- テキスト編集機能
  ビジュアル編集機能と連動したテキスト編集を行うことができます。
- イベント処理連携機能
  部品から[イベント処理定義]ダイアログボックスを起動し、部品に対する操作発生時の処理を入力できます。
- モデルバインディング機能
  部品から[モデルバインディング定義]ダイアログボックスを起動し、モデルや画面部品とのデータバインディングを指定できます。

Ajaxページエディタの詳細は、「付録F Ajaxページエディタ」を参照してください。

5.7.4 ユーザーロジック定義の作成

複数画面で共通に使用するユーザーデータ/ユーザーロジック定義は、以下のウィザードのどれかを使用して作成します。

- Ajax JSPウィザード
- HTMLウィザード
- JSPウィザード

注意

HTMLウィザードおよびJSPウィザードは、文字コードにUTF-8以外を利用したい場合に使用します。文字コードをUTF-8以外にする方法は、「5.11 文字コードについて」を参照してください。

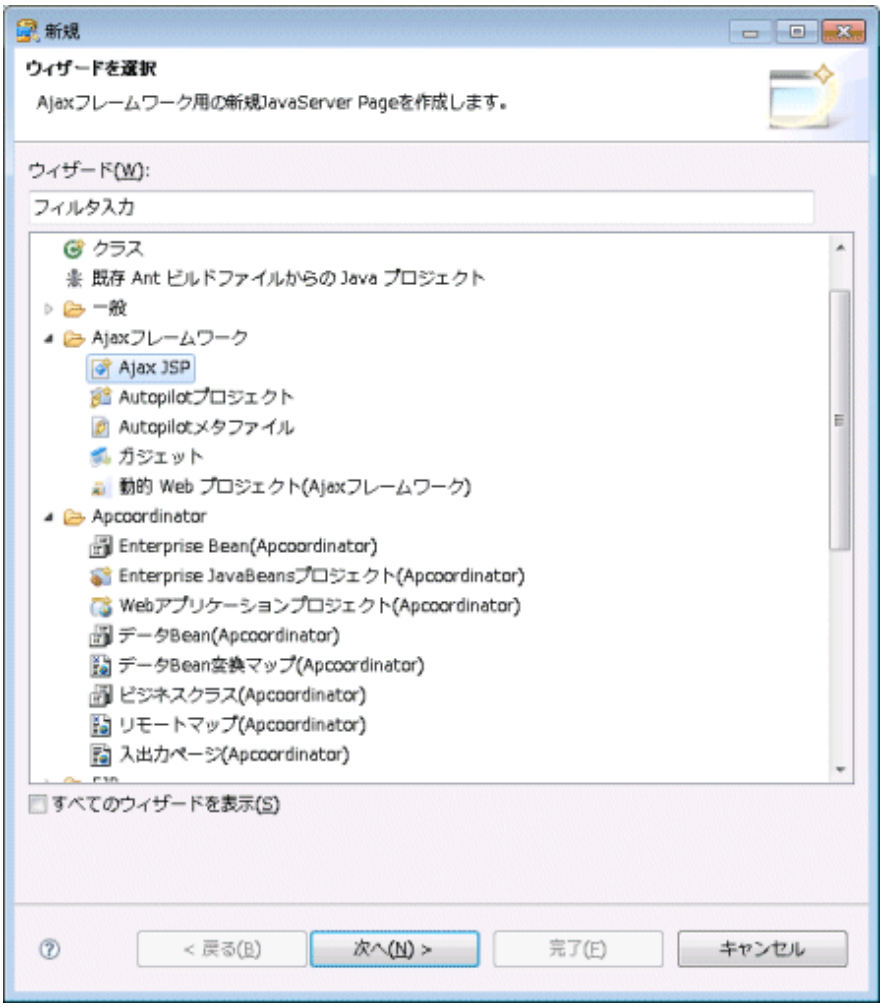
Ajax JSPウィザードで作成する場合

以下に、作成手順を示します。

1. [新規]ウィザードで、[Ajax JSP]を選択します。
2. [Ajax JavaServer Page]ウィザードで、ファイル名と作成先を指定します。

新規ウィザード

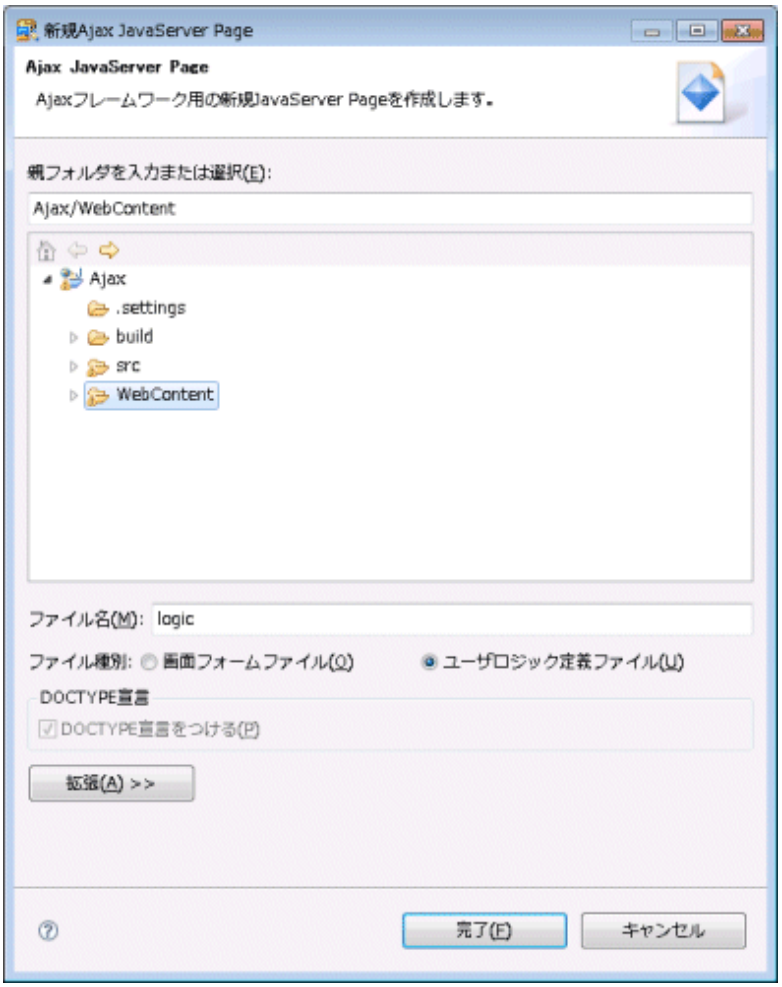
[ファイル]メニューの[新規] > [その他]で起動します。



「Ajax JSP」を選択し、[次へ]ボタンをクリックします。

Ajax JavaServer Pageウィザード

作成するJSPのファイル名と格納先を指定します。



指定する項目は、以下のとおりです。

項目	説明
親フォルダを入力または選択	作成先フォルダを入力またはリストから選択します。
ファイル名	作成するJSPのファイル名を指定します。
ファイル種別	画面フォームファイルとユーザーロジック定義ファイルのどちらを作成するかを指定します。ここでは、[ユーザーロジック定義ファイル]を選択してください。

項目を指定し、[完了]ボタンをクリックします。

HTMLウィザードまたはJSPウィザードで作成する場合

以下に、作成手順を示します。

1. [新規]ウィザードで、[HTML]または[JSP]を選択します。
2. [HTMLページ]ウィザードまたは[JavaServer Page]ウィザードで、ファイル名と作成先を指定します。
3. [HTMLテンプレートの選択]ページまたは[JSPテンプレートの選択]ページで、テンプレートを選択します。

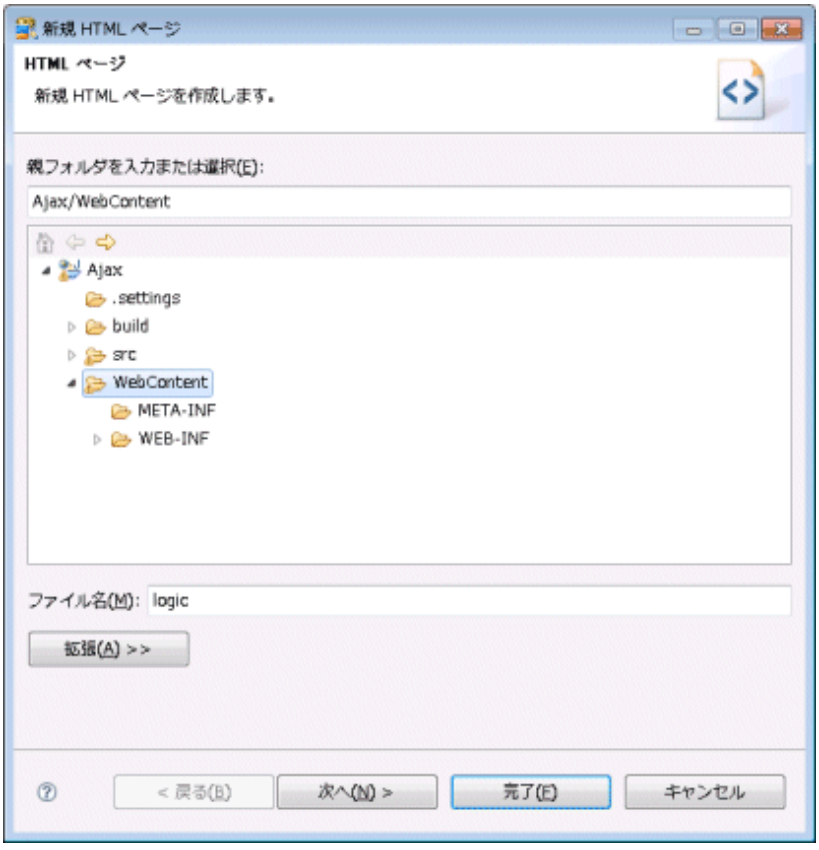
新規ウィザード

[ファイル]メニューの[新規] > [その他]で起動します。

「HTML」または「JSP」を選択し、[次へ]ボタンをクリックします。
以下では、「HTML」を選択した場合の手順を示します。

HTMLページウィザード

作成するHTMLのファイル名と格納先を指定します。



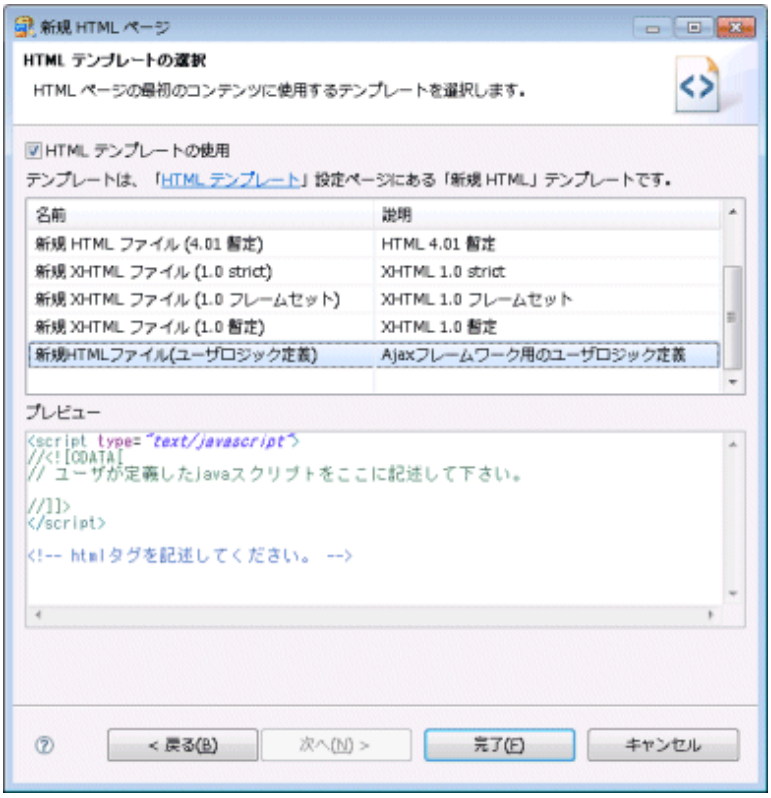
指定する項目は、以下のとおりです。

項目	説明
親フォルダを入力または選択	作成先フォルダを入力またはリストから選択します。
ファイル名	作成するHTMLのファイル名を指定します。

項目を指定し、[次へ]ボタンをクリックします。

HTMLテンプレートの選択

HTMLの作成に使用するテンプレートを選択します。



「新規HTMLファイル(ユーザーロジック定義)」を選択し、[完了]ボタンをクリックします。
なお、JSPウィザードの場合は、「新規JSPファイル(ユーザーロジック定義)」を選択します。

ここで生成されるHTMLファイルの文字コードは、Interstage Studioワークベンチで HTMLファイルに対して設定されたコード系になります。文字コードの確認および変更方法は、以下のとおりです。

1. [ウィンドウ]メニューから[設定]を選択します。
2. [設定]ダイアログボックスの設定項目で、[Web] > [HTMLファイル]を選択します。
3. HTMLファイルの作成時のエンコードを確認し、必要に応じて修正してください。

参考

HTMLページウィザードまたはJavaServer Pageウィザードで作成したファイルをAjaxページエディタで編集する場合は、Ajaxページエディタの設定が必要です。
詳細は、「F.2 Ajaxページエディタを利用した開発手順」を参照してください。

5.7.5 JavaScriptファイルの作成

JavaScriptファイルは、[新規]ウィザードで、[JavaScript] > [JavaScriptソースファイル]を選択して作成します。

詳細は、[ヘルプ]メニューの[ヘルプ目次]からヘルプを表示し、「Javascript開発ツールキットユーザーズガイド」を参照してください。

5.7.6 JavaScriptファイルの編集

JavaScriptファイルは、JavaScriptエディタで編集します。

JavaScriptエディタは、プロジェクトエクスプローラからJavaScriptファイルをダブルクリックするか、コンテキストメニューの[開く]を選択すると起動します。

以下に、JavaScriptエディタのビューを示します。

JavaScriptエディタでは、テキストの入力補完機能を利用して、JavaScriptのソースコードを簡単に入力することができます。

詳細は、「Interstage Studio ユーザーズガイド」を参照してください。

5.7.7 JavaBeanの作成

JavaBeanは、以下の手順で作成します。

1. [新規]ウィザードで、[クラス]を選択します。
2. 作成するJavaBeanの[ソースフォルダ]、[パッケージ]、[名前]を指定します。
   必要に応じて、[スーパークラス]、[インタフェース]を指定してください。

作成したJavaBeanのソースは、Javaエディタで編集します。

メンバを選択して、[ソース]メニューの[GetterおよびSetterの生成]をクリックすると、そのメンバに対してGetter/Setterを生成することができます。

5.7.8 ビジネスロジックの作成

ビジネスロジックを実装するサーブレットは、以下の手順で作成します。

1. [新規]ウィザードで、[クラス]を選択します。
2. 作成するサーブレットの[ソースフォルダ]、[パッケージ]、[名前]を指定します。
   [スーパークラス]には、「javax.servlet.http.HttpServlet」、または、HttpServletを継承したクラスを指定してください。
   必要に応じて、[インタフェース]を指定してください。

作成したサーブレットのソースは、Javaエディタで編集します。

5.7.9 Ajaxフレームワーク環境定義ファイルの作成

Ajaxフレームワークプロジェクトを作成すると、プロジェクト配下に、Ajaxフレームワーク環境定義ファイル(acf.xml)が作成されます。

作成されたAjaxフレームワーク環境定義ファイルとは別に、複数のプロジェクトで共有するAjaxフレームワーク環境定義ファイルを作成する場合には、[新規]ウィザードで、[XML] > [XML]を選択して作成します。

XMLの新規作成方法については、[ヘルプ]メニューの[ヘルプ目次]からヘルプを表示し、「Webツールプラットフォームユーザーズガイド」の「マークアップ言語ファイルの編集」を参照してください。

5.7.10 Ajaxフレームワーク環境定義ファイルの編集

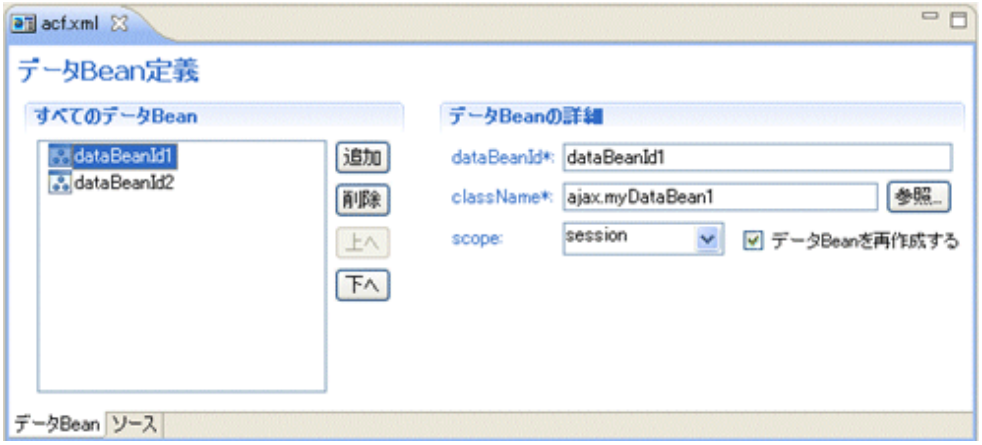
Ajaxフレームワーク環境定義ファイル(acf.xml)は、Ajaxフレームワーク環境定義ファイルエディタで編集します。

Ajaxフレームワーク環境定義ファイルエディタは、プロジェクトエクスプローラから、Ajaxフレームワーク環境定義ファイルをダブルクリックするか、コンテキストメニューの[開く]を選択すると起動します。

データBeanに関する定義

データBeanに関する定義は、[データBean]タブで編集します。

以下に、[データBean]タブで表示されるビューを示します。



指定する項目は、以下のとおりです。

項目		説明
すべてのデータBean		Ajaxフレームワーク環境定義ファイルに指定されたデータBeanの一覧が表示されます。追加、削除、順序の変更を行うことができます。 選択したデータBeanについては、詳細な情報を参照、編集できます。
データBeanの詳細		選択したデータBeanの詳細な情報を表示します。表示された値は変更することができます。
	dataBeanId	AjaxフレームワークからデータBeanを使用するときのIDを指定します。
	className	データBeanのクラス名を指定します。
	scope	データBeanのスコープを、以下の文字列から選択します。 session HTTPセション内でデータを共有する場合に指定します。 request リクエスト内でデータを共有する場合に指定します。 省略した場合は、「request」になります。
	データBeanを再作成する	セションタイムアウト発生時に、データBeanを再作成する場合はチェックします。再作成しない場合は、チェックを外します。 本チェックボックスは、[scope]で「session」を指定した場合に有効になります。

コンバータに関する定義

コンバータに関する定義は、[ソース]タブで編集します。[ソース]タブでは、データBeanの指定を編集することもできます。

以下に、[ソース]タブで表示されるビューを示します。

操作方法の詳細は、「Interstage Studioユーザーズガイド」を参照してください。

5.7.11 マッシュアップ定義ファイルの編集

マッシュアップフレームワークプロジェクトを作成すると、プロジェクト配下に、マッシュアップ定義ファイル(muf.xml)が作成されます。

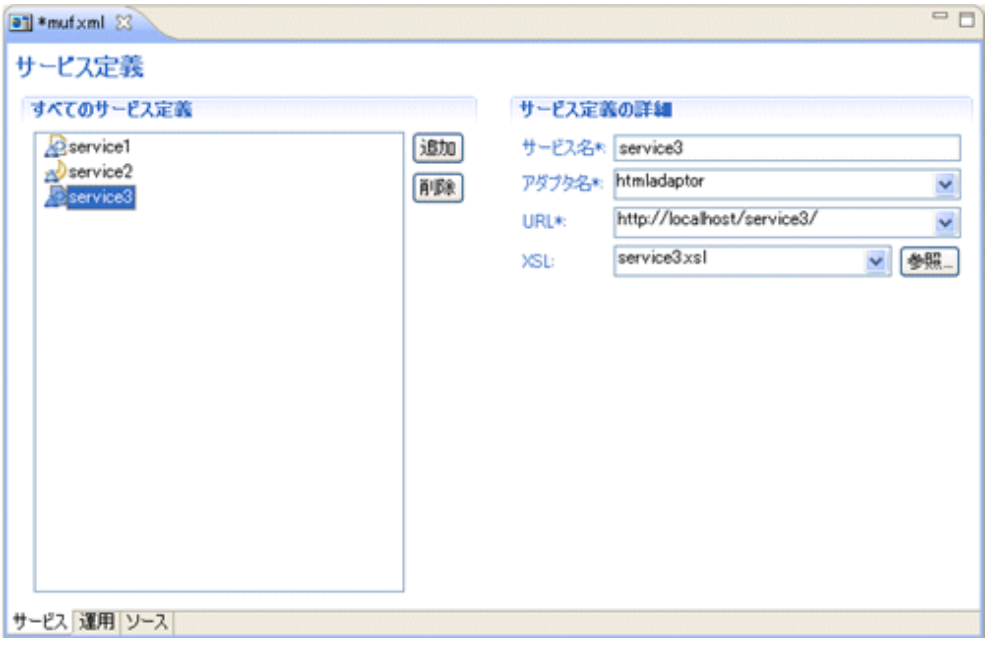
マッシュアップ定義ファイルは、マッシュアップ定義ファイルエディタで編集します。

マッシュアップ定義ファイルエディタは、プロジェクトエクスプローラから、マッシュアップ定義ファイルをダブルクリックするか、コンテキストメニューの[Open]を選択すると起動します。

サービスに関する定義

サービスに関する定義は、[サービス]タブで編集します。

以下に、[サービス]タブで表示されるビューを示します。



指定する項目は、以下のとおりです。

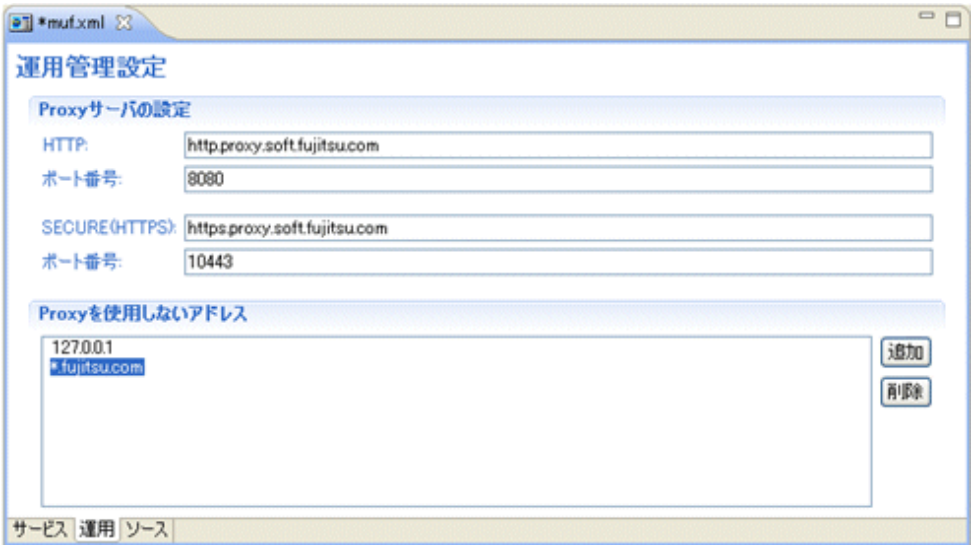
項目	説明
すべてのサービス定義	マッシュアップ定義ファイルに指定されたサービスの一覧が表示されます。サービス名の先頭には、サービスで使用するアダプタによって、以下のアイコンが表示されます。 ・ : HTMLアダプタ

項目		説明
		・  : RESTアダプタ サービスは、追加、削除することができます。 サービスは、最大128個まで定義できます。 選択したサービスについては、詳細な情報を参照、編集できます。
サービス定義の詳細		選択したサービスの詳細な情報を表示します。表示された値は変更することができます。
	サービス名	マッシュアップフレームワークからサービスを使用するときのIDを、128文字以下のシングルバイト文字(ASCII)で指定します。 サービス名は、すべてのサービスの中で一意である必要があり、省略できません。
	アダプタ名	サービスで使用するアダプタ名を、以下のどちらかから選択します。 ・ htmladaptor: HTMLアダプタ ・ restadaptor: RESTアダプタ
	URL	接続先のアドレスを、シングルバイト文字(ASCII)で指定します。 [アダプタ名]に「htmladaptor」を選択した場合は、選択リストからURLを選択することができます。選択リストには、プロジェクト内のXSLファイルに対応するURLが表示されます。選択したURLは、直接編集することができます。 なお、開発資産を移行した場合、選択リストにURLが表示されないことがあります。その場合は、直接入力してください。 URLは、省略できません。
	XSL	コンテンツに適用するXSLファイル名を、シングルバイト文字(ASCII)で指定します。 [アダプタ名]に「htmladaptor」を選択した場合は、選択リストからXSLファイル名を選択することができます。選択リストには、[URL]で指定した値に対応するXSLファイル名が表示されます。選択したXSLファイル名は、直接編集することができます。 なお、開発資産を移行した場合、選択リストにファイル名が表示されないことがあります。その場合は、直接入力してください。 XSLは省略できます。

運用に関する定義

プロキシサーブレットのプロキシ設定など、運用に関する定義は、[運用]タブで編集します。

以下に、[運用]タブで表示されるビューを示します。



指定する項目は、以下のとおりです。

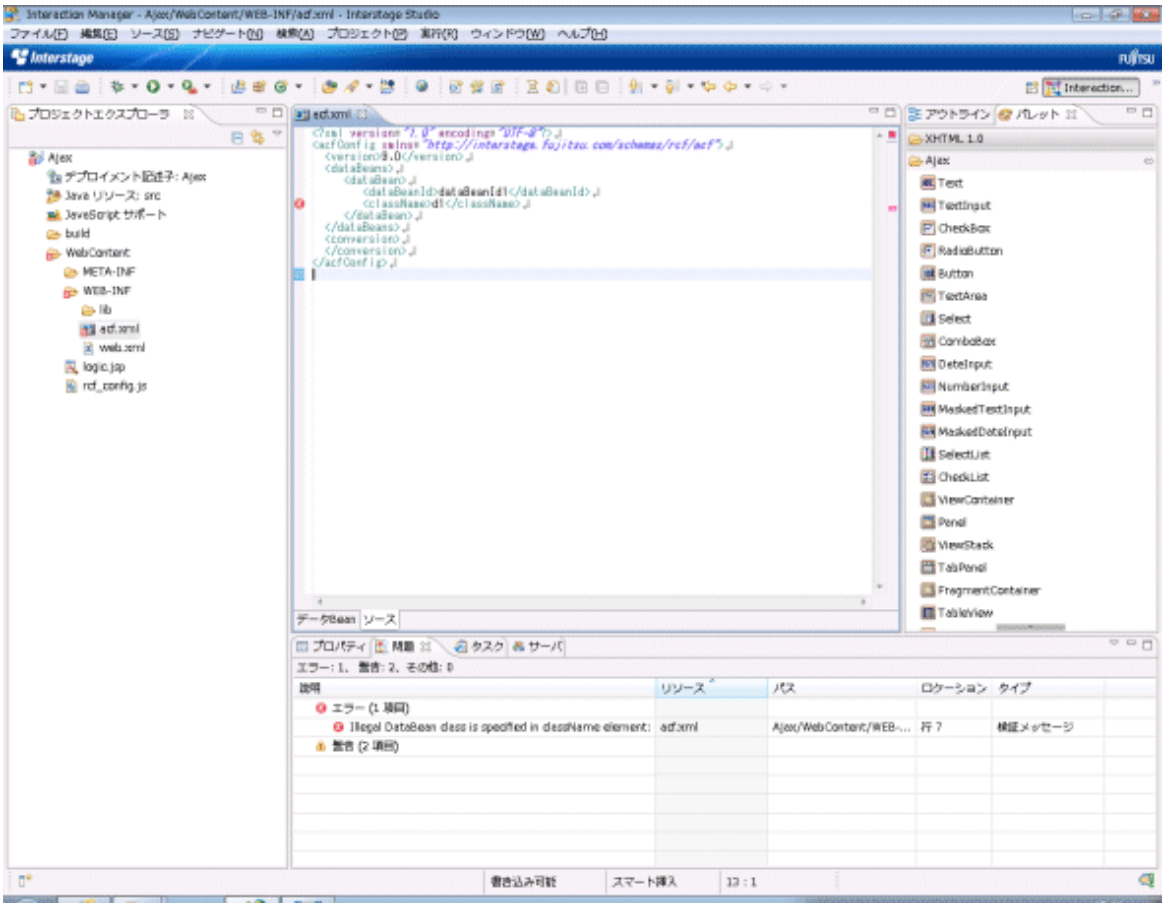
項目		説明
Proxyサーバの設定		マッシュアップフレームワークで使用するプロキシの情報を表示します。 表示された値は変更することができます。
	HTTP	HTTP通信で使用するプロキシサーバのアドレスを、128文字以下のシングルバイト文字(ASCII)で指定します。 HTTPは省略できます。
	ポート番号	HTTP通信で使用するプロキシサーバのポート番号を、0～65535の数値で指定します。 HTTPを指定した場合は、省略できません。
	SECURE(HTTPS)	HTTPS通信で使用するプロキシサーバのアドレスを、128文字以下のシングルバイト文字(ASCII)で指定します。 SECURE(HTTPS)は省略できます。
	ポート番号	HTTPS通信で使用するプロキシサーバのポート番号を、0～65535の数値で指定します。 SECURE(HTTPS)を指定した場合は、省略できません。
Proxyを使用しないアドレス		プロキシサーバを使用しないアドレスの一覧が表示されます。アドレスは、追加、削除することができます。 アドレスは、最大128個まで定義できます。

なお、サービスに関する定義と運用に関する定義は、[ソース]タブで編集することもできます。
操作方法の詳細は、[Help]メニューの[Help Contents]でヘルプを表示し、「Web Tools Platform User Guide」の「Editing markup language files」を参照してください。

5.7.12 検証

Ajaxフレームワーク環境定義ファイルバリデータでは、Ajaxフレームワーク環境定義ファイルの設定に誤りがないかを検証します。このバリデータは、Ajaxフレームワークのプロジェクトではデフォルトで有効となり、随時、検出したエラーを[プロジェクトエクスプローラ]、[ソース]ビュー、[問題]ビューに表示します。

以下に、表示例を示します。



問題が検出された場合のエラーメッセージについては、「J.3.2 環境定義ファイルに関するメッセージ」を参照してください。

5.7.13 ビルド

Interstage Studio ワークベンチを利用してAjaxフレームワークアプリケーションを開発する場合の一般的な開発資産を、以下に示します。

- デプロイメント記述子
  プロジェクトで開発したアプリケーションの配備情報を指定します。
- ソースフォルダ
  プロジェクトで開発したJavaソースを、パッケージの階層に従って格納します。デフォルトは、プロジェクト配下のsrcフォルダが指定されます。
- Javaのビルドパスに指定されたライブラリ
  Javaのビルド時に使用するライブラリです。デフォルトは、以下のライブラリが指定されます。
  - JREシステムライブラリ
  - Ajaxフレームワーク
- WebContentフォルダ
  アプリケーションのwarファイルにアーカイブするファイルを格納するフォルダです。コンパイルされたソースファイルは、build/classesフォルダに、パッケージ階層に従って格納されます。

以下に、ビルドの方法を示します。

- [プロジェクト]メニューの[自動的にビルド]にチェックが入っていない場合は、[プロジェクトのビルド]を選択します。
  チェックが入っている場合は、プロジェクト作成後に自動的にビルドされるため、操作は不要です。
- srcフォルダ配下のjavaソースは、ビルドによってコンパイルされ、パッケージ階層に従って、build/classesに格納されます。
  運用では、必要なファイルをwarファイルにまとめて配備してください。

5.7.14 実行・デバッグ

IJServer上で実行する場合は、プロジェクトのコンテキストメニューの[実行] > [サーバで実行]を選択してください。デバッグする場合は、[デバッグ] > [サーバで実行]を選択してください。
なお、実行環境の設定については、「5.6 実行環境の設定」を参照してください。

Ajaxフレームワーク固有のデバッグ機能については、「2.6.1 デバッグ機能」を参照してください。

参考

Java EE 6のアプリケーションを開発し、Interstage Studio上で実行する場合は、「5.6.2 実行時に必要なファイル」を参照して、[Java EE 6共通フォルダ]¥domains¥domain1¥libにライブラリをコピーして、Interstage Studioを再起動してください。
例) C:¥Interstage¥APS¥F3FMisje6¥var¥domains¥domain1¥lib

5.7.15 Apcoordinator連携アプリケーションの開発

Apcoordinatorと連携するAjaxフレームワークアプリケーションを開発する場合は、以下の手順が必要になります。

Apcoordinatorファセットの選択

「5.7.1 Ajaxフレームワークプロジェクトの作成」で、[動的Webプロジェクト]ページの[構成]の[変更]を選択し、[Apcoordinator Core]ファセットをチェックします。

プロジェクトの作成ウィザードに以下のページが追加されます。以下に記載した操作を行ってください。

- [テンプレートを選択]ページ
  Ajaxフレームワークアプリケーションのテンプレートを選択します。
- [制御ページ情報]ページ
  制御ページの情報を設定します。制御ページ情報の詳細は、「Apcoordinatorユーザーズガイド」を参照してください。
- [ファクトリ拡張情報]ページ
  ファクトリの拡張情報を設定します。ファクトリ拡張情報の詳細は、「Apcoordinatorユーザーズガイド」を参照してください。

データBeanの作成

「5.7.7 JavaBeanの作成」で、JavaBeanの代わりにApcoordinatorのデータBeanを作成します。データBeanの作成の詳細はInterstage Application Serverの「Apcoordinatorユーザーズガイド」を参照してください。

ビジネスクラスの作成

「5.7.8 ビジネスロジックの作成」で、サーブレットの代わりにApcoordinatorのビジネスクラスを作成します。ビジネスクラスの作成の詳細はInterstage Application Serverの「Apcoordinator入門ガイド」、および「Apcoordinatorユーザーズガイド」を参照してください。

5.8 Eclipseを利用した開発

Eclipseを利用して、サーブレットと連携するAjaxフレームワークアプリケーションを開発する場合、「Interstage Studio ワークベンチを利用した開発」の「Ajaxフレームワークプロジェクトの作成」～「ビルド」まで同じ手順です。
「5.7 Interstage Studioワークベンチを利用した開発」の手順を参考に、開発を行ってください。なお、日本語のメニュー名やコマンド名などは、英語の名称に読み替えてください。

ここでは、Eclipseを利用した実行とデバッグの方法について説明します。

注意

プロジェクトウィザードで作成したAjaxフレームワークプロジェクトをビルドするためには、ターゲットランタイムを選択するか、サーバランタイムのライブラリをクラスパスに追加する必要があります。サーバランタイムのライブラリの詳細は、使用するアプリケーションサーバに添付されているドキュメントを参照してください。

5.8.1 実行・デバッグ(Eclipse)

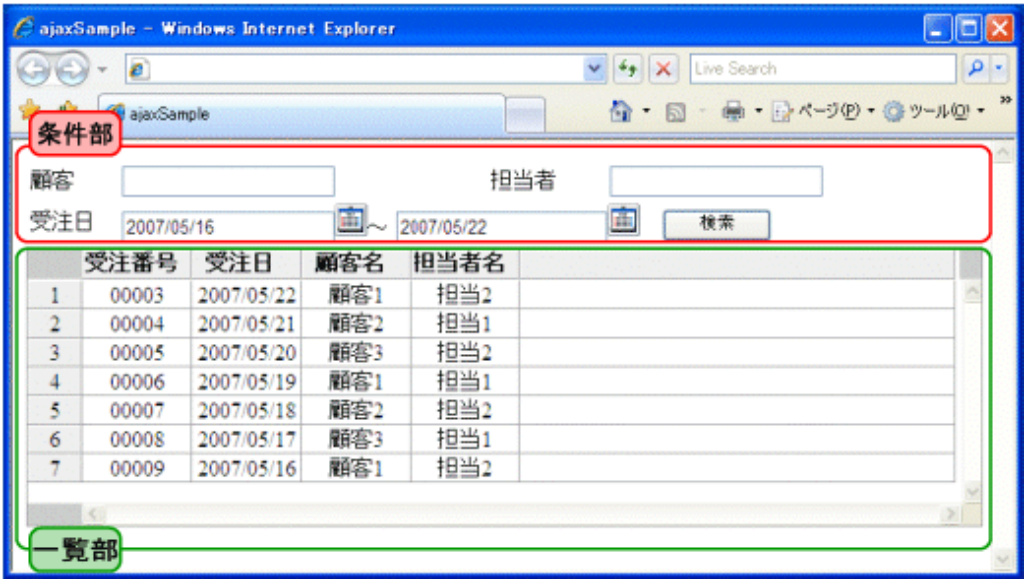
実行する場合は、プロジェクトのコンテキストメニューの[Run As] > [Run on Server]を選択してください。デバッグする場合は、[Debug As] > [Run on Server]を選択してください。
なお、実行環境の設定については、「5.6 実行環境の設定」を参照してください。

Ajaxフレームワーク固有のデバッグ機能については、「2.6.1 デバッグ機能」を参照してください。

5.9 アプリケーションの開発例

ここでは、入力条件に従って検索結果をテーブルに表示するAjaxフレームワークアプリケーションを例に、Interstage Studioワークベンチを利用した場合の開発手順を説明します。

画面イメージ



画面概要

画面項目
以下の()内は、UI部品名を示します。

- 条件部
  - 顧客: テキスト入力 (TextInput)
  - 担当者: テキスト入力 (TextInput)
  - 受注日(開始日): 日付入力 (DateInput)
  - カレンダボタン (CalendarButton、PopupCalendar)
  - 受注日(終了日): 日付入力 (DateInput)
  - カレンダボタン (CalendarButton、PopupCalendar)
  - [検索]ボタン (Button)
- 一覧部 (TableView)
  - 受注番号: テキスト (ViewColumn)
  - 受注日: テキスト (ViewColumn)
  - 顧客名: テキスト (ViewColumn)
  - 担当者名: テキスト (ViewColumn)

動作仕様

- カレンダボタン
  ボタンをクリックすると、カレンダを表示する。
  カレンダから日付を選択すると、入力欄に選択された日付を設定する。
- [検索]ボタン
  条件に応じた検索を行い、一覧部の内容を再表示する。

開発手順

1. Ajaxフレームワークプロジェクトの作成
   Interstage Studioワークベンチを利用して、Ajaxフレームワークアプリケーションを開発するためのプロジェクトを作成します。詳細は、「5.9.1 プロジェクトの作成(開発例)」を参照してください。
2. 画面フォーム(ひな形)の作成
   Interstage Studioワークベンチを利用して、画面フォームのひな形(JSPファイル)を作成します。詳細は、「5.9.2 画面フォーム(ひな形)の作成(開発例)」を参照してください。
3. 画面フォームの作成
   Interstage Studioワークベンチを利用して、画面フォームのひな形(JSPファイル)に以下の定義を記述し、画面フォームを作成します。
   - ユーザーデータの定義
     詳細は、「5.9.3 ユーザーデータの定義(開発例)」を参照してください。
   - 画面部品の定義
     詳細は、「5.9.4 画面部品の定義(開発例)」を参照してください。
4. データBeanの作成
   Apcoordinator用のデータBeanを作成します。詳細は、「5.9.5 データBeanの作成(開発例)」を参照してください。
5. ビジネスクラスの作成・コマンドマップの定義
   Apcoordinator用のビジネスクラスを作成し、コマンドマップを定義します。詳細は、「5.9.6 ビジネスクラスの作成(開発例)」を参照してください。
6. Ajaxフレームワーク環境定義ファイルの設定
   Interstage Studioワークベンチを利用して、Ajaxフレームワーク環境定義ファイル(acf.xml)に必要な定義を記述します。詳細は、「5.9.7 Ajaxフレームワーク環境定義ファイルの設定(開発例)」を参照してください。
7. 動作定義
   Interstage Studioワークベンチを利用して、JSPファイルに動作を実装するための定義を記述します。詳細は、「5.9.8 動作定義(開発例)」を参照してください。

ポイント

この開発例では、使用するJavaScriptを、外部ファイルではなく、画面フォーム(JSPファイル)に記述します。

5.9.1 プロジェクトの作成(開発例)

Interstage Studioワークベンチを利用して、Ajaxフレームワークアプリケーションを開発するためのプロジェクトを作成します。
Interstage Studioワークベンチの操作方法については、「5.7.1 Ajaxフレームワークプロジェクトの作成」を参照してください。

プロジェクト作成時には、以下の情報を入力します。

- プロジェクト名: ajaxSample
  プロジェクト名以外の項目は、すべてデフォルトの内容でプロジェクトを作成します。

プロジェクト作成完了後、main.jspに記載されている以下の記述は、本アプリケーションでは不要なため削除します。

削除内容

```
<uji:include pane="head" />
```

Interstage Studioワークベンチを利用してプロジェクトを作成すると、自動的にアプリケーションクラス(AjaxSampleApplication.java)が作成されます。
作成されたアプリケーションクラスに、アプリケーション共通のデータを定義します。

import宣言部分に、以下の記述を追加します。

記述内容

```
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
```

以下の宣言を追加します。

記述内容

```
public ArrayList listData;
```

コンストラクタを以下のように変更します。

記述内容

```
public AjaxSampleApplication(){
     /* 一覧表の初期データを設定します。 */

     listData = new ArrayList();
     Date currentDate = new Date();
     Calendar currentCal = Calendar.getInstance();
     currentCal.setTime(currentDate);
     for (int i = 1 ; i < 11 ; i++) {
          RecordBean orderBean = new RecordBean();
          orderBean.setOrderNo("0000" + i);
          Calendar calendar = Calendar.getInstance();
          calendar.clear();
          calendar.set(Calendar.YEAR, currentCal.get(Calendar.YEAR));
          calendar.set(Calendar.MONTH, currentCal.get(Calendar.MONTH));
          calendar.set(Calendar.DATE, currentCal.get(Calendar.DATE) - i);

          DateFormat df = new SimpleDateFormat("yyyy/MM/dd");
          orderBean.setOrderDate(df.format(calendar.getTime()));
          orderBean.setCustomer("顧客" + (i%3+1));
          orderBean.setCharge("担当" + (i%2+1));
          listData.add(orderBean);
     }
}
```

以下のメソッドを追加します。

記述内容

```
public boolean isXhtmlMode(){
     return true;
}

public ArrayList getListData() {
     return listData;
}
```

なお、アプリケーションクラスの詳細は、Interstage Application Serverの「Apcoordinator ユーザーズガイド」を参照してください。

5.9.2 画面フォーム(ひな形)の作成(開発例)

Interstage Studioワークベンチを利用して、画面フォームのひな形(JSPファイル)を作成します。
Interstage Studioワークベンチの操作方法については、「5.7.2 画面フォーム(ひな形)の作成」を参照してください。

画面フォームのひな形を作成するには、Ajax JSPウィザードを使用します。
画面フォームのひな形作成時には、以下の情報を入力します。

- ファイル名: search.jsp

5.9.3 ユーザーデータの定義(開発例)

「5.9.2 画面フォーム(ひな形)の作成(開発例)」で作成した画面フォームのひな形に、Interstage StudioワークベンチのJSPエディタを利用して、ユーザーデータの定義を記述します。
Interstage Studioワークベンチの操作方法については、「5.7.3 画面フォームの編集」を参照してください。

以下のように、画面項目のブロック単位にデータ格納域を定義します。

- 条件部: conditionData
- 一覧部: listData

ユーザーデータ定義の記述内容

```
// 条件部
  conditionData = {
    customer:'',            ←項目名:値 の形式で表記
    charge:'',
    fromDate:null,
    toDate:null
  };

// 行データ
  rowData = [];             ←リスト構造のデータは配列で定義

// 一覧部
  listData = {
    list:rowData
  };
```

5.9.4 画面部品の定義(開発例)

「5.9.3 ユーザーデータの定義(開発例)」でユーザーデータを定義した画面フォームに、Interstage StudioワークベンチのJSPエディタを利用して、画面部品の定義を記述します。
Interstage Studioワークベンチの操作方法については、「5.7.3 画面フォームの編集」を参照してください。

画面部品の定義では、ユーザーデータをモデルとして利用するための機能定義と、UI部品を使用するための定義を記述します。

条件部の機能定義とUI部品定義

機能定義の記述内容

```
<!-- 条件部のユーザーデータをモデルとして利用するための機能定義 -->
<div rcf:type="Model" rcf:id="conditionModel" rcf:object="conditionData"></div>
```

UI部品定義の記述内容

```
<!-- 条件部のUI部品定義とモデルバインディング -->
<table width="100%">
 <tbody>
  <tr>
   <td>顧客</td>
   <!-- 顧客名入力 テキスト入力フィールド定義 -->
   <td>
```

```
    <span rcf:type="TextInput" rcf:id="customer" rcf:value="{conditionModel.customer}"></span>  (1)
   </td>
   <td>担当者</td>
   <!-- 担当者名入力 テキスト入力フィールド定義 -->
   <td>
    <span rcf:type="TextInput" rcf:id="charge" rcf:value="{conditionModel.charge}"></span>      (2)
   </td>
  </tr>
  <tr>
   <td>受注日</td>
   <td colspan="3">
    <!-- 受注日(開始日)入力 日付入力フィールド定義 -->
    <span rcf:type="DateInput" rcf:id="fromDate" rcf:date="{conditionModel.fromDate}"></span>   (3)
    <!-- 受注日(開始日)入力 カレンダ定義 -->
    <div rcf:type="PopupCalendar"
     rcf:id="fromDateCalendar" rcf:selectedDate="{conditionModel.fromDate}"></div>              (4)
    <!-- 受注日(開始日)入力 カレンダ呼出ボタン定義 -->
    <span rcf:type="CalendarButton" rcf:target="fromDateCalendar"></span>                      (5)
     ～
    <!-- 受注日(終了日)入力 日付入力フィールド定義 -->
    <span rcf:type="DateInput" rcf:id="toDate" rcf:date="{conditionModel.toDate}"></span>      (6)
    <!-- 受注日(終了日)入力 カレンダ定義 -->
    <div rcf:type="PopupCalendar"
     rcf:id="toDateCalendar" rcf:selectedDate="{conditionModel.toDate}"></div>                 (4)
    <!-- 受注日(終了日)入力 カレンダ呼出ボタン定義 -->
    <span rcf:type="CalendarButton" rcf:target="toDateCalendar"></span>                        (5)
   </td>
  </tr>
 </tbody>
</table>
```

1) モデルオブジェクトconditionModelのcustomerプロパティをTextInputの値にバインディング
2) モデルオブジェクトconditionModelのchargeプロパティをTextInputの値にバインディング
3) モデルオブジェクトconditionModelのfromDateプロパティをDateInputの値にバインディング
4) カレンダで選択した日付を反映する入力フィールドを、バインディング式を介して関連付け
5) カレンダボタンで表示するカレンダを関連付け
6) モデルオブジェクトconditionModelのtoDateプロパティをDateInputの値にバインディング

一覧部の機能定義とUI部品定義

機能定義の記述内容

```
<!-- 一覧部のユーザーデータをモデルとして利用するための機能定義 -->
<div rcf:type="Model" rcf:id="listModel" rcf:object="listData"></div>
```

UI部品定義の記述内容

```
<!-- 一覧部のUI部品定義とモデルバインディング -->
<div rcf:id="list" rcf:type="TableView" rcf:data="{listModel.list}"            (1)
  rcf:showRowHeader="true" rcf:width="700px" rcf:height="200px"
  rcf:multipleSelect="false">
    <div rcf:type="ViewColumn" rcf:name="orderNo" rcf:label="受注番号"></div>   (2)
    <div rcf:type="ViewColumn" rcf:name="orderDate" rcf:label="受注日"></div>   (2)
    <div rcf:type="ViewColumn" rcf:name="customer" rcf:label="顧客名"></div>    (2)
    <div rcf:type="ViewColumn" rcf:name="charge" rcf:label="担当者名"></div>    (2)
</div>
```

1) モデルオブジェクトlistModelのlistプロパティをTableViewの値にバインディング
2) 列情報(1行分のデータ項目)を定義

5.9.5 データBeanの作成(開発例)

Interstage Studioワークベンチを利用して、データBeanを作成します。
詳細については、「Apcoordinator入門ガイド」、および「Apcoordinatorユーザーズガイド」を参照してください。

条件部および一覧部のデータをApcoordinatorに受け渡すために、以下のデータBeanを作成します。

- 条件用データBean: ConditionBean
- 一覧用データBean: ListBean
- 一覧レコード用データBean: RecordBean

条件用データBean作成時には、以下の情報を入力します。

- パッケージ名: ajaxSample
- クラス名: ConditionBean
- プロパティには、以下の内容を追加します。

項目名	プロパティ名	型
顧客	customer	String
担当	charge	String
受注日検索(開始)	fromDate	java.util.Date
受注日検索(終了)	toDate	java.util.Date

一覧用データBean作成時には、以下の情報を入力します。

- パッケージ名: ajaxSample
- クラス名: ListBean
- プロパティには、以下の内容を追加します。

項目名	プロパティ名	型
一覧用データ	list	java.util.Vector

一覧レコード用データBean作成時には、以下の情報を入力します。

- パッケージ名: ajaxSample
- クラス名: RecordBean
- プロパティには、以下の内容を追加します。

項目名	プロパティ名	型
予約番号	orderNo	String
予約日時	orderDate	String
顧客	customer	String
担当	charge	String

検索結果の1行分をRecordBeanに格納し、ListBeanのlistの要素とすることで、一覧構造を定義します。

5.9.6 ビジネスクラスの作成(開発例)

Interstage Studioワークベンチを利用して、ビジネスクラスを作成します。
詳細については、「Apcoordinator入門ガイド」、および「Apcoordinatorユーザーズガイド」を参照してください。

ビジネスクラスの作成

ビジネスクラス作成時には、以下の情報を入力します。

- パッケージ名: ajaxSample
- クラス名: AjaxSampleHandler
- メソッドには、以下の内容を追加します。

入力データ	入力情報	コマンド	メソッド	復帰値型
入力データなし	指定しない	指定しない	startup	void
データBean	クラス名: ConditionBean	search	search	object

作成されたビジネスクラスに、以下の記述を追加します。

import宣言部分の記述内容

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Locale;
import java.util.TimeZone;
import java.util.Vector;
```

startupメソッド部分の記述内容

```
/* 初期化処理 */
public void startup(DispatchContext context) {
  ConditionBean conditionBean = new ConditionBean();
  context.setResponseBean("body", conditionBean);
}
```

searchメソッド部分の記述内容

```
/* 検索処理 */
public Object search(DispatchContext context, ajaxSample.ConditionBean dataBean) {
  ListBean listBean = new ListBean();
  AjaxSampleApplication application = (AjaxSampleApplication)context.getApplicationProfile();
  ArrayList listData = application.getListData();

  Vector v = new Vector();
  for (int i = 0 ; i <listData.size() ; i++) {
    RecordBean orderBean = (RecordBean)listData.get(i);
    if (matches(dataBean, orderBean)) {
      v.add(orderBean);
    }
  }
  listBean.setList(v);
  return listBean;
}
```

matchesメソッド(注)の記述内容

```
/* 検索処理サブルーチン */
private boolean matches(ajaxSample.ConditionBean dataBean, RecordBean orderBean) {
 String customer = dataBean.getCustomer();
 if (customer != null && customer.length()!=0 && !customer.equals((String)orderBean.getCustomer())) {
   return false;
 }
 String charge = dataBean.getCharge();
 if (charge != null && charge.length()!=0 && !charge.equals((String)orderBean.getCharge())) {
   return false;
 }

 Date orderDate = null;
 SimpleDateFormat formatter = new SimpleDateFormat("yyyy/MM/dd",Locale.JAPAN);
```

```
 formatter.setTimeZone(TimeZone.getDefault());
 try {
   orderDate = formatter.parse(orderBean.getOrderDate());
 } catch(ParseException e) {
   e.printStackTrace();
 }

 Date fromDate = dataBean.getFromDate();
 if (fromDate != null) {
   if (orderDate.before(fromDate)) {
     return false;
   }
 }

 Date toDate = dataBean.getToDate();
 if (toDate!= null) {
   if (orderDate.after(toDate)) {
     return false;
   }
 }
 return true;
}
```

注) このメソッドは自動生成されません。

コマンドマップの定義

Interstage Studioワークベンチを利用してビジネスクラスを作成すると、自動的にコマンドマップ(commands.map)が作成されます。

コマンドマップの記述内容

```
# 初期化処理
;=ajaxSample.AjaxSampleHandler.startup
# 検索処理実行
ajaxSample.ConditionBean;search=ajaxSample.AjaxSampleHandler.search
```

コマンドマップの詳細は、「Apcoordinator ユーザーズガイド」を参照してください。

ページマップの定義

ページマップ(pages.map)に必要な定義を追加します。

ページマップの記述内容

```
# pages.map
ajaxSample.ConditionBean;=search.jsp
```

ページマップの詳細は、「Apcoordinator ユーザーズガイド」を参照してください。

5.9.7 Ajaxフレームワーク環境定義ファイルの設定(開発例)

Interstage Studioワークベンチを利用して、Ajaxフレームワーク環境定義ファイル(acf.xml)に必要な定義を記述します。
Interstage Studioワークベンチの操作方法については、「5.7.9 Ajaxフレームワーク環境定義ファイルの作成」を参照してください。

データBeanの定義

Apcoordinator用に作成したデータBeanを、それぞれ、定義します。

acf.xmlの記述内容

```
<!-- データBean定義 -->
  <dataBeans>
  <!-- 検索条件 -->
    <dataBean>
      <dataBeanId>searchCondition</dataBeanId>
```

```
    <className>ajaxSample.ConditionBean</className>
    <scope>request</scope>
  </dataBean>
 <!-- 一覧 -->
  <dataBean>
    <dataBeanId>searchList</dataBeanId>
    <className>ajaxSample.ListBean</className>
    <scope>request</scope>
  </dataBean>
 <!-- 一覧レコード -->
  <dataBean>
    <dataBeanId>searchRecord</dataBeanId>
    <className>ajaxSample.RecordBean</className>
    <scope>request</scope>
  </dataBean>
 </dataBeans>
```

コンバータ設定の定義

JavaScriptオブジェクトとJavaオブジェクト(データBean)のデータ型変換を行うために、コンバータおよびコンバータを適用するルールを定義します。

acf.xmlの記述内容

```
<conversion>
 <!-- コンバータの定義 -->
 <!-- 検索条件 -->
  <defConverter>
    <converterId>conditionBeanConverter</converterId>
    <beanConverter>
      <concreteType>ajaxSample.ConditionBean</concreteType>
    </beanConverter>
  </defConverter>
 <!-- 一覧レコード-->
  <defConverter>
    <converterId>recordBeanConverter</converterId>
    <beanConverter>
      <concreteType>ajaxSample.RecordBean</concreteType>
    </beanConverter>
  </defConverter>
 <!-- 一覧 -->
  <defConverter>
    <converterId>listBeanConverter</converterId>
    <beanConverter>
      <concreteType>ajaxSample.ListBean</concreteType>
      <child>
        <property>list</property>
        <collectionConverter>
          <concreteType>java.util.Vector</concreteType>
          <child>
            <property>element</property>
            <type>ajaxSample.RecordBean</type>
          </child>
        </collectionConverter>
      </child>
    </beanConverter>
  </defConverter>

<!-- コンバージョンルール定義 -->
 <!-- 検索条件 -->
  <conversionRule>
    <type>ajaxSample.ConditionBean</type>
    <converterId>conditionBeanConverter</converterId>
```

```
    </conversionRule>
  <!-- 一覧レコード -->
    <conversionRule>
      <type>ajaxSample.RecordBean</type>
      <converterId>recordBeanConverter</converterId>
    </conversionRule>
  <!-- 一覧 -->
    <conversionRule>
      <type>ajaxSample.ListBean</type>
      <converterId>listBeanConverter</converterId>
    </conversionRule>
</conversion>
```

5.9.8 動作定義(開発例)

「5.9.4 画面部品の定義(開発例)」で画面部品を定義した画面フォームに、Interstage Studioワークベンチの JSPエディタを利用して、検索処理に関する動作定義を記述します。
Interstage Studioワークベンチの操作方法については、「5.7.3 画面フォームの編集」を参照してください。

動作定義では、検索ボタンの定義と、イベントの定義・登録、検索処理の定義を記述します。

検索ボタンの定義

画面の[検索]ボタンを定義します。

記述内容

```
<!-- 検索ボタン定義 -->
<div rcf:type="Button" rcf:id="searchButton" rcf:width="80px">検索</div>
```

イベントの定義

作成したボタンにイベントを定義します。

記述内容

```
// イベント定義
sampleEvent = {
  searchButton:{                (1)
    click:search                (2)
  }
};
```

1) 作成したボタンを指定
2) クリック時にsearch関数を実行するように定義

イベントの登録

初期処理の登録APIを利用して、定義したイベントを登録します。

記述内容

```
// 初期処理定義
RCF.addInitializedListener(
  function(eventObject) {
    // イベント登録
    rcf.event.EventRegistrar.registerEvents(sampleEvent, "sampleEvent");    (1)
  }
);
```

1) 定義したイベントに「sampleEvent」という名前を付けて登録

検索処理の定義

イベントの定義で記述した関数を定義し、Apcoordinatorのビジネスクラスを呼び出す関数を記述します。

記述内容

```
// 検索処理定義
function search(){
  // requestParam
  var reqParam = {
    beanId:'searchCondition',                    (1)
    verb:'search'                                (2)
  };
  // option
  var option = {
    url:'acf/apc',
    callback:function(res){                      (3)
      if(res.list != null){
        listModel.setProperty("list",res.list);
      }
      else {
        listModel.setProperty("list",[]);
      }
    }
  };
  UjiRequest.send(conditionData,reqParam,option);      (4)
}
```

1) 送信データを格納するデータBean(Ajaxフレームワーク環境定義ファイルに定義したデータBean ID)を指定
2) 呼び出すメソッドに対応するコマンド名(コマンドマップに定義したコマンド名)を指定
3) 通信完了時のコールバック処理を定義
4) 送信するデータオブジェクト、リクエストパラメーターオブジェクト、通信設定オブジェクトを指定して、通信実行

5.10 アプリケーションの開発例(マッシュアップ)

ここでは、Webサービスへの認証後、私用カレンダをスクレイピングツールで取得して表示するアプリケーションを例に、マッシュアップフレームワークおよびInterstage Studioワークベンチを利用した開発手順を説明します。

画面イメージ

図5.4 初期表示時/ログアウト後の画面

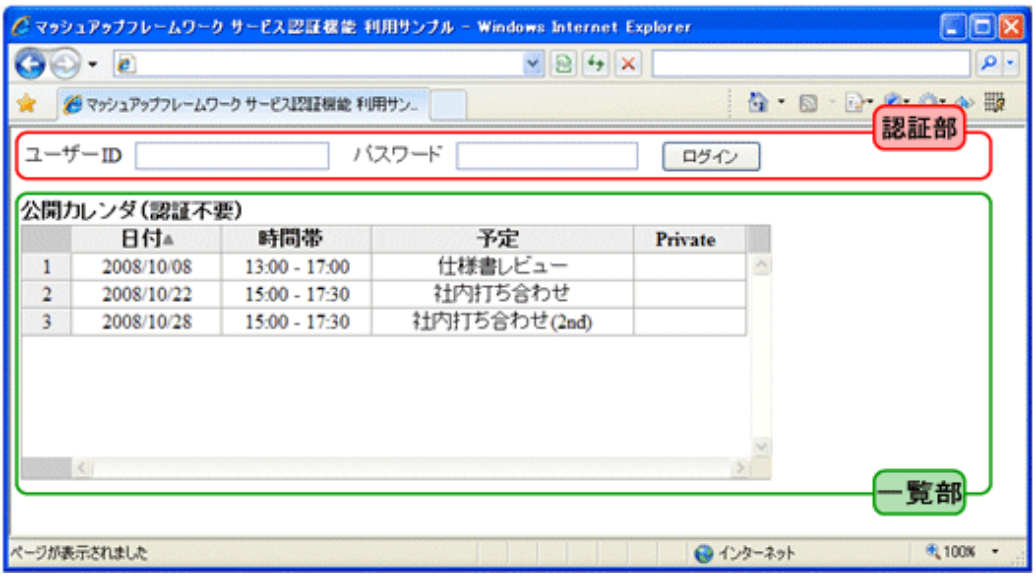
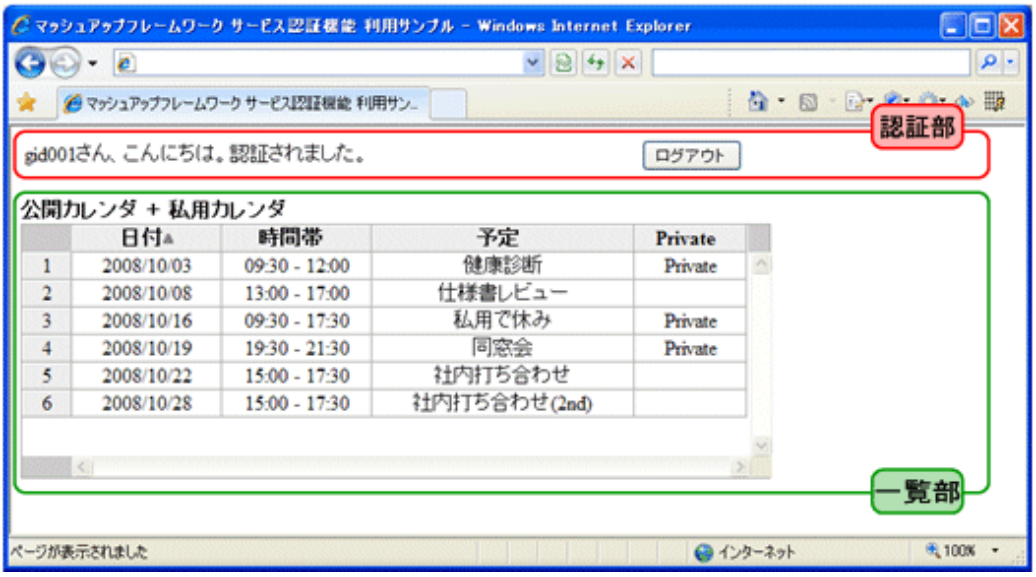


図5.5 ログイン後の画面



画面概要

画面項目

以下の()内は、UI部品名を示します。

- 認証部 (ViewStack)
  - 初期表示時/ログアウト後 (ViewContainer)
    - ユーザーID: テキスト入力 (TextInput)
    - パスワード: パスワード入力 (TextInput)
    - [ログイン]ボタン (Button)
  - ログイン後 (ViewContainer)
    - ユーザーID: テキスト (Text)
    - [ログアウト] ボタン (Button)
- 一覧部 (TableView)
  - 日付: テキスト (ViewColumn)
  - 時間帯: テキスト (ViewColumn)
  - 予定: テキスト (ViewColumn)
  - Private: テキスト (ViewColumn)

動作仕様

- 初期表示時
  認証部には、空の入力フィールドと[ログイン]ボタンを表示する。
  一覧部には、公開カレンダ(認証不要)を表示する。
- [ログイン]ボタン
  ボタンをクリックすると、入力されたユーザー情報に基づいて、認証情報データベース内の情報と照合して基本認証処理を行う。
  (ユーザー情報については、「5.10.9 ビジネスクラスの作成(マッシュアップ開発例)」の「ログイン情報および認証情報データベースの内容」を参照してください。)
  認証に成功した場合、認証情報をセションに格納する。認証エラーの場合は、ポップアップでエラーメッセージを表示する。
  認証部には、ユーザーIDと[ログアウト]ボタンを表示する。
  一覧部には、公開カレンダと認証後の私用カレンダをマージして、ソート処理を行ったものを表示する。

・ [ログアウト] ボタン
認証部には、空の入力フィールドと[ログイン]ボタンを表示する。
一覧部には、公開カレンダ(認証不要)を表示する。
セションに格納した認証情報をクリアする。

開発手順

1. 利用するWebサービスの決定
Ajaxフレームワークアプリケーションで利用するWebサービスを決定します。詳細は、「5.10.1 利用するWebサービスの決定(マッシュアップ開発例)」を参照してください。

2. Ajaxフレームワークプロジェクトの作成
Interstage Studioワークベンチを利用して、Ajaxフレームワークアプリケーションを開発するためのプロジェクトを作成します。詳細は、「5.10.2 プロジェクトの作成(マッシュアップ開発例)」を参照してください。

3. XSLの作成(スクレイピング)
スクレイピングツールを利用して、Webアプリケーションをスクレイピングし、XSLファイルを作成します。詳細は、「5.10.3 XSLの作成(マッシュアップ開発例)」を参照してください。

4. マッシュアップ定義ファイルの設定
Interstage Studioワークベンチを利用してマッシュアップ定義ファイル(muf.xml)に必要な定義を記述します。詳細は、「5.10.4 マッシュアップ定義ファイルの設定(マッシュアップ開発例)」を参照してください。

5. 画面フォーム(ひな形)の作成
Interstage Studioワークベンチを利用して、画面フォームのひな形(JSPファイル)を作成します。詳細は、「5.10.5 画面フォーム(ひな形)の作成(マッシュアップ開発例)」を参照してください。

6. 画面フォームの作成
Interstage Studioワークベンチを利用して、画面フォームのひな形(JSPファイル)に以下の定義を記述し、画面フォームを作成します。

   ー ユーザーデータの定義
   詳細は、「5.10.6 ユーザーデータの定義(マッシュアップ開発例)」を参照してください。

   ー 画面部品の定義
   詳細は、「5.10.7 画面部品の定義(マッシュアップ開発例)」を参照してください。

7. データBeanの作成
Apcoordinator用のデータBeanを作成します。詳細は、「5.10.8 データBeanの作成(マッシュアップ開発例)」を参照してください。

8. ビジネスクラスの作成・コマンドマップの定義
Apcoordinator用のビジネスクラスを作成し、コマンドマップを定義します。詳細は、「5.10.9 ビジネスクラスの作成(マッシュアップ開発例)」を参照してください。

9. Ajaxフレームワーク環境定義ファイルの設定
Interstage Studioワークベンチを利用して、Ajaxフレームワーク環境定義ファイル(acf.xml)に必要な定義を記述します。詳細は、「5.10.10 Ajaxフレームワーク環境定義ファイルの設定(マッシュアップ開発例)」を参照してください。

10. 動作定義
Interstage Studioワークベンチを利用して、JSPファイルに動作を実装するための定義を記述します。詳細は、「5.10.11 動作定義(マッシュアップ開発例)」を参照してください。

ポイント

この開発例では、使用するJavaScriptを、外部ファイルではなく、画面フォーム(JSPファイル)に記述します。

5.10.1 利用するWebサービスの決定(マッシュアップ開発例)

Ajaxフレームワークアプリケーションで利用するWebサービスを決定します。HTML形式やXML形式のWebサービスからHTMLアダプタとRESTアダプタによって、JSON形式に変換されたものを利用することができます。

この開発例では、以下のようなWebサービスにアクセスして、マッシュアップを行うことを想定しています。

- REST形式のWebサービス(サービス名: PublicCalendar)
  アクセスに認証が不要で、公開カレンダの一覧をREST形式で出力するWebサービスです。
- HTML形式のWebサービス(サービス名: PrivateCalendar)
  アクセスに基本認証(ユーザーID/パスワード)が必要で、個人の私用カレンダの一覧をHTML形式で出力するWebサービスです。

ポイント

上記のWebサービスは、サンプルアプリケーション(muServicesサンプルアプリケーション)として提供されています。

REST形式のWebサービス

REST形式のデータ例

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<appointment_list>
  <appoint>
    <date>2008/10/08</date>
    <timerange>13:00 - 17:00</timerange>
    <subject>仕様書レビュー</subject>
  </appoint>
  <appoint>
    <date>2008/10/22</date>
    <timerange>15:00 - 17:30</timerange>
    <subject>社内打ち合わせ</subject>
  </appoint>
  <appoint>
    <date>2008/10/28</date>
    <timerange>15:00 - 17:30</timerange>
    <subject>社内打ち合わせ(2nd)</subject>
  </appoint>
</appointment_list>
```

JSON形式に変換後のデータ(注)

```
{
  "tagName":"appointment_list",
  "childNodes":[
    {"tagName":"appoint",
     "childNodes":[
      {"tagName":"date","childNodes":["2008/10/08"]},
      {"tagName":"timerange","childNodes":["13:00 - 17:00"]},
      {"tagName":"subject","childNodes":["仕様書レビュー"]}
    ]},
    {"tagName":"appoint",
     "childNodes":[
      {"tagName":"date","childNodes":["2008/10/22"]},
      {"tagName":"timerange","childNodes":["15:00 - 17:30"]},
      {"tagName":"subject","childNodes":["社内打ち合わせ"]}
    ]},
    {"tagName":"appoint",
     "childNodes":[
      {"tagName":"date","childNodes":["2008/10/28"]},
      {"tagName":"timerange","childNodes":["15:00 - 17:30"]},
      {"tagName":"subject","childNodes":["社内打ち合わせ(2nd)"]}
    ]}
  ]
}
```

注) 実際のデータに改行は含まれません。

HTML形式のWebサービス

HTML形式のデータ例

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>私用カレンダ</title>
<style type="text/css">
th {background-color: silver;}
</style>
</head>
<body>
<span>user1 さんの予定を表示しています。</span>
<hr />
<table border="1">
  <tr>
    <th>予定日</th><th>時間帯</th><th>表題</th>
  </tr>
  <tr>
    <td>2008/10/03</td><td>09:30 - 12:00</td><td>健康診断</td>
  </tr>
  <tr>
    <td>2008/10/16</td><td>09:30 - 17:30</td><td>私用で休み</td>
  </tr>
  <tr>
    <td>2008/10/19</td><td>19:30 - 21:30</td><td>同窓会</td>
  </tr>
</table>
</body>
</html>
```

HTML形式のデータの場合、スクレイピングツールを利用する必要があります。詳細は、「5.10.3 XSLの作成(マッシュアップ開発例)」を参照してください。

5.10.2 プロジェクトの作成(マッシュアップ開発例)

Interstage Studioワークベンチ(Mashupファセット)を利用して、Ajaxフレームワークアプリケーションを開発するためのプロジェクトを作成します。
Interstage Studioワークベンチの操作方法については、「5.7.1 Ajaxフレームワークプロジェクトの作成」を参照してください。

プロジェクト作成時には、以下の情報を入力します。

- プロジェクト名: calendar
- プロジェクト構成: マッシュアップフレームワーク機能を利用する(注)

  上記以外の項目は、デフォルトの内容でプロジェクトを作成します。
  注) Project Facetsダイアログボックスで、Mashupにチェックします。

Interstage Studioワークベンチを利用してプロジェクトを作成すると、自動的にアプリケーションクラス(CalendarApplication.java)が作成されます。
通常は、作成されたアプリケーションクラスに、アプリケーション共通のデータを定義しますが、この開発例では、共通のデータは定義しません。
なお、セションクラス、ファクトリクラスも同様に作成されますが、この開発例では編集する必要はありません。

5.10.3 XSLの作成(マッシュアップ開発例)

スクレイピングツールを利用して、Webアプリケーションをスクレイピングし、XSLファイルを作成します。
スクレイピングツールの操作方法については、「付録G スクレイピングツール」を参照してください。

XSL作成時には、以下の情報を入力します。

・ ファイル名：private_cal.xsl
bodyタグ内のtableタグを選択します。

HTML形式のデータは、HTMLアダプタで正規化され、XHTML形式のデータとして変換されます。スクレイピングツールを利用すると、XSLTプロセッサを使用して、XHTML形式のデータから必要な部分を切り出すことができます。

XSLファイルの例

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XSLT stylesheet generated by IIM on 2009/08/17 -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0"
 xpath-default-namespace="http://www.w3.org/1999/xhtml">
<xsl:output method="xml" omit-xml-declaration="yes" indent="yes" />

<!-- create tags for the document root -->
<xsl:template match="/">
<html>
<xsl:copy-of select="/html[1]/body[1]/table[1]
" />
</html>
</xsl:template>

<!-- default text template -->
<xsl:template priority="0" match="text()" />

</xsl:stylesheet>
```

切り出されたデータの例

```
<html>
  <table border="1">
    <tr>
      <th>予定日</th><th>時間帯</th><th>表題</th>
    </tr>
    <tr>
      <td>2008/10/03</td><td>09:30 - 12:00</td><td>健康診断</td>
    </tr>
    <tr>
      <td>2008/10/16</td><td>09:30 - 17:30</td><td>私用で休み</td>
    </tr>
    <tr>
      <td>2008/10/19</td><td>19:30 - 21:30</td><td>同窓会</td>
    </tr>
  </table>
</html>
```

JSON形式に変換後のデータ(注)

```
{
  "tagName":"html",
  "childNodes":[
    {"tagName":"table",
     "attributes":{"border":"1"},
     "childNodes":[
       {"tagName":"tr",
        "childNodes":[
          {"tagName":"th","childNodes":["予定日"]},
          {"tagName":"th","childNodes":["時間帯"]},
          {"tagName":"th","childNodes":["表題"]}
       ]},
       {"tagName":"tr",
        "childNodes":[
          {"tagName":"td","childNodes":["2008/10/03"]},
          {"tagName":"td","childNodes":["09:30 - 12:00"]},
```

```
          {"tagName":"td","childNodes":["健康診断"]}
        ]},
        {"tagName":"tr",
         "childNodes":[
          {"tagName":"td","childNodes":["2008/10/16"]},
          {"tagName":"td","childNodes":["09:30 - 17:30"]},
          {"tagName":"td","childNodes":["私用で休み"]}
        ]},
        {"tagName":"tr",
         "childNodes":[
          {"tagName":"td","childNodes":["2008/10/19"]},
          {"tagName":"td","childNodes":["19:30 - 21:30"]},
          {"tagName":"td","childNodes":["同窓会"]}
        ]}
      ]
     }
   ]
}
```

注) 実際のデータに改行は含まれません。

5.10.4 マッシュアップ定義ファイルの設定(マッシュアップ開発例)

Interstage Studioワークベンチを利用して、マッシュアップ定義ファイル(muf.xml)に必要な定義を記述します。
Interstage Studioワークベンチの操作方法については、「5.7.11 マッシュアップ定義ファイルの編集」を参照してください。

利用するWebサービスの定義

Ajaxフレームワークアプリケーションで利用するWebサービスを定義します。

サービス定義の編集時には、以下の情報を入力します。

サービス名	アダプタ名	URL	XSL
PrivateCalendar	htmladaptor	http://localhost/muServices/Services/auth/service1(注)	private_cal.xsl
PublicCalendar	restadaptor	http://localhost/muServices/Services/auth/service2(注)	指定なし

注) muServicesサンプルアプリケーションを利用した場合のURLです。ホスト名やポート番号など、実際に利用するWebサービスのURLを指定してください。

マッシュアップ定義ファイル(muf.xml)の記述内容

```
<!-- Webサービスの定義 -->
  <muf_services>
      <!-- HTML形式の私用カレンダ（サービスID："PrivateCalendar"） -->
      <muf_service name="PrivateCalendar">
          <adaptor name="htmladaptor"/>
          <parameters>
              <parameter
                  name="URL"
                  value="http://localhost/muServices/Services/auth/service1" />
              <!-- スクレイピングツールの定義 -->
              <parameter
                  name="XSL"
                  value="private_cal.xsl" />
          </parameters>
      </muf_service>
      <!-- REST形式の公開カレンダ（サービスID："PublicCalendar"） -->
      <muf_service name="PublicCalendar">
          <adaptor name="restadaptor"/>
          <parameters>
              <parameter
                  name="URL"
```

```
                value="http://localhost/muServices/Services/auth/service2" />
        </parameters>
    </muf_service>
</muf_services>
```

5.10.5 画面フォーム(ひな形)の作成(マッシュアップ開発例)

Interstage Studioワークベンチを利用して、画面フォームのひな形(JSPファイル)を作成します。
Interstage Studioワークベンチの操作方法については、「5.7.2 画面フォーム(ひな形)の作成」を参照してください。

画面フォームのひな形を作成するには、Ajax JSPウィザードを使用します。
画面フォームのひな形作成時には、以下の情報を入力します。

- ファイル名: mainPage.jsp

5.10.6 ユーザーデータの定義(マッシュアップ開発例)

「5.10.5 画面フォーム(ひな形)の作成(マッシュアップ開発例)」で作成した画面フォームのひな形に、Ajaxページエディタを利用して、ユーザーデータの定義を記述します。
Ajaxページエディタの操作方法については、「付録F Ajaxページエディタ」を参照してください。

以下のように、画面項目のブロック単位にデータ格納域を定義します。

- 認証部: authData
- 一覧部: listData

ユーザーデータ定義の記述内容

```
// 認証部
authData = {userid: "", password: ""};

// 一覧部
listData = {list: []};
```

5.10.7 画面部品の定義(マッシュアップ開発例)

「5.10.6 ユーザーデータの定義(マッシュアップ開発例)」でユーザーデータを定義した画面フォームに、Ajaxページエディタを利用して、画面部品の定義を記述します。
Ajaxページエディタの操作方法については、「付録F Ajaxページエディタ」を参照してください。

画面部品の定義では、ユーザーデータをモデルとして利用するための機能定義と、UI部品を使用するための定義を記述します。

認証部の機能定義とUI部品定義

機能定義の記述内容

```
<!--  認証部のユーザーデータをモデルとして利用するための機能定義 -->
<div rcf:type="Model" rcf:id="authinfoModel" rcf:object="authData"></div>
```

UI部品定義の記述内容

```
<div rcf:id="authinfoContainer" rcf:type="ViewStack" rcf:width="700px" rcf:height="50px"
 style="left: 0px; top: 5px; position: absolute" rcf:selectedIndex="0">
 <!-- ログイン認証前 -->
 <div rcf:id="beforeauthContainer" rcf:type="ViewContainer" rcf:label="ログイン認証前"
  rcf:width="620px" rcf:height="30px" style="left: 5px; top: 10px; position: absolute">
  <!-- 認証部のUI部品定義とモデルバインディング -->
  <div id="before_div"
   style="visibility: hidden; width: 620px; height: 30px; left: 0px; top: 0px; position: absolute">

   <div rcf:id="label_userid" rcf:type="Text" rcf:value="ユーザーID"
    style="left: 5px; top: 8px; position: absolute"></div>
   <!-- userid入力 テキスト入力フィールド定義 -->
```

```
  <div rcf:id="userid" rcf:type="TextInput" rcf:width="155px" rcf:height="20px"
   style="left: 80px; top: 5px; position: absolute" rcf:value="{authinfoModel.userid}"></div>  (1)
  <div rcf:id="label_password" rcf:type="Text" rcf:value="パスワード"
   style="left: 260px; top: 8px; position: absolute"></div>
  <!-- password入力 テキスト入力フィールド定義 -->
  <div rcf:id="password" rcf:type="TextInput" rcf:width="155px" rcf:height="20px"
   style="left: 340px; top: 5px; position: absolute" rcf:value="{authinfoModel.password}"       (2)
   rcf:password="true"></div>
  <!-- ログインボタン定義 -->
  <div rcf:id="loginButton" rcf:type="Button" rcf:label="ログイン"
   rcf:width="80px" rcf:height="24px" style="left: 515px; top: 5px; position: absolute"></div>

 </div><!-- id="before_div" -->
</div>

<!-- ログイン認証後 -->
<div rcf:id="afterauthContainer" rcf:type="ViewContainer" rcf:label="ログイン認証後"
 rcf:width="620px" rcf:height="30px" style="left: 5px; top: 10px; position: absolute">
 <!-- 認証部のUI部品定義とモデルバインディング -->
 <div id="after_div"
  style="visibility: hidden; width: 620px; height: 30px; left: 0px; top: 0px; position: absolute">

   <!-- userid テキストフィールド定義 -->
   <div rcf:id="userid_Text" rcf:type="Text" rcf:value="        "
    style="left: 5px; top: 8px; position: absolute"></div>
   <div rcf:id="label_message" rcf:type="Text" rcf:value=" さん、こんにちは。認証されました。"
    style="left: 50px; top: 8px; position: absolute"></div>
   <!-- ログアウトボタン定義 -->
   <div rcf:id="logoutButton" rcf:type="Button" rcf:label="ログアウト"
    rcf:width="80px" rcf:height="24px" style="left: 515px; top: 5px; position: absolute"></div>

 </div><!-- id="after_div" -->
</div>

</div>
```

1) モデルオブジェクトauthinfoModelのuseridプロパティをTextInputの値にバインディング
2) モデルオブジェクトauthinfoModelのpasswordプロパティをTextInputの値にバインディング

一覧部の機能定義とUI部品定義

機能定義の記述内容

```
<!-- 一覧部のユーザーデータをモデルとして利用するための機能定義 -->
<div rcf:type="Model" rcf:id="listModel" rcf:object="listData"></div>
```

UI部品定義の記述内容

```
<!-- 一覧部のUI部品定義とモデルバインディング -->
<div rcf:id="list_Title" rcf:type="Text" rcf:value="公開カレンダ(認証不要)"
 style="left: 10px; top: 70px; position: absolute" rcf:fontWeight="bold"></div>

<div rcf:id="list" rcf:type="TableView" rcf:width="610px" rcf:height="200px"
 style="left: 10px; top: 95px; position: absolute" rcf:data="{listModel.list}"                  (1)
 rcf:showRowHeader="true" rcf:multipleSelect="false" rcf:showDummyColumn="false">
  <div rcf:type="ViewColumn" rcf:name="date" rcf:label="日付" rcf:columnWidth="120"></div>       (2)
  <div rcf:type="ViewColumn" rcf:name="timerange" rcf:label="時間帯" rcf:columnWidth="120"></div> (2)
  <div rcf:type="ViewColumn" rcf:name="subject" rcf:label="予定" rcf:columnWidth="220"></div>    (2)
  <div rcf:type="ViewColumn" rcf:name="secret" rcf:label="Private" rcf:columnWidth="90"></div>   (2)
</div>
```

1) モデルオブジェクトlistModelのlistプロパティをTableViewの値にバインディング
2) 列情報(1行分のデータ項目)を定義

5.10.8 データBeanの作成(マッシュアップ開発例)

Interstage Studioワークベンチを利用して、データBeanを作成します。

認証部のデータをApcoordinatorに受け渡すために、以下のデータBeanを作成します。

- 認証用データBean: CalendarBean

認証用データBean作成時には、以下の情報を入力します。

- パッケージ名: calendar
- クラス名: CalendarBean
- プロパティには、以下の内容を追加します。

項目名	プロパティ名	型
ユーザーID	userid	String
パスワード	password	String

5.10.9 ビジネスクラスの作成(マッシュアップ開発例)

Interstage Studioワークベンチを利用して、ビジネスクラスを作成します。

ビジネスクラスの作成

ビジネスクラス作成時には、以下の情報を入力します。

- パッケージ名: calendar
- クラス名: CalendarHandler
- メソッドには、以下の内容を追加します。

入力データ	入力情報	コマンド	メソッド	復帰値型
入力データなし	指定しない	指定しない	startup	void
データBean	クラス名: CalendarBean	login	login	object
入力データなし	指定しない	logout	logout	void

作成されたビジネスクラスに、以下の記述を追加します。

startupメソッドの記述内容

```
/* 初期化処理 */
public void startup(DispatchContext context) {
     CalendarBean calendarBean = new CalendarBean();
     context.setResponseBean("body", calendarBean);
}
```

loginメソッドの記述内容

```
/* ログイン処理 */
public Object login(DispatchContext context, calendar.CalendarBean dataBean) {
    String globalId = dataBean.getUserid();   // 入力値：ユーザーID
    String password = dataBean.getPassword(); // 入力値：パスワード
    String serviceId = "PrivateCalendar";     // サービス名："PrivateCalendar"(固定)

    // ログイン処理(入力値によるログイン可否)を実行
    if(doLogin(globalId, password) == false){
        // ログイン失敗時の処理
        dataBean.setUserid("");
        dataBean.setPassword("");
        return dataBean;
```

```
    }

    // ログイン成功時、以下の処理を実行
    // ユーザー管理簿からglovalIdとserviceIdをもとに基本認証情報を取得
    BasicAuthInfo basicinfo = BasicAuthDB.getBasicAuthInfoFromDB(globalId, serviceId);
    // 基本認証情報からユーザーID、パスワードを取得
    String basic_uid = basicinfo.getUserid();
    String basic_pwd = basicinfo.getPassword();
    // 基本認証情報の生成
    BasicAuth bAuth = new BasicAuth();
    bAuth.setGlobalID(globalId);
    try {
        bAuth.setUserID(basic_uid);
        bAuth.setPassword(basic_pwd);
    } catch (MUEncryptException e) {
        // 暗号化の設定に対する例外の発生・または暗号化アルゴリズムの初期化に失敗
        dataBean.setUserid("");
        dataBean.setPassword("");
        return dataBean;
    }
    // サービス認証情報の生成
    ServiceAuth sAuth = new ServiceAuth(serviceId);
    // サービス認証情報に基本認証情報を格納
    sAuth.setAuth(bAuth);
    // サービス認証情報リストの生成
    AuthorizationList authlist = new AuthorizationList(globalId);
    // 認証情報リストにサービス認証情報を格納
    authlist.add(sAuth);
    // セションの取得
    HttpSessionProfile sessionprofile = (HttpSessionProfile)context.getSessionProfile();
    HttpSession session = sessionprofile.getSession();
    // セションに認証情報リストを格納
    session.setAttribute(Authorization.SESSION_AUTHORIZATION_KEY, authlist);

    return dataBean;
}
```

logoutメソッドの記述内容

```
/* ログアウト処理 */
public void logout(DispatchContext context) {
    // セションの取得
    HttpSessionProfile sessionprofile = (HttpSessionProfile)context.getSessionProfile();
    HttpSession session = sessionprofile.getSession();
    // セション情報をクリア
    session.removeAttribute(Authorization.SESSION_AUTHORIZATION_KEY);
}
```

doLoginメソッド(注1)の記述内容

```
/* 入力された情報でのログイン処理 */
private boolean doLogin(String globalId, String password) {
    // ユーザーID/パスワードの照合
    if(globalId.equals("gid001") && password.equals("001") ||
        :(省略)
        globalId.equals("gid005") && password.equals("005"))
    {
        return true;
    }
    // ユーザーID/パスワードが違っている場合
    else{
        return false;
```

```
    }
}
```

注1) このメソッドは自動生成されません。

BasicAuthDBクラス(注2)の記述内容

```
/**
 * 仮想の基本認証データベース(ユーザー管理簿)です。
 */
public class BasicAuthDB {
    // 基本認証データベース
    private static final BasicAuthInfo[] basicDB = {
        // 基本認証情報(グローバルID、サービスID、ユーザーID、パスワード)
        new BasicAuthInfo("gid001","PrivateCalendar","user1","user1"),
            :(省略)
        new BasicAuthInfo("gid005","PrivateCalendar","user5","user5")
    };

    /** グローバルID、サービスIDをキーに基本認証データベースから認証処理を取得します。 */
    public static BasicAuthInfo getBasicAuthInfoFromDB(String globalid, String serviceid){
        // データベース内の検索
        for(int i=0; i<basicDB.length; i++){
            // 一致した場合、認証情報を返す
            if(basicDB[i].isTargetBasicAuthinfo(globalid, serviceid) == true){
                return basicDB[i];
            }
        }
        // 一致しない場合、nullを返す
        return null;
    }
}
```

注2) このクラスは自動生成されません。

BasicAuthInfoクラス(注3)の記述内容

```
/** 基本認証情報クラスです。 */
public class BasicAuthInfo {
    private String globalid;
    private String serviceid;
    private String userid;
    private String password;

    /** コンストラクタです。 */
    public BasicAuthInfo(String gid,String sid,String uid,String pw){
        globalid = gid;
        serviceid = sid;
        userid = uid;
        password = pw;
    }
    public String getGlobalid(){
        return globalid;
    }
    public String getServiceid(){
        return serviceid;
    }
    public String getUserid(){
        return userid;
    }
    public String getPassword(){
        return password;
    }
    /** グローバルID、サービスIDの値と一致した場合、true を返します。 */
```

```
    public boolean isTargetBasicAuthinfo(String gid, String sid){
        if(globalid.equals(gid) && serviceid.equals(sid)){
            return true;
        }
        else{
            return false;
        }
    }
}
```

注3) このクラスは自動生成されません。

ログイン情報および認証情報データベースの内容(注4)

サンプルアプリケーションのログイン情報です。

ログインユーザーID	ログインパスワード
gid001	001
gid002	002
gid003	003
gid004	004
gid005	005

基本認証サービス(私用カレンダアプリケーション)の認証情報です。GLOBALID(入力されたユーザーID)とSERVICEID("PrivateCalendar")をキーに、認証情報(USERIDとPASSWORD)を取得します。
以下の情報は、内部情報として保持されており、認証サービスにアクセスする際に利用されます。

GLOBALID	SERVICEID	USERID	PASSWORD
gid001	PrivateCalendar	user1	user1
gid002	PrivateCalendar	user2	user2
gid003	PrivateCalendar	user3	user3
gid004	PrivateCalendar	user4	user4
gid005	PrivateCalendar	user5	user5

注4) 本来は、RDBやLDAPなどで管理、運用されるデータです。

コマンドマップの定義

ビジネスクラスを作成すると、自動的にコマンドマップ(commands.map)が作成されます。

コマンドマップの記述内容

```
# 初期化処理
;=calendar.CalendarHandler.startup
# ログイン処理実行
calendar.CalendarBean;login=calendar.CalendarHandler.login
# ログアウト処理実行
;logout=calendar.CalendarHandler.logout
```

コマンドマップの詳細は、Interstage Application Serverの「Apcoordinator ユーザーズガイド」を参照してください。

ページマップの定義

ページマップ(pages.map)に必要な定義を追加します。

ページマップの記述内容

```
# pages.map
calendar.CalendarBean;=mainPage.jsp
```

ページマップの詳細は、Interstage Application Serverの「Apcoordinator ユーザーズガイド」を参照してください。

5.10.10 Ajaxフレームワーク環境定義ファイルの設定(マッシュアップ開発例)

Interstage Studioワークベンチを利用して、Ajaxフレームワーク環境定義ファイル(acf.xml)に必要な定義を記述します。

データBeanの定義

Apcoordinator用に作成したデータBeanを定義します。

acf.xmlの記述内容

```
<!-- データBean定義 -->
  <dataBeans>
    <!-- 認証条件(ログイン) -->
     <dataBean>
        <dataBeanId>dataBeanId1</dataBeanId>
        <className>calendar.CalendarBean</className>
        <scope>request</scope>
     </dataBean>
  </dataBeans>
```

5.10.11 動作定義(マッシュアップ開発例)

「5.10.7 画面部品の定義(マッシュアップ開発例)」で画面部品を定義した画面フォームに、Ajaxページエディタを利用して、ログイン・ログアウト処理に関する動作定義を記述します。
Ajaxページエディタの操作方法については、「付録F Ajaxページエディタ」を参照してください。

動作定義では、ログインボタン・ログアウトボタンの定義と、イベントの定義・登録、初期表示・ログイン・ログアウト処理の定義を記述します。

ログインボタン・ログアウトボタンの定義

画面の[ログイン]ボタンと[ログアウト]ボタンを定義します。

記述内容

```
<!-- ログインボタン定義 -->
<div rcf:id="loginButton" rcf:type="Button" rcf:label="ログイン" rcf:width="80px" rcf:height="24px"
 style="left: 515px; top: 5px; position: absolute"></div>
<!-- ログアウトボタン定義 -->
<div rcf:id="logoutButton" rcf:type="Button" rcf:label="ログアウト" rcf:width="80px" rcf:height="24px"
 style="left: 515px; top: 5px; position: absolute"></div>
```

イベントの定義

作成したボタンにイベントを定義します。

記述内容

```
// イベント定義
calEvent = {
  loginButton:{              (1)
    click:login              (2)
  },
  logoutButton:{             (1)
    click:logout             (3)
  }
};
```

1) 作成したボタンを指定
2) クリック時にlogin関数を実行するように定義
3) クリック時にlogout関数を実行するように定義

イベントの登録

初期処理の登録APIを利用して、定義したイベントを登録します。

記述内容

```
// 初期処理定義
RCF.addInitializedListener(
  function(eventObject) {
    // イベント登録
    rcf.event.EventRegistrar.registerEvents(calEvent, "calEvent");    (1)
    init();                                                           (2)
  }
);
```

1) 定義したイベントに「calEvent」という名前を付けて登録
2) 初期表示時にinit関数を実行するように定義

初期表示処理の定義

イベントの初期処理定義で記述した関数を定義し、Apcoordinatorのビジネスクラスを呼び出す関数を記述します。

記述内容

```
// 初期処理定義
function init(){
  // 初期表示時のみコンテナ部品の表示制御
  document.getElementById("before_div").style.visibility = "visible";
  document.getElementById("after_div").style.visibility = "visible";

  // 一覧部の情報をクリア
  listModel.setProperty("list", []);

  // Webサイト(公開カレンダ)にアクセス
  accessCalendar("PublicCalendar", "json");
}

// 外部との通信を行うメイン処理定義(マッシュアップフレームワークを利用)
function accessCalendar(serviceId, type){
  // queryData(送信パラメーターの指定)
  var queryData = {};
  // requestParam(サービスIDとタイプの指定)
  var reqParam = {serviceId: serviceId, type: type};          (1) (2)
  // option
  var option = {
    url: "muf/proxy",
    callback: function(res){                                  (3)
      // 公開カレンダ・データの取得(RESTアダプタ：XML→JSON形式)
      if(serviceId == "PublicCalendar"){
        var app_list = res.childNodes;
        for(i=0; i<app_list.length; i++){
          var responseData = new Object();
          responseData.date      = app_list[i].childNodes[0].childNodes[0];
          responseData.timerange = app_list[i].childNodes[1].childNodes[0];
          responseData.subject   = app_list[i].childNodes[2].childNodes[0];
          responseData.secret    = "";
          // 1行ごとテーブルに追加(画面部品に関連づけ)
          listModel.getDataProvider("list").addItem(responseData);
        }
      }
      // 私用カレンダ・データの取得(HTMLアダプタ：HTML→JSON形式)
      else if(serviceId == "PrivateCalendar"){
        var tbl_tr = res.childNodes[0].childNodes;
        // 1行目(i=0)は見出し行のため省略(インデックス1より開始)
        for(i=1; i<tbl_tr.length; i++){
          var responseData = new Object();
          responseData.date      = tbl_tr[i].childNodes[0].childNodes[0];
          responseData.timerange = tbl_tr[i].childNodes[1].childNodes[0];
          responseData.subject   = tbl_tr[i].childNodes[2].childNodes[0];
```

```
          responseData.secret    = "Private";
          // 1行ごとテーブルに追加(画面部品に関連づけ)
          listModel.getDataProvider("list").addItem(responseData);
        }
      }

      // TableView のソート(callbackの最後で行う)
      list.sort("date", true);
    },
    // エラー処理定義
    errorHandler: function(err) {
        alert("マッシュアップフレームワークでエラーが発生しました。¥n" + err.message);
    }
  };

  // Webサービス呼出し
  MuRequest.send(queryData, reqParam, option);                    (4)
}
```

1) マッシュアップ定義ファイルに定義したWebサービスのサービスIDを指定
2) マッシュアッププロキシからのレスポンス形式(XML形式またはJSON形式)を指定
3) 通信完了時のコールバック処理を定義
4) 送信するクエリ文字列オブジェクト、リクエストパラメーターオブジェクト、通信設定オブジェクトを指定して、通信実行

ログイン処理の定義

イベントの定義で記述した関数を定義し、Apcoordinatorのビジネスクラスを呼び出す関数を記述します。

記述内容

```
// ログイン処理定義
function login(){
  // ログイン認証チェック(通信フレームワークを利用)
  // requestParams
  var reqParam = {
    beanId: "dataBeanId1",                                        (1)
    verb: "login"                                                 (2)
  };
  // option
  var option = {
    url: "acf/apc",
    callback: function(res){                                      (3)
      // ユーザーIDが空でない場合、ログイン成功
      if(res.userid != ""){
        // 認証に成功した場合、Webサイト(私用カレンダ)にアクセス
        accessCalendar("PrivateCalendar", "json");

        // Containerと見出しの切替え
        authinfoContainer.setSelectedContainerId("afterauthContainer");
        userid_Text.setProperty("value", res.userid);
        list_Title.setProperty("value","公開カレンダ ＋ 私用カレンダ");
      }
      // ユーザーIDが空の場合、認証エラー
      else{
        alert("認証に失敗しました。");
      }
    },
    // エラー処理定義
    errorHandler: function(err) {
        alert("通信フレームワークでエラーが発生しました。¥n" + err.message);
    }
  };
  // 通信サービス呼出し
```

```
  UjiRequest.send(authData, reqParam, option);          (4)
}
```

1) 送信データを格納するデータBean(Ajaxフレームワーク環境定義ファイルに定義したデータBean ID)を指定
2) 呼び出すメソッドに対応するコマンド名(コマンドマップに定義したコマンド名)を指定
3) 通信完了時のコールバック処理を定義
4) 送信するデータオブジェクト、リクエストパラメーターオブジェクト、通信設定オブジェクトを指定して、通信実行

ログアウト処理の定義

イベントの定義で記述した関数を定義し、Apcoordinatorのビジネスクラスを呼び出す関数を記述します。

記述内容

```
// ログアウト処理定義
function logout(){
  // 一覧部の情報をクリア
  listModel.setProperty("list", []);

  // Webサイト(公開カレンダ)にアクセス
  accessCalendar("PublicCalendar", "json");

  // ログアウト処理(通信フレームワークを利用)
  // requestParams
  var reqParam = {
    beanId: "",                                         (1)
    verb: "logout"                                      (2)
  };
  // option
  var option = {
    url: "acf/apc",
    callback: function(res){                            (3)
      // 入力域の初期化
      authinfoModel.setProperty("userid", "");
      authinfoModel.setProperty("password", "");

      // Containerと見出しの切替え
      authinfoContainer.setSelectedContainerId("beforeauthContainer");
      userid_Text.setProperty("value", "");
      list_Title.setProperty("value", "公開カレンダ(認証不要)");
    },
    // エラー処理定義
    errorHandler: function(err) {
        alert("通信フレームワークでエラーが発生しました。¥n" + err.message);
    }
  };
  // 通信サービス呼出し
  UjiRequest.send(null, reqParam, option);              (4)
}
```

1) 送信データを格納するデータBeanは不要なため、データBean IDは指定しない
2) 呼び出すメソッドに対応するコマンド名(コマンドマップに定義したコマンド名)を指定
3) 通信完了時のコールバック処理を定義
4) 送信するデータオブジェクト、リクエストパラメーターオブジェクト、通信設定オブジェクトを指定して、通信実行

5.11 文字コードについて

Ajaxフレームワークでは、複数のファイルを結合して1つの画面を作る場合があります。その際、異なる文字コードが混在すると文字化けする可能性があります。このため、プロジェクト内のHTML/JSPファイルは、文字コードを統一してください。

Ajaxフレームワーク開発環境では、以下の方法で作成した画面フォームの文字コードは、UTF-8で生成されます。

- Ajaxフレームワークプロジェクトウィザードで作成される以下のファイル
  - 制御ページ
  - エラーページ
- Ajax JSPウィザードで作成される以下のファイル
  - 画面フォーム定義
  - ユーザーロジック定義

UTF-8以外の文字コードで画面フォームを作成したい場合は、以下の手順で作成してください。

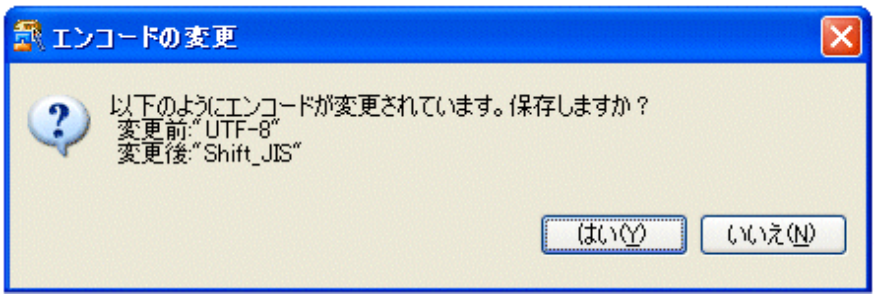
制御ページの場合

Ajaxフレームワークプロジェクトウィザードで生成される制御ページの文字コードは、UTF-8です。
文字コードをUTF-8 以外に設定したい場合は、以下の手順で設定してください。

1. 制御ページをJSPエディタで開いて、テキストタブを選択し、ソースビューを表示します。
   デフォルトの設定でプロジェクトを作成した場合の制御ページの表示例を以下に示します。

```
<%@ page contentType= "text/html; charset=UTF-8" %>
<%@ taglib uri="uji-taglib" prefix="uji" %>
 (省略)
```

2. <%@ page contentType= "text/html; charset=UTF-8" %>の「UTF-8」を、変更したい文字コードに修正します。
3. Interstage Studioの場合は、保存しようとすると以下のメッセージが表示されますので、[はい]ボタンをクリックします。Eclipseの場合は、特に確認のメッセージはなく保存されます。



エラーページの場合

Ajaxフレームワークプロジェクトウィザードで生成されるエラーページの文字コードは、UTF-8です。
文字コードをUTF-8以外に設定したい場合は、以下の手順で設定してください。

1. エラーページをJSPエディタで開いて、テキストタブを選択し、ソースビューを表示します。
   表示例を以下に示します。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<%@ page contentType= "text/html; charset=UTF-8" %>
<%@ page isErrorPage="true" %>
 (省略)
```

2. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">、<%@ page contentType= "text/html; charset=UTF-8" %>の2箇所の「UTF-8」を、変更したい文字コードに修正します。
3. Interstage Studioの場合は、保存しようとすると[エンコードの変更]メッセージボックスが表示されますので、[はい]ボタンをクリックします。Eclipseの場合は、特に確認のメッセージはなく保存されます。

画面フォーム定義の場合

Ajax JSPウィザードで生成される画面フォーム定義の文字コードは、UTF-8です。
UTF-8以外の文字コードの画面フォームを作成したい場合は、以下の手順で作成してください。

Interstage Studioの場合

1. [ウィンドウ]メニューから[設定]を選択します。
2. [設定]ダイアログボックスの設定項目で、[Web] > [JSPファイル]を選択します。
3. JSPファイルの作成時のエンコードを、設定したい文字コードに修正します。
4. 「JSPウィザードで作成する場合」に従って、画面フォーム定義を作成します。

Eclipseの場合

1. [Window]メニューから[Preferences]を選択します。
2. [Preferences]ダイアログボックスの設定項目で、[Web] > [JSP Files]を選択します。
3. JSPファイルの作成時のエンコードを、設定したい文字コードに修正します。
4. 画面フォーム定義を作成します。
   開発手順は、「JSPウィザードで作成する場合」を参考にしてください。なお、日本語のメニュー名やコマンド名などは、英語の名称に読み替えてください。

ユーザーロジック定義の場合

Ajax JSPウィザードで生成されるユーザーロジック定義の文字コードは、UTF-8です。
UTF-8以外の文字コードのユーザーロジック定義を作成したい場合は、以下の手順で作成してください。

Interstage Studioの場合

- JSPで作成する場合

  1. [ウィンドウ]メニューから[設定]を選択します。
  2. [設定]ダイアログボックスの設定項目で、[Web] > [JSPファイル]を選択します。
  3. JSPファイルの作成時のエンコードを、設定したい文字コードに修正します。
  4. 「HTMLウィザードまたはJSPウィザードで作成する場合」に従って、ユーザーロジック定義を作成します。

- HTMLで作成する場合

  1. [ウィンドウ]メニューから[設定]を選択します。
  2. [設定]ダイアログボックスの設定項目で、[Web] > [HTMLファイル]を選択します。
  3. HTMLファイルの作成時のエンコードを、設定したい文字コードに修正します。
  4. 「HTMLウィザードまたはJSPウィザードで作成する場合」に従って、ユーザーロジック定義を作成します。
  5. 作成したファイルを選択し、コンテキストメニューから[プロパティ]を選択します。[プロパティ]ダイアログボックスで、テキストファイルエンコードに「その他」を選択し、3.で指定した文字コードを設定します。

Eclipseの場合

- JSPで作成する場合

  1. [Window]メニューから[Preferences]を選択します。
  2. [Preferences]ダイアログボックスの設定項目で、[Web] > [JSP Files]を選択します。
  3. JSPファイルの作成時のエンコードを、設定したい文字コードに修正します。
  4. ユーザーロジック定義を作成します。
     開発手順は、「HTMLウィザードまたはJSPウィザードで作成する場合」を参考にしてください。なお、日本語のメニュー名やコマンド名などは、英語の名称に読み替えてください。

- HTMLで作成する場合

  1. [Window]メニューから[Preferences]を選択します。
  2. [Preferences]ダイアログボックスの設定項目で、[Web] > [HTML Files]を選択します。
  3. HTMLファイルの作成時のエンコードを、設定したい文字コードに修正します。
  4. ユーザーロジック定義を作成します。
     開発手順は、「HTMLウィザードまたはJSPウィザードで作成する場合」を参考にしてください。なお、日本語のメニュー名やコマンド名などは、英語の名称に読み替えてください
  5. 作成したファイルを選択し、コンテキストメニューから[Properties]を選択します。[Properties]ダイアログボックスで、テキストファイルエンコードに「Other」を選択し、3.で指定した文字コードを設定します。

HTMLファイルやJSPファイルをUTF-8以外で作成する場合は、Ajaxフレームワークの初期化処理の記述にcharsetアトリビュートを追加し、使用する文字コード名を指定してください。

以下に、記述例を示します。

```
<script type="text/javascript" src="acf/file/rcf/rcf.js" charset="文字コード名"></script>
```

5.12 留意事項

ここでは、Ajaxフレームワークアプリケーションを開発する際の留意事項について説明します。

5.12.1 セキュリティについて

Ajaxフレームワークアプリケーションを開発する際は、セキュリティに関して、以下の点に注意してください。

- JavaScriptやCSSを外部ファイルにした場合は、アプリケーションに認証の仕組みを実装していても、アプリケーション外からJavaScriptやCSSにアクセスすることが可能な場合があります。そのため、JavaScriptやCSSには、コメント部分も含めて、アプリケーション外への漏えいを防ぐべき情報(秘密情報)を記述しないでください。
  秘密情報を画面上に表示する場合は、HTMLやJSPファイル内に直接記述したうえで認証の仕組みを実装するか、直接記述せずにサーバから動的にデータを取得するようにしてください。
- 入力値をチェックする場合は、クライアント側とサーバ側の両方でチェックしてください。
  クライアントでチェックする場合は、機能部品のModelでデータオブジェクトのスキーマを定義することができます。
  サーバ側でチェックする場合は、Apcoordinatorが提供するXMLデータ仕様記述を利用することができます。
  また、予期しないデータが送信された場合や、セション情報に不整合がある場合(セション内に必要なデータが格納されてない状態で、次のデータが送信されてきた場合など)は、アプリケーションの処理を継続しないように注意してください。
- Ajaxフレームワークが提供する画面部品では、画面表示時に以下の5種の文字をエスケープして表示しています。

  ```
  & < > " '
  ```

  ただし、Button部品のコンテンツに記述された文字列、および以下のプロパティに指定されたオブジェクト内の処理はエスケープされませんので、アプリケーションでエスケープ処理を実施してください。

  - SelectList
    ・rendererプロパティ
  - TabPanel
    ・tabRendererプロパティ
  - DataGrid
    ・selectedRendererプロパティ
  - ViewColumn
    ・rendererプロパティ

- ViewColumnGrid
  ・rendererプロパティ
  ・selectedRendererプロパティ
- ViewColumnCheck
  ・rendererプロパティ
  ・selectedRendererプロパティ
- ViewColumnTree
  ・rendererプロパティ
  ・selectedRendererプロパティ
- ViewColumnSelect
  ・rendererプロパティ
  ・selectedRendererプロパティ
- ViewColumnImage
  ・rendererプロパティ
  ・selectedRendererプロパティ
- Calendar
  ・naviButtonRendererプロパティ
  ・dayOfWeekCellRendererプロパティ
  ・dateCellRendererプロパティ
- PopupCalendar
  ・naviButtonRendererプロパティ
  ・dayOfWeekCellRendererプロパティ
  ・dateCellRendererプロパティ

・ イベントログ機能は、非同期通信でログをサーバ側に送信する機能です。ログの書込みに対して信頼性を保証する必要がある場合は、サーバ側の業務ロジックで行うようにしてください。

・ クライアントから複数のリクエストが同時に送信された場合、サーバ側の業務ロジックが同時に実行される可能性があります。同時に実行すべきではない業務処理については、アプリケーションで排他制御を行うようにしてください。

5.12.2 タグの属性値に関する警告について

Ajaxフレームワークアプリケーションを開発する際は、タグの属性値に関して、以下の点に注意してください。

・ Interstage StudioまたはEclipseのエディタを利用して、タグの属性値にバインディング式を記述した場合、以下の警告が表示される場合があります。

属性値 xxxx が未定義です。

この警告メッセージは、入力候補の選択値に指定値がないことを警告するもので、エラーを示すものではありません。正しいバインディング式が指定されていれば、動作には問題ありません。

5.12.3 タグの編集について

Ajaxフレームワークアプリケーションを開発する際は、タグの編集に関して、以下の点に注意してください。

・ HTMLエディタまたはJSPエディタでタグを編集する場合、以下の留意事項があります。

対象タグ: base、meta、link、hr、br、img、input、param、area、col

- 画面モードで編集しないでください。
- アウトラインビューやプロパティビューの表示は、正しく表示されないことがあります。
- メニューやタグパレットで挿入されるタグの一部は、XHTML形式に対応していません。

5.12.4 JavaScriptについて

Ajaxフレームワークアプリケーションを開発する際は、JavaScriptに関して、以下の点に注意してください。

- windowオブジェクトのlocationに格納されている情報は、将来的に、Ajaxフレームワークの内部で利用する可能性があります。アプリケーションで利用する場合は、参照だけを行い、値は変更しないように注意してください。

付録A　環境定義ファイル

本付録では、Ajaxフレームワーク環境定義ファイル(以降、環境定義ファイルと略します)について説明します。

A.1　環境定義ファイルの概要

環境定義ファイルを利用して、通信フレームワークの動作を設定します。

環境定義ファイルの形式とファイル名

環境定義ファイルは、XML形式で記述します。
以下に、環境定義ファイルのデフォルトのファイル名および格納場所を示します。

```
[コンテキストルート]/WEB-INF/acf.xml
```

環境定義ファイルのファイル名は、web.xmlでのサーブレットのパラメーター設定で変更できます。変更する場合は、rcf.acfConfigパラメーターに、コンテキストルートを基点とした相対パスで指定します。
以下に、環境定義ファイルのファイル名を「acf01.xml」に変更する場合の設定例を示します。

```
<context-param>
  <param-name>rcf.acfConfig</param-name>
  <param-value>/WEB-INF/conf/acf01.xml</param-value>
</context-param>
```

環境定義ファイルの分割

環境定義ファイルは、分割して格納することができます。分割する場合は、acf.xmlという名前のフォルダ、またはrcf.acfConfigで指定された名前のフォルダを作成して、その中に分割した環境定義ファイルを格納してください。

以下に、環境定義ファイルを分割する場合の注意事項を示します。

- 分割された環境定義ファイルは、それぞれ単体で環境定義ファイルとして利用できなければなりません。すなわち、環境定義ファイルの一部を切り出して、別ファイルとすることはできません。
- データBean IDなど、環境設定内で一意となる必要がある値は、分割された環境定義ファイルが格納されているフォルダ内で一意でなければなりません。
- 分割された環境定義ファイルを格納するフォルダ内に、環境定義ファイル以外のファイルやサブフォルダを格納することはできません。
- 環境定義ファイルは、初回利用時に読み込まれます。アプリケーション動作中の動的変更はできません。
- 環境定義ファイルの分割個数に、制限はありません。ただし、環境定義ファイルの読込み時間は、環境定義ファイルの個数に正比例して増加します。このため、分割すると、環境定義ファイル使用時の性能に影響します。
  環境定義ファイル使用時の性能が著しく遅くなる場合には、業務や環境定義ファイルの単位で環境定義ファイルの構成を見直してください。

環境定義ファイルのエラー

環境定義ファイルが正しくない場合、Ajaxフレームワークのサーブレットは起動しません。

環境定義ファイルに関するエラーメッセージは、「J.3.2 環境定義ファイルに関するメッセージ」を参照してください。

A.2　環境定義ファイルの要素

以下に、環境定義ファイルの構成を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<acfConfig xmlns="http://interstage.fujitsu.com/schemas/rcf/acf">

  <version>9.0</version>
```

```
  <dataBeans>
    <!-- データBeanの設定 -->
    <dataBean>
      <!-- データBean ID -->
      <dataBeanId>データBean ID</dataBeanId>

      <!-- データBeanのクラス名 -->
      <className>Javaクラス名</className>

      <!-- スコープ -->
      <scope>スコープ名</scope>
    </dataBean>
    <dataBean> ... </dataBean>
  </dataBeans>

  <!-- データ型変換の設定 -->
  <conversion>
    <!-- コンバータの定義（複数回参照のための） -->
    <defConverter>
      <converterId>コンバータ・インスタンスの参照名</converterId>
      <converter>コンバータのインスタンス</converter>
    </defConverter>
    <defConverter> ... </defConverter>

    <!-- コンバータ選択ルール -->
    <conversionRule>
      <type>Javaの型名</type>
      <converterId>コンバータ・インスタンスの参照名</converterId>
    </conversionRule>
    <conversionRule> ... </conversionRule>
  </conversion>

</acfConfig>
```

以下の表に、環境定義ファイルに指定できる要素を示します。

表A.1 環境定義ファイルの要素

要素名	説明	省略	複数指定
acfConfig	環境定義ファイルの開始と終了を定義します。 詳細は、「A.3 環境定義ファイルの開始と終了(acfConfig)」を参照してください。	×	×
version	バージョン情報を定義します。 詳細は、「A.4 バージョンの定義(version)」を参照してください。	×	×
dataBeans	データBean情報を定義します。 詳細は、「A.5 データBeanの定義(dataBeans)」を参照してください。 なお、サーブレット連携機能の場合は、データBean情報を定義する必要はありません。	○	×
conversion	データ型変換機能の設定を定義します。 詳細は、「A.6 コンバータ設定の定義(conversion)」を参照してください。	×	×

○: 可能、×: 不可能

これらの要素は、以下の名前空間URIに属するように記述します。ただし、以降の記述例では、名前空間の定義は省略しています。

```
http://interstage.fujitsu.com/schemas/rcf/acf
```

A.3 環境定義ファイルの開始と終了(acfConfig)

環境定義ファイルの開始と終了は、acfConfig要素で定義します。

記述形式

```
<acfConfig>
  ...
</acfConfig>
```

記述例

```
<?xml version="1.0" encoding="UTF-8" ?>
<acfConfig>
  <version>...</version>
   ...
</acfConfig>
```

A.4 バージョンの定義(version)

環境定義ファイルのバージョンは、version要素で定義します。

記述形式

```
<acfConfig>
  <version>...</version>
  ...
</acfConfig>
```

記述例

```
<?xml version="1.0" encoding="UTF-8" ?>
<acfConfig>
  <version>9.0</version>
   ...
</acfConfig>
```

注意

バージョンには、「9.0」を指定してください。それ以外の値は、無効になります。

A.5 データBeanの定義(dataBeans)

Apcoordinatorと連携する場合のデータBean情報は、dataBeans要素およびdataBean要素で定義します。
なお、サーブレット連携機能の場合は、データBean情報を定義する必要はありません。

- 記述形式
- 要素の内容
- 記述例

記述形式

```
<acfConfig>
  <dataBeans>
    <dataBean>
      <dataBeanId>データBean ID</dataBeanId>
```

```
      <className>Javaクラス名</className>
      <scope>スコープ名</scope>
      <recreate>再作成の有無</recreate>
    </dataBean>

    <dataBean>
     ...
    </dataBean>
   ...
  </dataBeans>
  ...
</acfConfig>
```

要素の内容

要素名	説明	省略	複数指定
dataBean	各データBeanの定義を指定します。	○	○
dataBean/dataBeanId	データBean ID(JavaScriptから指定する際の別名)を指定します。 指定できる値は、アルファベット(A-Z、a-z)および数字(0-9)の範囲で、先頭はアルファベットでなければなりません。	×	×
dataBean/className	データBeanのJavaクラス名を指定します。 指定するJavaクラスは、以下の条件を満たしている必要があります。 ・ WebアプリケーションのCLASSPATHに含まれるクラスであること ・ パブリックのデフォルトコンストラクタを持つこと ・ Ajaxフレームワークが提供するクラス以外であること ・ com.fujitsu.uji.DataBeanクラスを継承していること ・ JavaBeans形式であること	×	×
dataBean/scope	データBeanの有効範囲(スコープ)を、以下のどちらかの文字列で指定します。 ・ session HTTPセション内でデータを共有する ・ request リクエスト内でデータを共有する 省略した場合は、「request」になります。	○	×
dataBean/recreate	セションタイムアウトが発生した場合に、データBeanを再作成するかどうかを指定します。 ・ true データBeanを再作成する ・ false データBeanを再作成しない 本要素は、dataBean/scopeに「session」を指定した場合に有効です。 省略した場合は、「false」になります。	○	×

○: 可能、×: 不可能

記述例

```
<acfConfig>
  ...
  <dataBeans>
    <dataBean>
      <dataBeanId>HelloBean</dataBeanId>
      <className>samples.HelloBean</className>
      <scope>session</scope>
      <recreate>true</recreate>
    </dataBean>
  </dataBeans>

</acfConfig>
```

A.6 コンバータ設定の定義(conversion)

データ型変換機能で利用するコンバータ設定は、conversion要素で定義します。

データ型変換機能についての詳細は、「3.6 データ型変換機能」を参照してください。

- 記述形式
- 要素の内容
- コンバータ固有の設定項目
- 記述例

記述形式

```
<acfConfig>
  ...
  <conversion>
    <!-- コンバータの定義（複数回参照のための）-->
    <defConverter>
      <converterId>コンバータ・インスタンスの参照名</converterId>
      <[converter]>コンバータのインスタンス</[converter]>
    </defConverter>
    <defConverter>...</defConverter>

    <!-- コンバータ選択ルール -->
    <conversionRule>
      <type>データBeanの型名</type>
      <converterId>コンバータ・インスタンスの参照名</converterId>
    </conversionRule>
    <conversionRule>...</conversionRule>
  </conversion>
  ...
</acfConfig>
```

要素の内容

要素名	説明	子要素(注)
conversion	トップレベルの要素 すべての記述が含まれます。	(defConverter)* (conversionRule)*
defConverter	コンバータのインスタンスに名前を付けます。 同一コンバータを複数回参照することができます。 converterIdは参照するための名前を表し、[converter]が対応する定義を表します。	converterId [converter]

要素名	説明	子要素(注)
conversionRule	コンバータ選択のルールを1つ追加します。 状況(type)に対して、コンバータのインスタンス([converter]またはconverterId)を対応付けます。	type ([converter] | converterId)
type	Javaの型名(データBeanまたはJavaBeansの型名)を表します。	Java型名を表す文字列
[converter] (各コンバータ名の先頭を小文字にしたもの)	コンバータの定義を表します。 子要素の組合せは、コンバータによって異なります。詳細は、「コンバータ固有の設定項目」を参照してください。	(concreteType)? (child)*
converterId	コンバータの参照を表します。	コンバータのインスタンスの名前を表す文字列
concreteType	コンバータが生成する具象型名を表します。	Java型名を表す文字列
child	propertyに対応する子オブジェクト用の情報を指定します。 typeには、子オブジェクトの型を指定します。 [converter]またはconverterIdには、コンバータを指定します。	property (type)? ([converter] | converterId)?
property	プロパティ名を表します。	プロパティ名を表す文字列

注) 子要素の表記について、以下に説明します。

・A B: Aのあとに Bを記述します。
・(A|B): AまたはBのどちらかを記述します。
・(A)*: Aを1回以上繰り返して記述するか、全く記述しません。
・(A)?: Aを1回記述するか、全く記述しません。

コンバータ固有の設定項目

コンバータ固有の設定項目には、concreteTypeとchildの2種類があります。

以下の表に、各コンバータに子要素として設定可能な項目を示します。

コンバータ名(注)	concreteType		child	
	設定	省略値	設定	省略値
BooleanConverter	△	boolean, Boolean	×	−
ByteConverter	△	byte, Byte	×	−
ShortConverter	△	short, Short	×	−
IntegerConverter	△	int, Integer	×	−
LongConverter	△	long, Long	×	−
FloatConverter	△	float, Float	×	−
DoubleConverter	△	double, Double	×	−
CharacterConverter	△	char, Character	×	−
StringConverter	△	String	×	−
DateConverter	△	java.util.Date	×	−
BigIntegerConverter	△	java.math.BigInteger	×	−

コンバータ名(注)	concreteType		child	
	設定	省略値	設定	省略値
BigDecimalConverter	△	java.math.Decimal	×	−
ArrayConverter	○	ユーザー指定の配列型	○	property要素とtype要素の設定が必須。 [converter]またはconverterIdによるコンバータの指定はできません。
CollectionConverter	△	java.util.ArrayList	○	property要素とtype要素の設定が必須。 [converter]またはconverterIdによるコンバータの指定はできません。
MapConverter	△	java.util.HashMap	○	property要素とtype要素の設定が必須。 [converter]またはconverterIdによるコンバータの指定はできません。
BeanConverter	○	ユーザー指定のJavaBeanクラス	△	各プロパティの型と対応するコンバータ

○: 必須、△: 任意、×: 不可
注) 要素名は、コンバータ名の先頭を小文字にしたものです。

以下の表に、child要素を必要とするコンバータのchild要素の数、およびchild要素のpropertyに指定する値を示します。

コンバータ名	childの数	childのpropertyの値
ArrayConverter	1	"component"
CollectionConverter	1	"element"
MapConverter	1	"value"
BeanConverter	0～n(注)	JavaBeanに定義されているプロパティ名

注) nは、JavaBeanに定義されているプロパティの数です。

以下に、配列の要素がint型の、ArrayConverterの記述例を示します。

```
<conversionRule>
  <type>int[]</type>
  <arrayConverter>
    <concreteType>int[]</concreteType>
    <child>
      <property>component</property>
      <type>int</type>
    </child>
  </arrayConverter>
</conversionRule>
```

記述例

Java型情報の例

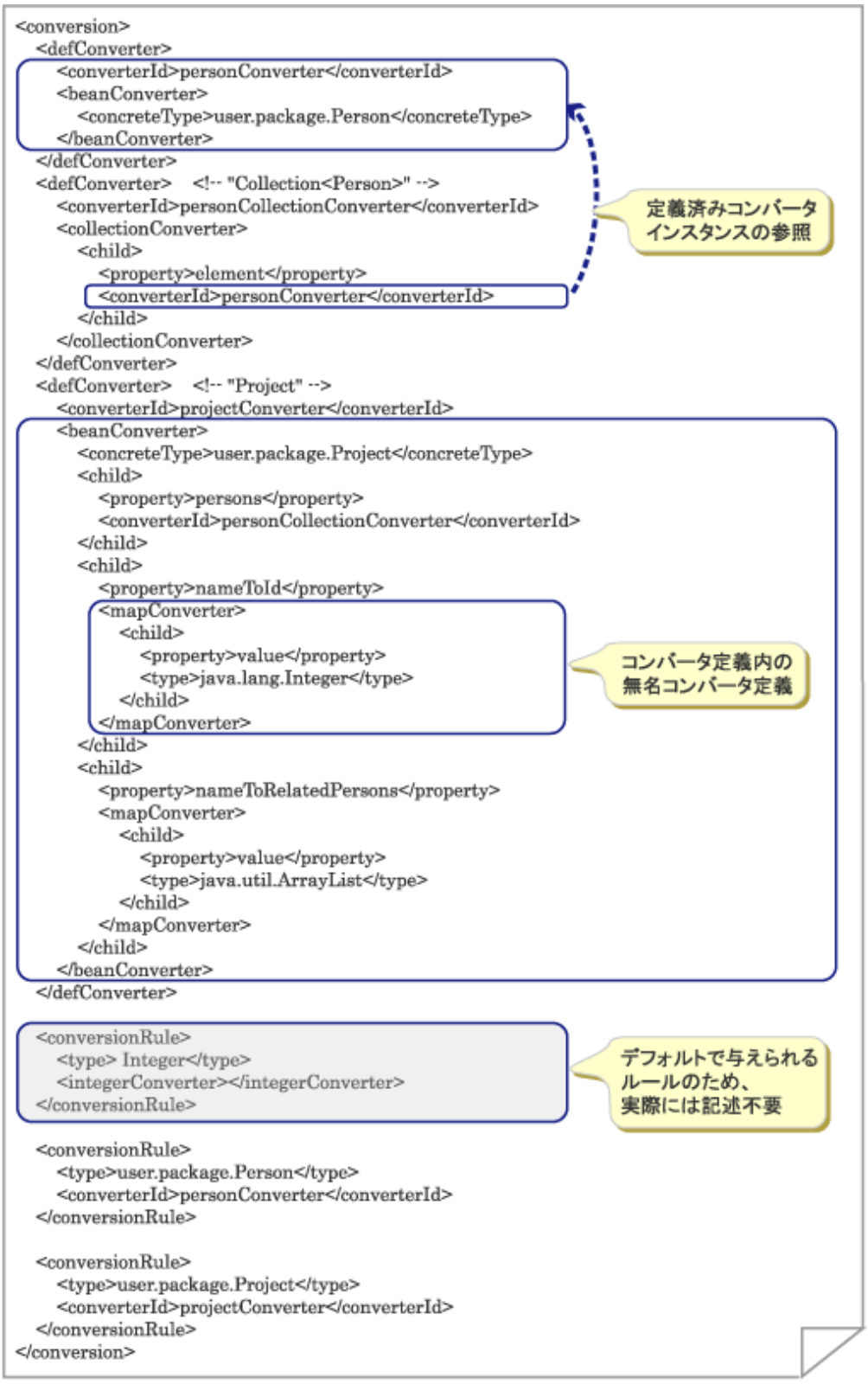
以下の例では、publicなどの修飾詞やsetter/getterを省略しています。
また、Java 5のgenericの記法を利用していますが、Java 1.4でも、設定の記述方法やコンバータの動作は変わりません。

```
package user.package;

// RPC 公開サービス
class ProjectBuilder {
    // 設定が不要な例
```

```
    int getNumberOfProjects();

    // 引数にコレクションが現れる例
    Project build(Person leader, Collection<Person> persons);

    // 戻り値にコレクションが現れる例
    Collection<Project> getProjects();
}

// 単純な JavaBean
class Person {
    String name;
    Date birthday;
}

// 複雑な JavaBean
class Project {
    String name;
    Person leader;
    List<Person> persons;
    Map<String, Integer> nameToId;
    Map<String, Collection<Person>> nameToRelatedPersons;
}
```

コンバータ設定の例

上記の例のようなJavaクラスをデータ型変換機能で変換するには、環境定義ファイルに以下のように記述します。




```
<conversion>
  <defConverter>
    <converterId>personConverter</converterId>
    <beanConverter>
      <concreteType>user.package.Person</concreteType>
    </beanConverter>
  </defConverter>
  <defConverter>   <!-- "Collection<Person>" -->
    <converterId>personCollectionConverter</converterId>
    <collectionConverter>
      <child>
        <property>element</property>
        <converterId>personConverter</converterId>
      </child>
    </collectionConverter>
  </defConverter>
  <defConverter>   <!-- "Project" -->
    <converterId>projectConverter</converterId>
    <beanConverter>
      <concreteType>user.package.Project</concreteType>
      <child>
        <property>persons</property>
        <converterId>personCollectionConverter</converterId>
      </child>
      <child>
        <property>nameToId</property>
        <mapConverter>
          <child>
            <property>value</property>
            <type>java.lang.Integer</type>
          </child>
        </mapConverter>
      </child>
      <child>
        <property>nameToRelatedPersons</property>
        <mapConverter>
          <child>
            <property>value</property>
            <type>java.util.ArrayList</type>
          </child>
        </mapConverter>
      </child>
    </beanConverter>
  </defConverter>

  <conversionRule>
    <type> Integer</type>
    <integerConverter></integerConverter>
  </conversionRule>

  <conversionRule>
    <type>user.package.Person</type>
    <converterId>personConverter</converterId>
  </conversionRule>

  <conversionRule>
    <type>user.package.Project</type>
    <converterId>projectConverter</converterId>
  </conversionRule>
</conversion>
```

付録B　マッシュアップ定義ファイル

本付録では、マッシュアップ定義ファイルについて説明します。

B.1　マッシュアップ定義ファイルの概要

マッシュアップ定義ファイルを利用して、マッシュアップフレームワークの動作を設定します。

マッシュアップ定義ファイルの形式とファイル名

マッシュアップ定義ファイルは、XML形式で記述します。
以下に、マッシュアップ定義ファイルのデフォルトのファイル名および格納場所を示します。

```
[コンテキストルート]/WEB-INF/conf/muf.xml
```

マッシュアップ定義ファイルのエラー

マッシュアップ定義ファイルが正しくない場合、プロキシサーブレットは起動しません。

マッシュアップ定義ファイルに関するエラーメッセージは、「J.3.2 環境定義ファイルに関するメッセージ”を参照してください。

B.2　マッシュアップ定義ファイルの要素

以下に、マッシュアップ定義ファイルの構成を示します。

```
<?xml version="1.0" encoding="UTF-8"?>

<muf_property>

    <muf_services>
        <muf_service> サービスの定義 </muf_service>
    </muf_services>

    <muf_admin>
        <security> セキュリティの定義 </security>
        <network> プロキシの定義 </network>
        <log> ログの定義 </log>
    </muf_admin>

</muf_property>
```

以下の表に、マッシュアップ定義ファイルに指定できる要素を示します。

表B.1 マッシュアップ定義ファイルの要素

要素名	説明	省略	複数指定
muf_property	マッシュアップ定義ファイルの開始と終了を定義します。 詳細は、「B.3 マッシュアップ定義ファイルの開始と終了(muf_property)」を参照してください。	×	×
muf_services	サービス情報を定義します。 詳細は、「B.4 サービスの定義(muf_services)」を参照してください。	×	×
muf_admin	運用情報を定義します。 詳細は、「B.5 運用の定義(muf_admin)」を参照してください。	×	×

×: 不可能

B.3 マッシュアップ定義ファイルの開始と終了(muf_property)

マッシュアップ定義ファイルの開始と終了は、muf_property要素で定義します。

記述形式

```
<muf_property>
  ...
</muf_property>
```

記述例

```
<?xml version="1.0" encoding="UTF-8" ?>
<muf_property>
  <muf_services>...</muf_services>
   ...
</muf_property>
```

B.4 サービスの定義(muf_services)

マッシュアップフレームワークがアクセスするサービスは、muf_services要素およびmuf_service要素で定義します。

記述形式

```
<muf_property>
  <muf_services>
    <muf_service name="サービスID">
      <adaptor name="アダプタ名"/>
      <parameters>
        <parameter name="キー" value="値" />
        <parameter name="キー" value="値" />
        ...
      </parameters>
    </muf_service>
    <muf_service>
     ...
    </muf_service>
    ...
  </muf_services>
  ...
</muf_property>
```

要素の内容

要素名	説明	省略	複数指定
muf_service	各サービスの定義を指定します。 name属性に、サービスを識別する名前(サービスID)を指定します。すべてのサービスの中で一意な名前を指定します。	○	○
adaptor	サービスで使用するアダプタを指定します。 name属性に、以下のどちらかのアダプタ名を指定します。 ・ htmladaptor: HTMLアダプタ ・ restadaptor: RESTアダプタ	×	×
parameters	アダプタに受け渡すパラメーターを子要素parameterで指定します。	×	×

要素名	説明	省略	複数指定
parameter	アダプタに受け渡すパラメーターをひとつずつ指定します。 name属性にキーを、value属性に値を記述します。 記述内容については、「4.7 アダプタ」を参照してください。	×	○

○: 可能、×: 不可能

注意

各要素の入力制限については、「5.7.11 マッシュアップ定義ファイルの編集」を参照してください。

記述例

```
<muf_property>
  <muf_services>
    <muf_service name="REST_AJAXSOFTWARE">
      <adaptor name="restadaptor"/>
      <parameters>
        <parameter name="URL" value="http://restserver/listrest.cgi?product=A" />
        <parameter name="XSL" value="listrest.xsl" />
      </parameters>
    </muf_service>
    <muf_service>
     ...
    </muf_service>
    ...
  </muf_services>
  ...
</muf_property>
```

B.5 運用の定義(muf_admin)

マッシュアップフレームワークの運用情報は、muf_admin要素で定義します。

記述形式

```
<muf_property>
  ...
  <muf_admin>
    <security> セキュリティの定義 </security>
    <network> プロキシの定義 </network>
    <log> ログの定義 </log>
  </muf_admin>

</muf_property>
```

要素の内容

要素名	説明	省略	複数指定
security	セキュリティ情報を定義します。 詳細は、「B.5.1 セキュリティの定義(security)」を参照してください。	×	×
network	プロキシ情報を定義します。 詳細は、「B.5.2 プロキシの定義(network)」を参照してください。	×	×

要素名	説明	省略	複数指定
log	ログ情報を定義します。 詳細は、「B.5.3 ログの定義(log)」を参照してください。	×	×

×: 不可能

B.5.1 セキュリティの定義(security)

マッシュアップフレームワークのセキュリティは、security要素で定義します。

記述形式

```
<muf_property>
  ...
  <muf_admin>
    <security>
      <authentication>
        <class name="クラス名"/>
      </authentication>
    </security>
    ...
  </muf_admin>

</muf_property>
```

要素の内容

要素名	説明	省略	複数指定
authentication	リクエストの正当性をチェックする情報を子要素classで指定します。	×	×
class	リクエストの正当性をチェックするのに使用するクラス名をname属性に指定します。	×	×

×: 不可能

注意

クラス名は、「com.fujitsu.interstage.rcf.mashup.auth.SessionChecker」固定です。変更しないでください。

記述例

```
<muf_property>
  ...
  <muf_admin>
    <security>
      <authentication>
        <class name="com.fujitsu.interstage.rcf.mashup.auth.SessionChecker"/>
      </authentication>
    </security>
    ...
  </muf_admin>

</muf_property>
```

B.5.2 プロキシの定義(network)

マッシュアップフレームワークで使用するプロキシは、network要素で定義します。

記述形式

```
<muf_property>
  ...
  <muf_admin>
    ...
    <network>
      <proxy>
        <http name="プロキシサーバのアドレス" port="ポート番号"/>
        <https name="プロキシサーバのアドレス" port="ポート番号"/>
        <excludes>
          <exclude>プロキシサーバを使用しないアドレス</exclude>
          <exclude>プロキシサーバを使用しないアドレス</exclude>
          ...
        </excludes>
      </proxy>
      <threadpool>
        <thread 通信で使用する管理情報 />
      </threadpool>
    </network>
    ...
  </muf_admin>

</muf_property>
```

要素の内容

要素名	説明	省略	複数指定
proxy	プロキシの情報を子要素http、https、excludesで指定します。	×	×
http	HTTP通信で使用するプロキシサーバを指定します。 name属性にアドレスを、port属性にポート番号を記述します。	○	×
https	HTTPS通信で使用するプロキシサーバを指定します。 name属性にアドレスを、port属性にポート番号を記述します。	○	×
excludes	ローカルのサーバなど、プロキシの対象外としたいアドレスを子要素excludeで指定します。	○	×
exclude	プロキシサーバを使用しないアドレスを指定します。 最大128個まで定義できます。	×	○
threadpool	サービスとの通信で使用する管理情報が子要素threadで設定されています。	×	×
thread	通信で使用する管理情報が設定されています。	×	×

○: 可能、×: 不可能

注意

各要素の入力制限については、「5.7.11 マッシュアップ定義ファイルの編集」を参照してください。

記述例

```
<muf_property>
  ...
  <muf_admin>
    ...
```

```
    <network>
      <proxy>
        <http name="proxy.soft.fujitsu.com" port="8080"/>
        <https name="proxy.soft.fujitsu.com" port="8080"/>
        <excludes>
          <exclude>*.fujitsu.co.jp</exclude>
          <exclude>*.fujitsu.com</exclude>
          <exclude>10.*</exclude>
        </excludes>
      </proxy>
      <threadpool>
        <thread init="30" max="150" timeout="300" />
      </threadpool>
    </network>
    ...
  </muf_admin>

</muf_property>
```

注意

thread要素の値は固定です。変更しないでください。

B.5.3 ログの定義(log)

マッシュアップフレームワークのログの出力情報は、log要素で定義します。

記述形式

```
<muf_property>
  ...
  <muf_admin>
    ...
     <log>
       <accesslog valid="出力の有無">
         <output path="出力先パス" prefix="ファイル名" extension="ファイル拡張子" />
         <rotation daily="日付分割の有無" />
         <rotation_size maxsize="分割サイズ" backup="分割数" />
       </accesslog>
       <systemlog level="ログレベル">
         <output path="出力先パス" prefix="ファイル名" extension="ファイル拡張子" />
         <rotation daily="日付分割の有無" />
         <rotation_size maxsize="分割サイズ" backup="分割数" />
       </systemlog>
     </log>
    ...
  </muf_admin>

</muf_property>
```

要素の内容

要素名	説明	省略	複数指定
accesslog	アクセスログの出力情報を指定します。 valid属性にログの出力の有無を指定します。 ・ true 出力する	×	×

要素名	説明	省略	複数指定
	・ false 出力しない		
accesslog/output	アクセスログの出力先を指定します。 path属性に出力先パスを、prefix属性にファイル名を、extension属性にファイル拡張子を指定します。 出力先パスは、ディレクトリ分割記号に「/」を使用して、絶対パス形式で記述します。出力先パスが「/」で終了していない場合は、「/」が自動的に付加されます。なお、出力先パスに「[contextroot]」を指定すると、コンテキストルートが参照されます。 ファイル名とファイル拡張子は、シングルバイト文字列で記述します。	×	×
accesslog/rotation	アクセスログファイルの日付分割の有無をdaily属性に指定します。 ・ true サイズ分割指定に加えて日付分割を行う ・ false 日付分割を行わない	×	×
accesslog/rotation_size	maxsize属性にアクセスログファイルの分割サイズ(Mバイト単位)を、backup属性にアクセスログファイルの最大分割数を指定します。	×	×
systemlog	システムログの出力情報を指定します。 level属性にログの出力レベルを以下の数値で指定します。 ・ 5 ERRORレベルのログだけを出力 ・ 10 ERRORおよびWARNレベルのログを出力 ・ 20 ERROR、WARN、およびINFOレベルのログを出力	×	×
systemlog/output	システムログの出力先を指定します。 path属性に出力先パスを、prefix属性にファイル名を、extension属性にファイル拡張子を指定します。 出力先パスは、ディレクトリ分割記号に「/」を使用して、絶対パス形式で記述します。出力先パスが「/」で終了していない場合は、「/」が自動的に付加されます。なお、出力先パスに「[contextroot]」を指定すると、コンテキストルートが参照されます。 ファイル名とファイル拡張子は、シングルバイト文字列で記述します。	×	×
systemlog/rotation	システムログファイルの日付分割の有無をdaily属性に指定します。 ・ true サイズ分割指定に加えて日付分割を行う ・ false 日付分割を行わない	×	×
systemlog/rotation_size	maxsize属性にシステムログファイルの分割サイズ(Mバイト単位)を、backup属性にシステムログファイルの最大分割数を指定します。	×	×

×:不可能

記述例

```
<muf_property>
  ...
  <muf_admin>
    ...
     <log>
       <accesslog valid="true">
         <output path="[contextroot]/WEB-INF/log/" prefix="muacclog" extension="txt" />
         <rotation daily="true" />
         <rotation_size maxsize="10" backup="10" />
       </accesslog>
       <systemlog level="5">
         <output path="[contextroot]/WEB-INF/log/" prefix="musyslog" extension="txt" />
         <rotation  daily="false" />
         <rotation_size maxsize="10" backup="10" />
       </systemlog>
     </log>
    ...
  </muf_admin>

</muf_property>
```

ログファイル名

以下に、出力されるログファイル名の書式を示します。

```
[ログファイル名][日付][連番][拡張子]
```

以下に、各項目について説明します。

項目	説明
ログファイル名	output要素のprefix属性に指定したファイル名が設定されます。
日付	rotation要素のdaily属性にtrueを指定した場合に、「YYYYMMDD」の形式で設定されます。 rotation要素のdaily属性にfalseを指定した場合は省略されます。
連番	連番は0から始まり、rotation_size要素のmaxsize属性に指定した分割サイズに従って、backup属性に指定した最大分割数まで設定されます。
拡張子	output要素のextension属性に指定したファイル拡張子が設定されます。

付録C　JavaScript API一覧

Ajaxフレームワークアプリケーションで利用可能なJavaScript APIの一覧を以下に示します。

JavaScript API	説明	関連記事
RCF.addInitializedListener	画面の初期化直後に実行されるイベントリスナを登録します。	画面初期表示時のユーザーロジックの呼出し
RCF.addLoadedListener	画面の初期化前に実行されるイベントリスナを登録します。	画面初期表示時のユーザーロジックの呼出し
RCF.addErrorListener	画面でエラーが発生した場合に実行されるイベントリスナを登録します。	UI部品でエラーが発生した場合の処理
RCF.debug	デバッグ文を出力する場合に利用します。	デバッグ機能
RCF.info		
RCF.warn		
RCF.error		
RCF.getComponent	APIを経由してモデルオブジェクトやUI部品のオブジェクトを取得する場合に利用します。	「UI部品リファレンス」の「JavaScriptからの操作」
RCF.setFocus	ユーザーロジックからUI部品にフォーカスを設定する場合に利用します。	RCF.setFocus関数を使用したフォーカスの設定
RCF.setQueryString	URLに付加するクエリ文字列を連想配列で指定します。	JavaScript APIによってクエリ文字列を付加する方法
RCF.addQueryString	URLに付加するクエリ文字列のキー名と値を追加します。	JavaScript APIによってクエリ文字列を付加する方法
RCF.encodeURL	パラメーターで指定されたURLに、RCF_configオブジェクトのjsessionidで指定されたセションIDを付加して返します。	JavaScript APIによって任意のURLにセションIDを付加する方法
RCF.createQueryString	RCF_configオブジェクトに設定されたクエリ文字列を、URLに付加可能な形式に変換して返します。	JavaScript APIによって任意のURLにクエリ文字列を付加する方法
AcfEventLog.trace	イベントログをサーバ側に出力するためのAPIです。	イベントログAPI
AcfEventLog.info		
AcfEventLog.warn		
AcfEventLog.error		
UjiRequest.send	Apcoordinator連携機能でビジネスロジックを呼び出すためのAPIです。	ビジネスメソッドの実行
rcf.event.EventRegistrar.registerEvents	イベントを登録するためのAPIです。	イベントリスナの登録
rcf.event.EventRegistrar.unregisterEvents	イベントを破棄するためのAPIです。	イベントリスナの登録

JavaScript API	説明	関連記事
rcf.event.ActionEvent.target	ActionEventは、画面部品が操作されたときに発生するイベントです。	アクションイベント
rcf.event.ActionEvent.type		
rcf.event.ActionEvent.browserEvent		
rcf.event.BrowserEvent.target	BrowserEventは、ブラウザ上で発生したイベントをラップしているオブジェクトです。	アクションイベント
rcf.event.BrowserEvent.type		
rcf.event.BrowserEvent.nativeEvent		
rcf.event.PropertyChangeEvent.target	PropertyChangeEventは、モデルの値が変更されたときに発生するイベントです。	プロパティチェンジイベント
rcf.event.PropertyChangeEvent.type		
rcf.event.PropertyChangeEvent.propertyName		
rcf.event.PropertyChangeEvent.oldValue		
rcf.event.PropertyChangeEvent.newValue		
MuRequest.send	マッシュアッププロキシを経由して、WebサービスにアクセスするためのAPIです。	クライアント通信API
RcfObjectConverter.encode	指定されたオブジェクトを文字列に変換するAPIです。	ビジネスロジックの実行
RcfRequest.send	サーブレット連携機能でビジネスロジックを呼び出すためのAPIです。 通信は非同期で行います。	ビジネスロジックの実行(非同期通信)
RcfRequest.invoke	サーブレット連携機能でビジネスロジックを呼び出すためのAPIです。 通信は同期で行います。	ビジネスロジックの実行(同期通信)

付録D　サンプルアプリケーション

Ajaxフレームワークを利用したサンプルとして、以下のアプリケーションが提供されています。

各サンプルアプリケーションは、Interstage StudioまたはEclipseのプロジェクト形式で格納されています。

Interstage Studioワークベンチ用サンプルアプリケーション

- helloAjaxサンプルアプリケーション
  入力した文字列をエコーバックする、Ajaxフレームワークの入門的なアプリケーションです。
  UI部品、モデル、通信フレームワークを利用したApcoordinator連携の方法などを確認することができます。

- helloAjaxGeneralサンプルアプリケーション
  通信フレームワークのサーブレット連携(汎用通信方式)を利用したアプリケーションです。
  動作は、helloAjaxサンプルアプリケーションと同じです。

- helloAjaxSimpleサンプルアプリケーション
  通信フレームワークのサーブレット連携(簡易通信方式)を利用したアプリケーションです。
  動作は、helloAjaxサンプルアプリケーションと同じです。

  注意

  簡易通信方式(同期通信)は、Internet Explorerのみで利用できます。

- orderサンプルアプリケーション
  受注業務を想定したアプリケーションです。
  FragmentContainerの利用方法や、通信フレームワークを利用したApcoordinator連携の方法などを確認することができます。

- uiSetサンプルアプリケーション
  画面部品を列挙したアプリケーションです。
  各画面部品の表示例を確認することができます。

- ajaxSampleサンプルアプリケーション
  入力条件に従って検索結果を表示するアプリケーションです。
  「5.9 アプリケーションの開発例」で使用しています。

- eventLogサンプルアプリケーション
  イベントログを出力するアプリケーションです。
  イベントログの使用方法を確認することができます。

- uiSet2サンプルアプリケーション
  機能付加部品を使用したアプリケーションです。
  入力フィールドへの入力値制限、入力値の検証機能、グループ化機能などの使用例を確認することができます。

- sampleDataGridサンプルアプリケーション
  DataGrid部品の使い方を紹介するサンプル集です。
  テーブル内にチェックボックスやツリーなどを表示する場合の使用例を確認することができます。

- contextMenuサンプルアプリケーション
  ContextMenu部品を使用したアプリケーションです。
  項目によってContextMenuを変更する方法などが確認できます。

Eclipse用サンプルアプリケーション

- transportサンプルアプリケーション
  運送依頼画面に社員検索アプリケーションのユーザー情報を集約するアプリケーションです。
  HTMLアダプタを利用したマッシュアップフレームワークの使用例を確認することができます。
  以下のログイン情報でログインできます。

  ー　ID: user1 ～ user5

  ー　パスワード: なし

- salesサンプルアプリケーション
  各営業所の売上情報を集約するアプリケーションです。
  RESTアダプタを利用したマッシュアップフレームワークの使用例を確認することができます。

- calendarサンプルアプリケーション
  利用者のカレンダを表示するアプリケーションです。
  「5.10 アプリケーションの開発例(マッシュアップ)」で使用しています。
  以下のログイン情報でログインできます。

  ー ID: gid001 ～ gid005

  ー パスワード: 001 ～ 005

- muServicesサンプルアプリケーション
  各種Webサービスをシミュレートするアプリケーションです。
  transportサンプルアプリケーション、salesサンプルアプリケーション、calendarサンプルアプリケーションで使用しています。

- orderサンプルアプリケーション
  受注業務を想定したアプリケーションです。

- uiSetサンプルアプリケーション
  画面部品を列挙したアプリケーションです。
  各画面部品の表示例を確認することができます。

- uiSet2サンプルアプリケーション
  機能付加部品を使用したアプリケーションです。
  入力フィールドへの入力値制限、入力値の検証機能、グループ化機能などの使用例を確認することができます。

サンプルアプリケーションの格納場所

Ajaxフレームワークを利用したサンプルアプリケーションは、開発環境パッケージをインストールした場合、以下のフォルダにインストールされます。

Interstage Studioワークベンチ用サンプルアプリケーションの場合

```
C:¥InteractionManager¥sample¥Studio_JavaEE
```

Eclipse用サンプルアプリケーションの場合

```
C:¥InteractionManager¥sample¥Eclipse
```

C:¥InteractionManagerは標準のインストールフォルダです。製品のインストールフォルダを変更した場合は、読み替えてください。

サンプルアプリケーションの利用方法

以下の手順で利用できます。

Interstage Studioワークベンチの場合

1. Interstage Studioワークベンチを起動します。
   起動時の環境設定でサンプルアプリケーション用のワークスペースを指定し、そのワークスペースでInterstage Studioワークベンチを起動します。
2. Interstage Studioワークベンチのメニューで、[ファイル] > [インポート]を選択します。
3. [インポート]ウィザードで、[一般] > [既存プロジェクトをワークスペースへ]を選択し、[次へ]ボタンをクリックします。
4. [プロジェクトのインポート]ページで、[ルートフォルダの選択]にサンプルアプリケーションの格納場所を指定し、[プロジェクトをワークスペースにコピー]をチェックして、[終了]ボタンをクリックします。
5. Interstage Studioワークベンチでプロジェクトをビルドし、デバッグを実行すると、サンプルアプリケーションの動作を確認することができます。
   サンプルアプリケーションのビルド時に警告が表示される場合がありますが、問題はありません。
   プロジェクトのビルドやデバッグの方法については、「5.7 Interstage Studioワークベンチを利用した開発」を参照してください。

注意

Interstage Studioワークベンチ用のサンプルアプリケーションは、Interstage Studio V10.0向けに作成されています。
Interstage Studio V10.0より新しいバージョン・レベルで使用する場合は、以下の手順を実施してからビルドしてください。

1. Interstage Studioワークベンチのメニューで、[ウィンドウ] > [設定]を選択します。
2. [設定]ダイアログボックスで、[サーバ] > [ランタイム環境]を選択します。
3. [サーバランタイム環境]ページで[追加]ボタンをクリックします。
4. [新規サーバランタイム環境]ダイアログボックスで、[ランタイム環境のタイプを選択]の一覧から「FUJITSU LIMITED」の「Interstage Application Server V10.0 IJServer Cluster (JavaEE)」を選択し、[完了]ボタンをクリックします。
5. [OK]ボタンをクリックします。

Eclipseの場合

1. Eclipseを起動します。
   起動時の環境設定でサンプルアプリケーション用のワークスペースを指定し、そのワークスペースでEclipseを起動します。
2. Eclipseのメニューで、[File] > [Import]を選択します。
3. [Import]ウィザードで、[General] > [Existing Projects into Workspace]を選択し、[Next]ボタンをクリックします。
4. [Import Projects]ページで、[Select root directory]にサンプルアプリケーションの格納場所を指定し、[Copy projects into workspace]をチェックして、[Finish]ボタンをクリックします。
5. Eclipseでプロジェクトをビルドし、デバッグを実行すると、サンプルアプリケーションの動作を確認することができます。
   サンプルアプリケーションのビルド時に警告が表示される場合がありますが、問題はありません。
   プロジェクトのビルドやデバッグの方法については、「5.8 Eclipseを利用した開発」を参照してください。

ポイント

Interstage Studioのプロジェクトを Eclipseにインポートして使用することも可能です。
プロジェクトの移行については、「付録I 開発資産の移行」を参照してください。

注意

- 以下のサンプルアプリケーションを実行するには、muServicesサンプルアプリケーションをあらかじめ起動しておく必要があります。また、muServicesの実行環境のホスト名がlocalhost以外、またはポート番号が80以外の場合、以下のサンプルアプリケーションに設定されているサービス定義を編集する必要があります。サービス定義の編集方法については、「5.7.11 マッシュアップ定義ファイルの編集」を参照してください。ホスト名またはポート番号の指定に誤りがある場合は、エラーメッセージ(RCF0805)が出力されます。
  - transport
  - sales
  - calendar
- サンプルプログラムはInteraction Managerを利用したプログラム開発を理解するために利用してください。
  Interaction Managerを実運用する場合には、セキュリティなどを考慮した設計を行ってください。

参考

Ajaxフレームワークを利用したJ2EEのサンプルアプリケーションは、以下のフォルダにインストールされます。

```
C:¥InteractionManager¥sample¥Studio
```

C:¥InteractionManagerは標準のインストールフォルダです。製品のインストールフォルダを変更した場合は、読み替えてください。

利用方法については、「Interstage Studio ユーザーズガイド」を参照してください。

- サンプルアプリケーション
  インストールされるサンプルアプリケーションは以下のとおりです。概要については、「Interstage Studioワークベンチ用サンプルアプリケーション」を参照してください。
  - helloAjaxサンプルアプリケーション
  - helloAjaxGeneralサンプルアプリケーション
  - helloAjaxSimpleサンプルアプリケーション
  - orderサンプルアプリケーション
  - uiSetサンプルアプリケーション
  - ajaxSampleサンプルアプリケーション
  - eventLogサンプルアプリケーション
  - uiSet2サンプルアプリケーション
  - sampleDataGridサンプルアプリケーション
  - contextMenuサンプルアプリケーション

付録E　モックアップの作成

アプリケーションサーバや開発環境がない場合でも、Ajaxフレームワークを利用した画面モックアップを開発することができます。この場合は、開発環境パッケージに含まれている以下のフォルダを、HTMLファイルと同じフォルダにコピーしてください。

```
C:¥InteractionManager¥rcf¥js¥acf
```

C:¥InteractionManagerは標準のインストールフォルダです。製品のインストールフォルダを変更した場合は、読み替えてください。

この場合、Ajaxフレームワークを利用するための宣言方法は、通常の開発方法と同様です。詳細は、「2.3 Ajaxフレームワークの宣言」を参照してください。

開発した画面モックアップを実行する場合は、「5.5 開発環境の設定」に従って、ブラウザのオプションを設定してください。

注意

モックアップ用のJavaScriptファイルには、通信フレームワークおよびマッシュアップフレームワークに関するものは含まれていません。「5.6 実行環境の設定」を参照のうえ、クラスパスに以下のライブラリを設定してください。

- 通信フレームワークの機能を利用する場合
  - ujircf.jar
  - uji.jar
- マッシュアップフレームワークの機能を利用する場合
  - ujircf.jar
  - muf.jar
  - Tidy.jar
  - uji.jar

また、運用時には、必ず上記のライブラリを使用してください。

付録F　Ajaxページエディタ

本付録では、Ajaxページエディタについて説明します。
Ajaxページエディタは、EclipseおよびInterstage Studioワークベンチのプラグインとして動作します。

注意

本付録では、Eclipseを使用した場合の操作方法を説明しています。Interstage Studioワークベンチを使用する場合は、本付録に記載されている英語のメニュー名やコマンド名などを日本語の名称に読み替えてください。

F.1　Ajaxページエディタの概要

Ajaxページエディタは、Ajaxフレームワークアプリケーションで利用する画面をGUIベースで編集するツールです。

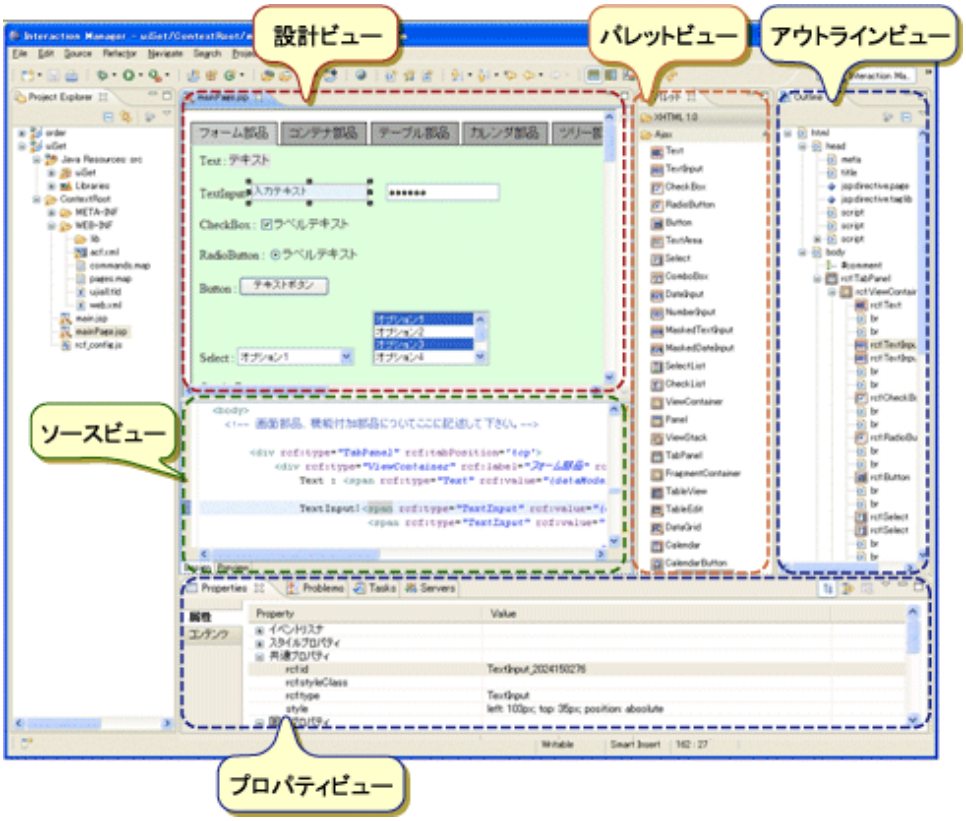
Ajaxページエディタには、以下のような特長があります。

- 実行イメージを確認しながら、WYSIWYGで業務画面を編集することができる
  - 画面のレイアウトを、実際に表示する画面イメージで表示する
  - 部品の位置や大きさを、実際に表示する画面イメージで編集できる
  - 属性やCSSの設定に従って画面を表示する
  - UI部品と同様に、HTMLタグも実際に表示する画面イメージで編集できる
- ビジュアル編集機能と連動したテキスト編集を行うことができる
- 部品から[イベント処理定義]ダイアログボックスを起動し、部品に対する操作発生時の処理を入力できる
- 部品から[モデルバインディング定義]ダイアログボックスを起動し、モデルや画面部品とのデータバインディングを指定できる

F.1.1　ビューの概要

以下に、Ajaxページエディタのビューの概要を説明します。

図F.1 Ajaxページエディタのビュー



- 設計ビュー
  UI部品を使用した画面をWYSIWYGで編集するビューです。
  以下の特長があります。
  - UI部品を、実行時エンジン(JavaScript)で表示する
  - 絶対座標を使用した編集が可能
  - パレットからマウスで選択した部品を配置
  - 選択した部品をマウスで移動、リサイズすることが可能
  - UI部品、HTMLタグに対してクリップボードを使用した編集操作が可能
  - 編集結果は、ソースビュー、プロパティビュー、アウトラインビューと連動
- ソースビュー
  UI部品を使用した画面のソースコードを編集するビューです。
  以下の機能を提供します。
  - UI部品の定義に従った入力支援機能
  - UI部品の定義に従ってバリデータの警告を抑止
  - 編集結果を、設計ビュー、プロパティビュー、アウトラインビューに反映
- パレットビュー
  HTMLタグおよびUI部品をパレットに表示して、設計ビュー上に配置するためのビューです。
- アウトラインビュー
  編集中の画面のタグの構造を表示するビューです。
  UI部品は、部品の定義構造を表示します。UI部品以外は、XHTMLの構造を表示します。

- プロパティビュー
  設計ビューで選択されたUI部品、ソースビューでキャレットのあるタグおよびUI部品、アウトラインビューで選択されたタグおよびUI部品のプロパティを表示するビューです。以下のように、タブによって表示内容を切り替えることができます。

  ー [属性]タブ: 選択されたタグや部品のプロパティの属性が表示されます。

  ー [コンテンツ]タブ : コンテンツ編集エリアに切り替わり、選択されたタグや部品のコンテンツ内容が表示されます。

- **プレビュー**
  エディタのタブにより、設計ビュー/ソースビューとプレビューを切り分けて表示することができます。

  ー [Design]タブ: 設計ビューとソースビューが表示されます。

  ー [Preview]タブ: プレビューが表示されます。

  プレビューでは、Internet ExplorerのIEコンポーネントを使用して、実行時の画面表示およびブラウザで動作するイベント処理を確認することができます。

注意

プレビューで使用するJavaScriptファイルには、通信フレームワークおよびマッシュアップフレームワークに関するものは含まれていません。
「5.6 実行環境の設定」を参照のうえ、クラスパスに以下のライブラリを設定してください。

ー 通信フレームワークの機能を利用する場合

- ujircf.jar
- uji.jar

ー マッシュアップフレームワークの機能を利用する場合

- ujircf.jar
- muf.jar
- Tidy.jar
- uji.jar

ポイント

ビュー間の連携
ビュー間では、データが同期します。それぞれのビューで編集操作により変更があった場合は、ほかのビューにもその変更が反映されます。ただし、ソースビュー編集中の自動反映を行わない場合は、ソースビューの変更は設計ビューに反映されません。ソースビュー編集中の自動反映については、「F.5.1 設計ビューのコンテキストメニュー」を参照してください。
また、設計ビューで複数部品を選択して操作する場合、選択状態はほかのビューと同期しません。

F.1.2 編集可能なファイル

Ajaxページエディタで編集できるファイルは、以下の条件を満たす、HTML/JSPファイルです。

- XHTML 1.0に適合している
- 以下のDOCTYPE宣言が記述されている

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- Ajaxフレームワークの動作定義(rcf_config.js)を使用している
- Ajaxフレームワークの初期化処理(rcf.js)を使用している

なお、rcf_config.jsおよびrcf.jsが指定されていないHTML/JSPファイル(FragmentContainer部品で使用するファイルなど)を編集する場合は、ファイルのプロパティから設定する必要があります。詳細は、「F.2.1 画面フォームの修正」を参照してください。

注意

- Ajaxフレームワークの動作定義(rcf_config.js)は、Ajaxフレームワークの初期化処理(rcf.js)よりも先に記述しなければなりません。
- サブフォルダに配置されているHTML/JSPファイルを直接表示する場合、srcプロパティで指定するrcf_config.jsおよびrcf.jsについて、パスの考慮が必要です。アプリケーションフォルダの直下に存在する「rcf_config.js」、およびアプリケーションフォルダを基点とした「acf/file/rcf/rcf.js」が参照できるように相対パスで指定してください。
  アプリケーションフォルダ直下のJSPフォルダにJSPファイルが存在する場合の記述例を以下に示します。

```
<script type="text/javascript" src="../rcf_config.js"></script>
<script type="text/javascript" src="../acf/file/rcf/rcf.js"></script>
```

  HTML/JSPファイルをインクルードしている場合には、インクルード元のHTML/JSPファイルが存在するフォルダを基点とした相対パスで指定してください。
- 画面の初期化、画面のロード、画面上でのエラー発生、タイムアウトを契機に動作するように指定したJavaScriptのイベント処理は、画面編集時にも実行されます。
  以下のような場合、処理の結果によって設計ビューの編集操作ができなくなることがあります。
  - JavaScriptで動的にUI部品やHTMLタグを追加/削除/プロパティ変更した場合、Ajaxページエディタで認識されないため、編集対象に含まれない場合があります。
  - キーボード操作を変更/抑止した場合、キーボードによる編集操作ができません。
  - マウス操作を変更/抑止した場合、マウスによる編集操作ができません。

  これらに該当するスクリプトを記述する場合は、設計ビューでの画面編集後に、ソースビューで記述してください。

F.1.3 使用するブラウザ

Ajaxページエディタの設計ビュー機能およびプレビュー機能は、Internet ExplorerのIEコンポーネントで動作します。Ajaxページエディタを使用するためには、Internet Explorerをインストールしてください。
ブラウザの設定については、「5.5 開発環境の設定」を参照してください。

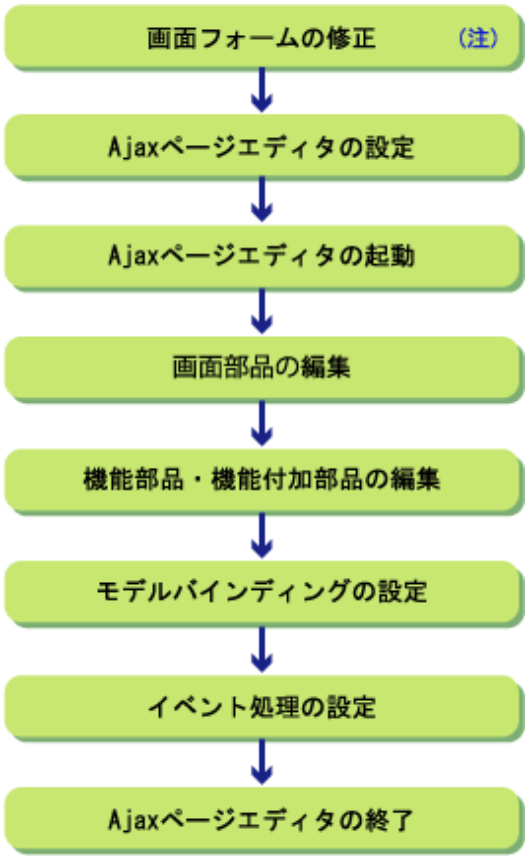
設計ビューおよびプレビューでは、編集するHTML/JSPファイルは、インストールされているInternet Explorerで表示されます。
ほかのブラウザの表示については、以下のどちらかの方法で確認してください。

- 該当のブラウザでモックアップ機能を使用して表示を確認する
- アプリケーションを実行し、該当のブラウザを使用して表示を確認する

F.2 Ajaxページエディタを利用した開発手順

Ajaxページエディタを利用してAjaxフレームワークアプリケーションを開発する場合、以下の手順で開発します。

図F.2 Ajaxページエディタを利用した開発手順



注) [Ajax JSP]ウィザード以外で作成した画面フォーム(HTML/JSPファイル)を、Ajaxページエディタで編集する場合にだけ必要です。

1. 画面フォームの修正
   [Ajax JSP]ウィザード以外で作成した画面フォーム(HTML/JSPファイル)を、Ajaxページエディタで編集する場合は、画面フォームの修正が必要になります。詳細は、「F.2.1 画面フォームの修正」を参照してください。

2. Ajaxページエディタの設定
   編集するファイルごとに、Ajaxページエディタの動作オプションを設定します。詳細は、「F.2.2 Ajaxページエディタの設定」を参照してください。

3. Ajaxページエディタの起動
   以下の方法で、Ajaxページエディタを起動します。
   HTML/JSPファイルのコンテキストメニューから[Open]を選択します。または、[Open With]を選択して[Ajaxページエディタ]を選択します。

4. 画面部品の編集
   Ajaxページエディタの各ビューを使用して、画面部品を配置し、位置、大きさ、その他のプロパティを編集します。
   詳細は、使用するビューに応じて、「F.5 設計ビュー」、「F.6 ソースビュー」、「F.7 パレットビュー」、「F.8 アウトラインビュー」、「F.9 プロパティビュー」を参照してください。

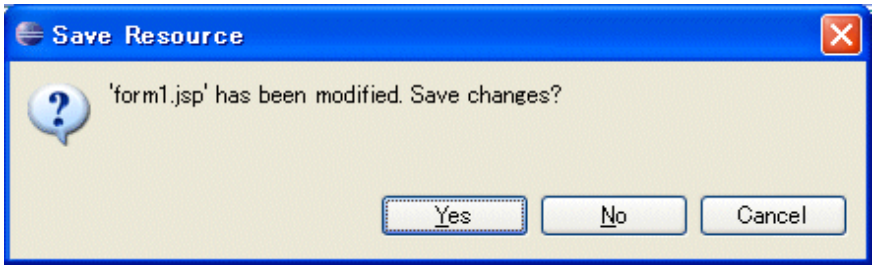
5. 機能部品・機能付加部品の編集
   ソースビューを使用して、機能部品(モデル)および機能付加部品を編集します。
   ソースビューの詳細は、「F.6 ソースビュー」を参照してください。

6. モデルバインディングの設定
   モデルバインディング定義機能を使用して、画面部品のプロパティに、モデルやほかの画面部品のプロパティを割り当てます。
   詳細は、「F.10.1 モデルバインディング定義」を参照してください。

7. イベント処理の設定
   イベント処理定義機能を使用して、画面部品のイベントに処理を割り当てます。

ソースビューを使用して、イベントに対する処理を記述します。
詳細は、「F.10.2 イベント処理定義」、「F.6 ソースビュー」を参照してください。

8. Ajaxページエディタの終了
エディタを閉じると、Ajaxページエディタは終了します。
エディタの終了時に編集内容が保存されていない場合は、以下のメッセージボックスが表示されるので、編集内容を保存するかどうかを選択します。



F.2.1 画面フォームの修正

[Ajax JSP]ウィザード以外で作成した画面フォーム(HTML/JSPファイル)を、Ajaxページエディタで編集する場合は、以下の修正が必要になります。

- 単独で利用するHTML/JSPファイルの場合
HTML/JSPファイルに、以下を記述します。

  ー DOCTYPE宣言

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

  ー <script>タグ

```
<script type="text/javascript" src="rcf_config.js"></script>
<script type="text/javascript" src="acf/file/rcf/rcf.js"></script>
```

- FragmentContainer部品などで使用するHTML/JSPファイルの場合
ファイルのコンテキストメニューから[プロパティ]を選択し、[Ajaxページエディタ設定]ページで、以下を設定します。

  ー [編集時に指定したCSSファイル、JavaScriptファイルを読み込む]をチェックします。

  ー [編集時に読み込むJavaScriptファイル]に「rcf_config.js」および「acf/file/rcf/rcf.js」を指定します。

  設定方法の詳細は、「F.2.2 Ajaxページエディタの設定」を参照してください。

注意

- Ajaxフレームワークの動作定義(rcf_config.js)は、Ajaxフレームワークの初期化処理(rcf.js)よりも先に記述しなければなりません。
- サブフォルダに配置されているHTML/JSPファイルを直接表示する場合、srcプロパティで指定するrcf_config.jsおよびrcf.jsについて、パスの考慮が必要です。アプリケーションフォルダの直下に存在する「rcf_config.js」、およびアプリケーションフォルダを基点とした「acf/file/rcf/rcf.js」が参照できるように相対パスで指定してください。
アプリケーションフォルダ直下のJSPフォルダにJSPファイルが存在する場合の記述例を以下に示します。

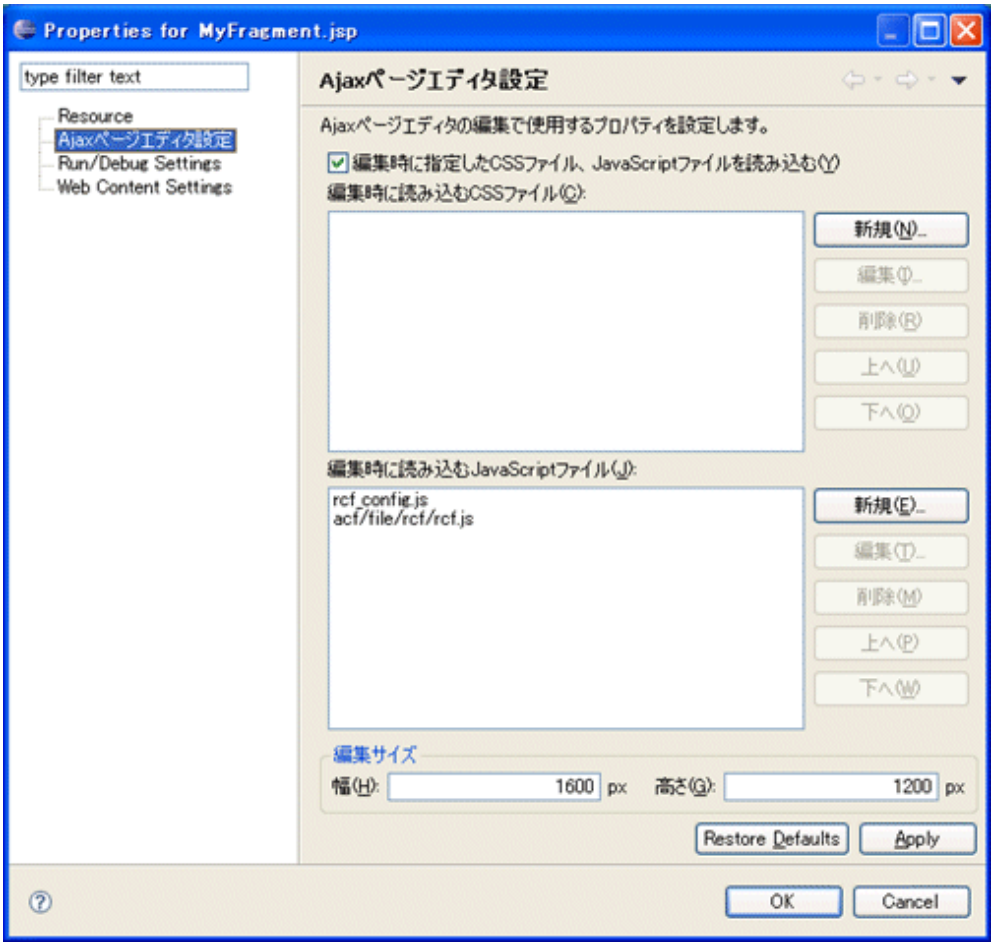
```
<script type="text/javascript" src="../rcf_config.js"></script>
<script type="text/javascript" src="../acf/file/rcf/rcf.js"></script>
```

  HTML/JSPファイルをインクルードしている場合には、インクルード元のHTML/JSPファイルが存在するフォルダを基点とした相対パスで指定してください。

F.2.2　Ajaxページエディタの設定

[Ajaxページエディタ設定]ページを利用して、編集するファイルごとに、Ajaxページエディタの動作オプションを設定します。
[Ajaxページエディタ設定]ページは、[Project Explorer]ビューでファイルのコンテキストメニューから[Properties]を選択すると表示されます。

以下に、[Ajaxページエディタ設定]ページを示します。



以下の表に、[Ajaxページエディタ設定]ページの項目を説明します。

項目	説明
編集時に指定したCSSファイル、JavaScriptファイルを読み込む	FragmentContainer内のHTMLを編集する場合にチェックします。 チェックすると、指定したCSSファイル、JavaScriptファイルを編集時に読み込みます。
編集時に読み込むCSSファイル	編集時に読み込む外部スタイルシートファイルを指定します。 参照するファイルは、新規に追加、削除、順序の変更をすることができます。
編集時に読み込むJavaScriptファイル	編集時に読み込む外部JavaScriptファイルを指定します。 参照するファイルは、新規に追加、削除、順序の変更をすることができます。
編集サイズ	選択したファイルに対して、Ajaxページエディタで編集できるサイズ(幅と高さ)をピクセルで指定します。Ajaxページエディタの設計ビューでは、指定したサイズの範囲内でUI部品を編集することができます。
Restore Defaults	デフォルトの設定に戻します。デフォルトは何も指定しないため、指定したすべてのファイルが削除されます。
Apply	指定した内容を適用します。

[Ajaxページエディタ設定]ページで、[編集時に指定したCSSファイル、JavaScriptファイルを読み込む]をチェックして、CSSファイルとJavaScriptファイルを指定すると、編集するHTML/JSPファイルには、編集時に以下のようなタグが付加されます。そのため、エディタを開くと、ファイル名に未保存を示すアスタリスクが表示されます。
なお、付加されたタグは編集中に使用されるだけで、ファイルには保存されません。

- ファイルの先頭に付加されるタグ

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
 xmlns:rcf="http://interstage.fujitsu.com/iim/ns/rcf">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=<<編集ファイルの文字コード>>" />

    <link rel="stylesheet" type="text/css" href="<<指定したcssファイル1>>" />
    <link rel="stylesheet" type="text/css" href="<<指定したcssファイル2>>" />
    (省略)

    <script type="text/javascript" src="<<指定したJavaScriptファイル1>>"></script>
    <script type="text/javascript" src="<<指定したJavaScriptファイル2>>"></script>
    (省略)
  </head>
  <body>
```

- ファイルの最後に付加されるタグ

```
  </body>
</html>
```

注意

- [編集時に指定したCSSファイル、JavaScriptファイルを読み込む]をチェックして編集する場合、文字コードは、以下の方法で変更してください。
  編集中のファイルのmetaタグ、pageディレクティブに指定された文字コード(charset)をすべて修正して保存してください。このとき、Ajaxページエディタで自動的に付加されたmetaタグの文字コードも修正してください。

  ー metaタグで文字コードを指定する例

  ```
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  ```

  ー pageディレクティブで文字コードを指定する例

  ```
  <%@page contentType="text/html; charset=utf-8"%>
  ```

- 編集対象のファイル種別がユーザーロジック定義ファイルでない場合、タグ構造が崩れてエラーが発生するため、[編集時に指定したCSSファイル、JavaScriptファイルを読み込む]はチェックしないでください。
  誤ってチェックした場合は、保存せずにエディタを閉じてからチェックを外してください。

F.3 Ajaxページエディタのメニュー

ここでは、Ajaxページエディタのメニューコマンドについて説明します。

F.3.1 [File]メニューのコマンド

Ajaxページエディタの[File]メニューのコマンドを以下の表に示します。

コマンド名	機能	キーボードショートカット
New	新規作成ウィザードを起動します。	Alt + Shift + N

コマンド名	機能	キーボードショートカット
Open File	ダイアログボックスを表示し、指定したファイルを開きます。	なし
Close	編集中のファイルを閉じます。	Ctrl + W
Close All	開いているファイルをすべて閉じます。	Ctrl + Shift + W
Save	編集中のファイルを保存します。	Ctrl + S
Save As	編集中のファイルを別名で保存します。	なし
Save All	開いているファイルをすべて保存します。	Ctrl + Shift + S
Revert	編集中のファイルを、前回保存した状態に戻します。	なし
Move	ナビゲータで選択したノードを移動します。	なし
Rename	ナビゲータで選択したノードの名前を変更します。	なし
Refresh	ナビゲータの表示を最新の状態に更新します。	F5
Convert Line Delimiters To	編集中のファイルの行内区切り文字を、以下の中から選択した文字に変換します。 ・ Windows ・ Unix ・ Mac OS 9	なし
Print	編集中のファイルを印刷します。	Ctrl + P
Switch Workspace	ワークスペースを切り替えます。	なし
Restart	ワークベンチをリスタートします。	なし
Import	ワークスペースに開発資産をインポートします。	なし
Export	ワークスペースの開発資産をエクスポートします。	なし
Properties	ナビゲータで選択したノードまたは編集中のファイルのプロパティを表示します。	Alt + Enter
Exit	Eclipseを終了します。	なし

なお、利用できる[File]メニューのコマンドは、フォーカスがあるビューによって、以下のように異なります。

コマンド名	フォーカスがあるビュー			
	設計ビュー	ソースビュー	アウトラインビュー	プロパティビュー
New	○	○	○	○
Open File	○	○	○	○
Close	○	○	○	○
Close All	○	○	○	○
Save	○	○	○	○
Save As	○	○	○	○
Save All	○	○	○	○
Revert	×	○	×	×
Move	×	×	×	×
Rename	×	×	×	×
Refresh	×	○	×	×
Convert Line Delimiters To	×	○	×	×
Print	×	○	×	×

コマンド名	フォーカスがあるビュー			
	設計ビュー	ソースビュー	アウトラインビュー	プロパティビュー
Switch Workspace	○	○	○	○
Restart	○	○	○	○
Import	○	○	○	○
Export	○	○	○	○
Properties	×	○	×	×
Exit	○	○	○	○

○:有効
×:無効

F.3.2 [Edit]メニューのコマンド

Ajaxページエディタの[Edit]メニューのコマンドを以下の表に示します。

コマンド名	機能	キーボードショートカット
Undo	HTML/JSPファイルに対する直前の変更を元に戻します。編集操作履歴がない場合は、編集操作は元に戻せません。	Ctrl + Z
Redo	HTML/JSPファイルに対する直前の変更を元に戻す操作を行った際、その操作を元に戻します。	Ctrl + Y
Cut	ソースビューで実行した場合は、選択した範囲を切り取ってテキストデータとしてクリップボードに保存します。設計ビューで実行した場合は、選択された部品を切り取り、テキストデータとしてクリップボードに保存します。	Ctrl + X
Copy	ソースビューで実行した場合は、選択した範囲をコピーしてテキストデータとしてクリップボードに保存します。設計ビューで実行した場合は、選択された部品をコピーし、テキストデータとしてクリップボードに保存します。	Ctrl + C
Paste	ソースビューで実行した場合は、クリップボードのデータをカーソル位置に貼り付けます。設計ビューで実行した場合は、クリップボードのデータを設計ビュー上に貼り付けて選択状態とします。	Ctrl + V
Delete	選択した範囲または要素を削除します。	Delete
Select All	Ajaxページエディタのソースビューの内容をすべて選択します。	Ctrl + A
Find/Replace	[検索/置換]ダイアログボックスを表示し、指定文字列を検索または置換します。	Ctrl + F
Find Next	現在選択されているテキスト、またはアウトラインビューで選択している要素名の次を検索します。	Ctrl + K
Find Previous	現在選択されているテキスト、またはアウトラインビューで選択している要素名の前を検索します。	Ctrl + Shift + K
Incremental Find Next	次をインクリメンタル検索モードで開始します。 ステータス・バーの指示に従って、検索テキストを入力してください。	Ctrl + J
Incremental Find Previous	前をインクリメンタル検索モードで開始します。 ステータス・バーの指示に従って、検索テキストを入力してください。	Ctrl + Shift + J

コマンド名	機能	キーボードショートカット
Add Bookmark	ブックマークを現在のテキスト選択または選択された要素に追加します。	なし
Add Task	ユーザー定義タスクを現在のテキスト選択または選択された要素に追加します。	なし
Content Assist	ソースビューのカーソル位置で使用可能な入力候補のリストを開きます。	Ctrl + Space
Smart Insert Mode	スマート挿入モードに設定します。	Ctrl + Shift + Insert
Expand Selection To	現在カーソルがある位置や、その前後のセレクタやプロパティを選択します。	なし
Show Tooltip Description	ツールチップ記述を表示します。	F2
Word Completion	ソースビューのカーソル位置で単語の入力を補完します。	Alt + /
Quick Fix	ソースビューのカーソル位置で単語のクイックフィックスを行います。	Ctrl + 1
更新	設計ビューの表示を最新の状態に更新します。 また、エラー発生により設計ビューで編集操作ができなくなった場合、ソースコードを修正したあとに更新する必要があります。	F5

なお、利用できる[Edit]メニューのコマンドは、フォーカスがあるビューによって、以下のように異なります。

コマンド名	フォーカスがあるビュー			
	設計ビュー	ソースビュー	アウトラインビュー	プロパティビュー
Undo	○	○	×	×
Redo	○	○	×	×
Cut	○	○	×	×
Copy	○	○	×	×
Paste	○	○	×	×
Delete	○	○	×	×
Select All	×	○	×	×
Find/Replace	×	○	×	×
Find Next	×	○	×	×
Find Previous	×	○	×	×
Incremental Find Next	×	○	×	×
Incremental Find Previous	×	○	×	×
Add Bookmark	×	○	×	×
Add Task	×	○	×	×
Content Assist	×	○	×	×
Smart Insert Mode	×	○	×	×
Expand Selection To	×	○	×	×
Show Tooltip Description	×	○	×	×
Word Completion	×	○	×	×
Quick Fix	×	○	×	×

コマンド名	フォーカスがあるビュー			
	設計ビュー	ソースビュー	アウトラインビュー	プロパティビュー
更新	○	○	×	×

○:有効
×:無効

F.3.3 [Source]メニューのコマンド

Ajaxページエディタの[Source]メニューのコマンドを以下の表に示します。

コマンド名	機能	キーボードショートカット
Toggle Comment	キャレットがある行、または、選択されている行をコメントにします。	Ctrl + Shift + C
Add Block Comment	キャレットがあるタグの開始タグから終了タグまでをコメントにします。	Ctrl + Shift + /
Remove Block Comment	ブロックコメントを削除します。	Ctrl + Shift + ¥
Shift Left	左側にインデントします。	なし
Shift Right	右側にインデントします。	なし
Cleanup Document	[Cleanup]ダイアログボックスを表示し、タグや属性などの大文字/小文字の統一など、規則に沿った文書に修正します。[Cleanup]ダイアログボックスについては、「[Cleanup]ダイアログボックス」を参照してください。	なし
Format	エディタの内容をフォーマットします。	Ctrl + Shift + F
Format Active Elements	アクティブ要素の内容をフォーマットします。	Ctrl + I
Occurrences in File	ファイル内での出現箇所を検索します。	Ctrl + Shift + A

なお、利用できる[Source]メニューのコマンドは、フォーカスがあるビューによって、以下のように異なります。

コマンド名	フォーカスがあるビュー			
	設計ビュー	ソースビュー	アウトラインビュー	プロパティビュー
Toggle Comment	×	○	×	×
Add Block Comment	×	○	×	×
Remove Block Comment	×	○	×	×
Shift Left	×	○	×	×
Shift Right	×	○	×	×
Cleanup Document	×	○	×	×
Format	×	○	×	×
Format Active Elements	×	○	×	×
Occurrences in File	×	○	×	×

○:有効
×:無効

[Cleanup]ダイアログボックス

[Cleanup]ダイアログボックスを利用すると、選択項目に従って、文書を一括で修正することができます。

以下に、[Cleanup]ダイアログボックスの選択項目を示します。

項目	説明
Tag name case for HTML	HTMLタグ名の大文字/小文字を以下から指定します。 ・ 現状のまま(As-is) ・ 小文字(Lower) ・ 大文字(Upper)
Attribute name case for HTML	HTML属性名の大文字/小文字を以下から指定します。 ・ 現状のまま(As-is) ・ 小文字(Lower) ・ 大文字(Upper)
Insert required attributes	必須となる属性値を挿入します。
Insert missing tags	欠落しているタグを挿入します。
Quote attribute values	引用符で囲まれていない属性値を引用符で囲みます。
Format source	文書をフォーマットします。
Convert line delimiters to	行内区切り文字を以下の中から選択した文字に変換します。 ・ Windows ・ Unix ・ Mac OS 9

F.4 Ajaxページエディタのツールバー

Ajaxページエディタでは、以下のツールバーを利用できます。

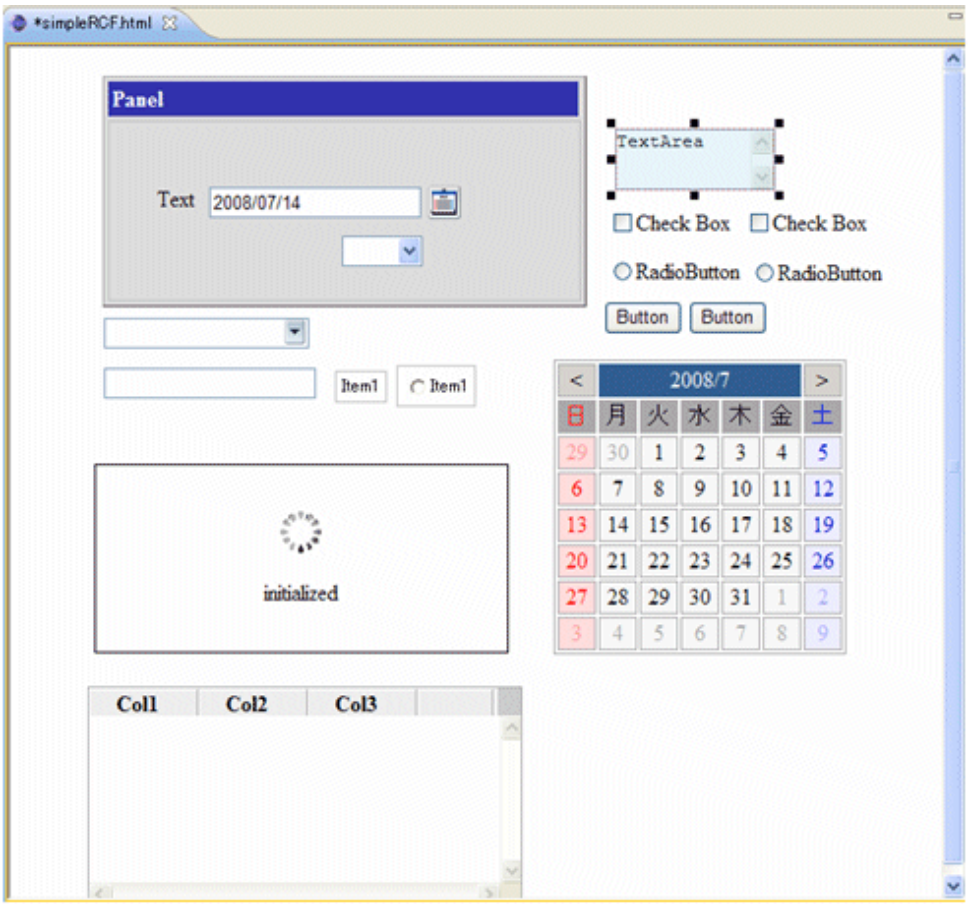
ボタン	ツールチップ	説明
	Ajaxページエディタを更新	設計ビューの表示を最新の状態に更新します。 また、エラー発生により設計ビューで編集操作ができなくなった場合、ソースコードを修正したあとに更新する必要があります。
	設計ビューとソースビューを上下に並べて表示	設計ビューとソースビューを上下に並べて表示します。
	設計ビューとソースビューを左右に並べて表示	設計ビューとソースビューを左右に並べて表示します。
	設計ビューのみを表示	設計ビューだけを表示します。
	ソースビューのみを表示	ソースビューだけを表示します。
	ソースビュー編集中に設計ビューを自動反映	ソースビュー編集中に設計ビューに自動反映するか、反映しないかを設定します。 自動反映する場合は、ボタンが押された状態で表示されます。

F.5　設計ビュー

設計ビューでは、HTMLタグおよび画面部品が実行時と同じイメージで表示されます。表示された部品を、部品単位で、選択、移動、リサイズ、削除を行うことができます。また、パレットビューで選択した部品を設計ビューに配置することができます。

以下に、設計ビューの画面例を示します。

図F.3 設計ビュー



注意

編集中のソースコードにエラーが存在する場合、設計ビューの編集操作ができません。その場合は、ソースビューでソースコードを修正してください。
また、ソースコードを修正後に編集操作が可能にならない場合は、ソースコードに問題がないことを確認してから、更新を実行してください。

F.5.1　設計ビューのコンテキストメニュー

設計ビューのコンテキストメニューのコマンドを以下の表に示します。

コマンド名	機能	キーボードショートカット
更新	設計ビューの表示を最新の状態に更新します。 また、エラー発生により設計ビューで編集操作ができなくなった場合、ソースコードを修正したあとに更新する必要があります。	F5
切り取り	選択した部品を切り取ってクリップボードに保存します。	Ctrl + X
コピー	選択した部品をコピーしてクリップボードに保存します。	Ctrl + C

コマンド名	機能	キーボードショートカット
貼り付け	クリップボードの内容を貼り付けます。 設計ビューで切り取りまたはコピーした場合に有効になります。	Ctrl + V
削除	選択した部品を削除します。	Delete
整列	部品の位置を揃えます。 [整列]コマンドには、サブメニューがあります。詳細は、「整列のサブメニュー」を参照してください。	なし
グリッド	グリッド表示を設定します。 [グリッド]コマンドには、サブメニューがあります。詳細は、「グリッドのサブメニュー」を参照してください。	なし
モデルバインディング定義	プロパティとモデルまたは画面部品のバインディングを定義します。	なし
イベント処理定義	画面部品にイベント処理を定義します。	なし
ソースビュー編集中の自動反映	ソースビュー編集中に設計ビューに自動反映するか、反映しないかを設定します。 自動反映する場合は、コマンド名の先頭にチェックマークが表示されます。	なし
プロパティ	プロパティビューを表示します。	なし

注意

編集ファイル内に右クリックを抑止するスクリプトがある場合、設計ビューのコンテキストメニューが表示されなくなります。

参考

表示/編集禁止状態の設計ビュー

ソースビュー編集中に設計ビューに自動反映しない設定の場合、ソースビューが編集されると、ソースコードと設計ビューの同期がとれなくなります。この時点で、設計ビューはグレー画面になり、表示および編集できない状態(表示/編集禁止状態)になります。
表示/編集禁止状態の設計ビューは、以下のタイミングで表示/編集禁止状態が解除され、最新の状態に更新されます。

- 設計ビューにフォーカスが当たった場合
- [Edit]メニューの[更新]コマンド、またはツールバーの(Ajaxページエディタを更新)をクリックした場合
- ツールバーの (ソースビュー編集中に設計ビューを自動反映)をクリックして自動反映する状態に変更した場合

なお、設計ビューが表示/編集禁止状態でも、[Preview]タブに切り替えた場合は、最新状態の画面が表示されます。

整列のサブメニュー

[整列]コマンドのサブメニューは、設計ビューで複数の部品を選択した場合にアクティブになります。
[整列]コマンドのサブメニューを以下の表に示します。

コマンド名	機能	キーボードショートカット
左揃え	複数部品を左に揃えます。	なし
左右中央揃え	複数部品を左右中央に揃えます。	なし
右揃え	複数部品を右に揃えます。	なし
上揃え	複数部品を上に揃えます。	なし
上下中央揃え	複数部品を上下中央に揃えます。	なし

コマンド名	機能	キーボードショートカット
下揃え	複数部品を下に揃えます。	なし

グリッドのサブメニュー

[グリッド]コマンドのサブメニューを以下の表に示します。

コマンド名	機能	キーボードショートカット
線を表示	設計ビューにグリッド線を表示します。	なし
線に吸着	部品を配置・移動したとき、配置座標を自動的にグリッド線にあわせます(注1)。	なし
指定間隔で吸着	部品を配置・移動したとき、配置座標を自動的にユーザー指定の間隔にあわせます(注1)。	なし

注1)[線に吸着]設定と[指定間隔で吸着]設定はどちらかしか選択できません。

F.5.2　設計ビューで編集可能な部品

設計ビューでは、HTMLタグと画面部品を表示、編集することができます。

注意

設計ビューでは、HTMLタグ、UI部品以外の要素の子要素は表示、編集できません。
例えば、以下のようにHTMLに定義されていない未知のタグ<unknownTag>や、uji:formタグの子要素は編集できません。

```
<unknownTag>
  <div rcf:type="TextInput" rcf:id="ti00001" rcf:left="15" rcf:top="36">
</unknownTag>

<uji:form name="input_form" method="post" verbs="ok" >
  <div rcf:id="text1" rcf:type="Text" rcf:value="商品名" ></div>
  <div rcf:id="input1" rcf:type="TextInput" ></div>
  <input type="SUBMIT" name="ok" value="決定" />
</uji:form>
```

HTMLタグ

以下の表に、設計ビューで表示、編集できるHTMLタグを示します。

部品名	タグ	WYSIWYG編集
Button	<input type="submit" />	○
CheckBox	<input type="checkbox" />	○
Form	<form></form>	○
Horizontal Rule	<hr />	○
Image	<img />	○
Image Button	<input type="image" />	○
Link	<a>	○
Password Field	<input type="password" />	○
Radio Button	<input type="radio" />	○
Select	<select></select>	○
Text Area	<textarea></textarea>	○

部品名	タグ	WYSIWYG編集
Text Field	<input />	○
Label	<label></label>	○
Div		○
Table	<table></table>	○(注1)

○:WYSIWYG編集が可能
注1) パレットからの配置はできません。ソースビューで追加してください。

画面部品

設計ビューでは、UI部品のうち、画面部品を表示、編集することができます。
機能部品と機能付加部品については、「F.10 クライアントフレームワーク連携編集」を参照してください。

以下に、設計ビューで表示、編集できる画面部品を示します。

- フォーム部品

部品名	WYSIWYG編集
Text	○(注2)
TextInput	○
CheckBox	○
RadioButton	○
Button	○
TextArea	○
Select	○
ComboBox	○
DateInput	○
NumberInput	○
MaskedTextInput	○
MaskedDateInput	○
SelectList	○
CheckList	○

○:WYSIWYG編集が可能
注2) Textはリサイズできません。

- コンテナ部品

部品名	WYSIWYG編集
ViewContainer	○
Panel	○
ViewStack	○
TabPanel	○
FragmentContainer	○
Window	○(注3)

○:WYSIWYG編集が可能
注3) Windowは、デフォルトでは不可視です。プロパティビューでWindowのshowWindowプロパティをtrueにし、Windowを表示させてから編集してください。また、パレットからの配置はできません。ソースビューで追加してください。

・ テーブル部品

部品名	WYSIWYG編集
TableView	○
TableEdit	○
DataGrid	○
ViewColumn	×
ViewColumnGrid	×
ViewColumnGroup	×
ViewColumnCheck	×
ViewColumnTree	×
ViewColumnSelect	×
ViewColumnImage	×

○:WYSIWYG編集が可能
×:WYSIWYG編集が不可能

・ カレンダ部品

部品名	WYSIWYG編集
Calendar	○
PopupCalendar	×(注4)
CalendarButton	○(注5)

○:WYSIWYG編集が可能
×:WYSIWYG編集が不可能
注4) PopupCalendarは編集時に表示、非表示を切り替える必要があるため、WYSIWYG編集はできません。なお、プロパティビューでのPopupCalendarの編集操作は有効です。
注5) CalendarButtonはリサイズできません。

・ ツリー部品

部品名	WYSIWYG編集
TreeView	○

○:WYSIWYG編集が可能

・ スクレイピング部品

部品名	WYSIWYG編集
ScrapingView	○

○:WYSIWYG編集が可能

・ メニュー部品

部品名	WYSIWYG編集
ContextMenu	×(注6)

×:WYSIWYG編集が不可能
注6) ContextMenuは実行時に、表示、非表示が動的に切り替わるため、WYSIWYG編集はできません。なお、プロパティビューでのContextMenuの編集操作は有効です。

JSPタグ

JSPタグは、設計ビューでは表示できません。JSPタグは、アウトラインビューで表示、選択し、プロパティビューで編集します。

F.5.3 部品の編集操作

設計ビューでの部品の編集操作には、部品共通の操作と、部品に固有の操作があります。

ここでは、部品共通の以下の操作について説明します。

- 配置
- 選択
- 移動
- 切り取り
- コピー
- 貼り付け
- 削除
- リサイズ
- 元に戻す
- やり直し
- 整列

部品に固有の操作については、部品に応じて、「F.5.4 コンテナ部品の編集操作」、「F.5.5 カレンダ部品の編集操作」、「F.5.6 ツリー部品の編集操作」を参照してください。

設計ビューで編集操作した場合、以下の表示位置およびサイズに関するプロパティの単位が「px」になります。

- style属性: top、left、width、height
- スタイルプロパティのサイズ: width、height
- Window部品のスタイルプロパティ: top、left

配置

設計ビューの部品を配置するには、以下の方法があります。

- パレットビューから設計ビュー領域内の任意の位置へ配置
  以下に、部品を配置する手順を示します。

  1. パレットビューから部品を選択します。
  2. 設計ビュー上で配置場所をマウスカーソルでポイントしてクリックします。
     配置場所が決定するまでは、部品枠が表示されるので、部品の大きさを確認することができます。

  パレットビューから設計ビューにUI部品が配置されると、部品ID(rcf:id属性)が自動的に付加されます。
  なお、パレットビューからの部品の選択を解除するには、ESCキーを押します。

パレットビューの詳細は、「F.7 パレットビュー」を参照してください。

なお、部品をコンテナ部品内に配置する場合は、「F.5.4 コンテナ部品の編集操作」を参照してください。

選択

設計ビューの部品を選択するには、以下の方法があります。

- 設計ビューの部品の選択可能領域上でクリック
- アウトラインビューで部品をクリック
- ソースビューで部品のタグ上にキャレットを置く

また、親要素が同じ部品を設計ビューで複数選択できます(Window部品を除きます)。
複数の部品を選択するには、以下の方法があります。

- 部品上でCtrl + クリック
  すでに選択状態になっている部品に加えて、クリックされた部品も選択状態になります。

なお、選択された部品は、選択枠で囲まれます。

移動

設計ビューの部品を移動するには、以下の方法があります。

- 部品を選択し、マウスによるドラッグ
- 部品を選択し、キーボード(カーソルキー)の押下

切り取り

設計ビューで選択状態の部品をクリップボードにコピーし、削除するには、以下の方法があります。

- 部品を選択し、コンテキストメニューから[切り取り]を選択
- 部品を選択し、キーボードでCtrl + Xを押下

コピー

設計ビューで選択状態の部品をクリップボードにコピーするには、以下の方法があります。

- 部品を選択し、コンテキストメニューから[コピー]を選択
- 部品を選択し、キーボードでCtrl + Cを押下

貼り付け

クリップボードにコピーされた部品を設計ビューに貼り付けるには、以下の方法があります。

- 設計ビューのコンテキストメニューから[貼り付け]を選択
- キーボードでCtrl + Vを押下

部品は、切り取りまたはコピーした部品の右下に貼り付けられます。
rcf:idが設定されている部品を貼り付けた場合、rcf:idに新しい値が設定されます。
なお、クリップボードが空の場合は、貼り付けできません。

削除

設計ビューから部品を削除するには、以下の方法があります。

- 選択部品のコンテキストメニューから[削除]を選択
- 部品を選択し、キーボードでDeleteキーを押下

なお、複数の部品が選択された場合は、選択状態の部品すべてが削除されます。子要素を持つ部品が選択された場合は、子要素も含めて削除されます。

リサイズ

設計ビューの部品をリサイズするには、以下の方法があります。

- リサイズ用マークのドラッグによるリサイズ
  部品を選択すると、矩形域の四隅と辺にリサイズ用のマークが表示されます。このマークをドラッグすることによって、部品をリサイズします。

コンテナ部品をリサイズした場合は、コンテナ部品の子要素の位置は、コンテナ部品の左上座標からの相対位置がリサイズ前後で変わりません。リサイズした結果、部品がコンテナ部品領域に含まれた場合は、その部品は子要素にはなりません。

元に戻す

設計ビューでの直前の操作を元に戻すには、以下の方法があります。

- [Edit]メニューから[Undo]を選択
- キーボードでCtrl + Zを押下

なお、元に戻せる操作は、配置、移動、切り取り、貼り付け、削除、リサイズ、整列です。
複数部品の操作を元に戻した場合、選択状態は元どおりになりません。ソースビューで部品のタグ上にキャレットのある部品が選択状態になります。

やり直し

設計ビューで元に戻した操作をやり直すには、以下の方法があります。

- [Edit]メニューから[Redo]を選択
- キーボードでCtrl + Yを押下

なお、やり直せる操作は、直前に元に戻した操作だけです。

整列

設計ビューで複数の部品を整列させるには、以下の方法があります。

- 2つ以上の部品を選択し、コンテキストメニューから[整列]の整列方法を選択

整列方法には、以下の種類があります。

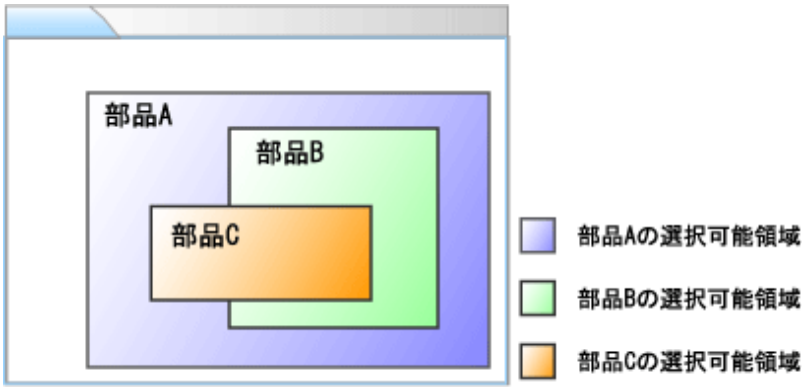
- 左揃え
- 左右中央揃え
- 右揃え
- 上揃え
- 上下中央揃え
- 下揃え

部品の選択可能領域

設計ビューで部品を選択するために、部品の選択可能領域があります。
設計ビューでUI部品が重なって配置された場合、各部品の表示されている部分が選択可能領域になります。選択可能領域をマウスでクリックすると、対象の部品が選択されます。

以下の図に、UI部品が重なって配置された場合の選択可能領域の例を示します。

図F.4 選択可能領域の例



F.5.4　コンテナ部品の編集操作

設計ビューで操作できる部品を子要素に持つ部品をコンテナ部品と呼びます。コンテナ部品には、以下のものがあります。

- 画面部品のコンテナ部品(FragmentContainerを除く)
- HTML formタグ

ここでは、コンテナ部品固有の操作について説明します。

コンテナ部品の移動

コンテナ部品の移動は、部品共通の操作と同様の方法で行います。詳細は、「移動」を参照してください。コンテナ部品の移動では、コンテナ部品上の部品(子ノード)も含めて移動します。

コンテナ部品内の編集

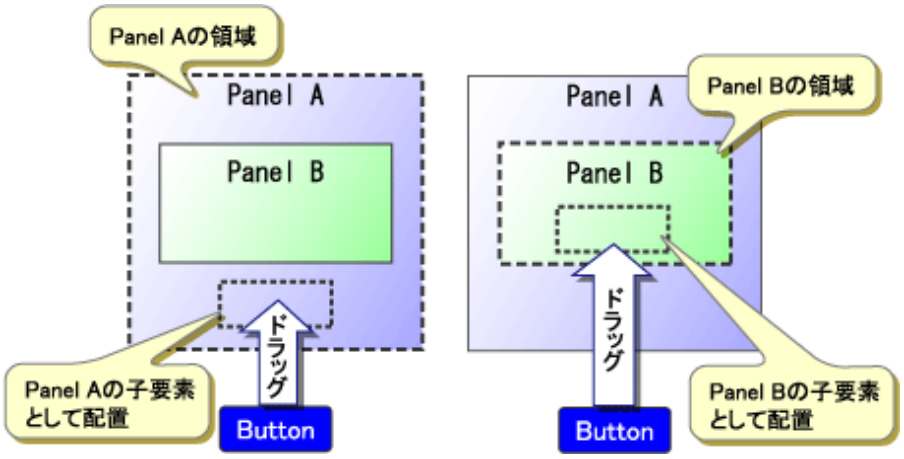
コンテナ部品内の部品の編集は、部品共通の操作と同様の方法で行います。詳細は、「F.5.3 部品の編集操作」を参照してください。

コンテナ部品内への部品配置・移動

コンテナ部品内に部品を追加するには、コンテナ部品の領域内に子要素として追加したい部品を配置、移動します。

以下の図に、コンテナ部品(Panel)にフォーム部品(Button)を追加する例を示します。

図F.5 コンテナ部品への子要素の追加



F.5.5　カレンダ部品の編集操作

カレンダ部品のCalendarとCalendarButtonの編集は、部品共通の操作と同様の方法で行います。詳細は、「F.5.3 部品の編集操作」を参照してください。

PopupCalendarはCalendarButtonと対で使用する部品です。CalendarButtonを配置すると、自動的にPopupCalendarも配置されます。

F.5.6　ツリー部品の編集操作

ツリー部品の編集は、部品共通の操作と同様の方法で行います。詳細は、「F.5.3 部品の編集操作」を参照してください。

ツリーのノードは、WYSIWYG編集はできません。TreeModelをバインディングすることによってモデルに従ったノードが表示されます。

ツリー部品のTreeViewは、TreeModelと対で使用する部品です。パレットからTreeViewを配置すると、TreeViewに対応付けられたTreeModelが自動的にソースコードに追加されます。

F.5.7　グリッド線表示による部品の整列支援機能

グリッド線の表示を有効にすることで、部品配置の目安として設計時のみ表示される方眼状の罫線を表示できます。このグリッド線は設計時にのみ表示され、WYSIWYGでの画面編集時に部品間の間隔や部品サイズ、配置座標をそろえる際の目安とすることができます。

同時にグリッド線への吸着機能を併用することで、部品の左上座標をグリッド線の交点に自動的に合わせて配置することができます。これにより、画面のレイアウトを指定した間隔で容易に整列させることができます。

また、グリッド線だけでなく、ユーザー指定の間隔で吸着機能を設定することも可能です。

グリッドの設定

グリッドの設定をすると、グリッド線の表示の有無、吸着の設定などを行うことができます。

本設定時に画面フォーム固有に設定がある場合、画面フォームの固有設定を上書きします。

グリッドの設定は、以下の手順で設定します。

1. [Window]メニューから[Preferences]を選択します。
2. [Preferences]ダイアログボックスの設定項目で、[Ajaxページエディタ]を選択します。
3. グリッド線の表示の有無、吸着設定などを行います。

以下の表に、[Ajaxページエディタ設定]ページのグリッドに関する項目を説明します。

項目	説明
線を表示	グリッド線を表示します。 デフォルトはチェックなしです。
間隔	表示するグリッド線の表示間隔を設定します。 デフォルトは10pxです。
線に吸着	部品の配置・移動時、部品をグリッド線に自動的に合わせて配置します(注1)。 グリッド線が表示・非表示どちらでも有効にできます。 デフォルトはチェックなしです。
指定間隔で吸着	部品の配置・移動時、部品を設定した間隔で自動的に合わせて配置します。 グリッド線が表示・非表示どちらでも有効にできます。 デフォルトはチェックなしです。
間隔	指定間隔で吸着させる場合、吸着の間隔を設定します。 有効な設定値は1～900までの整数です。 デフォルトは10pxです。

注1)[線に吸着]設定と[指定間隔で吸着]設定はどちらかしか選択できません。

以下に、[Ajaxページエディタ設定]ページのカーソルキーの移動幅に関する項目を説明します。

項目	説明
グリッド幅に合わせる	カーソルキーによる移動幅をグリッド線に合わせます。 デフォルトはチェックなしです(注1)。
ユーザー指定幅に合わせる	カーソルキーによる移動幅を縦、横の指定幅に合わせます。 デフォルトはチェックありです。
縦	上下カーソルキーを押した場合の縦の移動幅を設定します。 デフォルトは1pxです。
横	左右カーソルキーを押した場合の横の移動幅を設定します。 デフォルトは1pxです。

注1)グリッド線を表示していなくても設定が適用されます。

注意

- [Ajaxページエディタ設定]ページ上での設定は、ワークベンチ全体を適用対象とする共通設定です。
- グリッドの設定のうち、グリッド線の表示、吸着に関する設定は編集するファイルごとに設定することもできます。詳細は「F.5.1 設計ビューのコンテキストメニュー」を参照してください。
- [Ajaxページエディタ設定]ページ上での設定内容と編集するファイルごとの設定内容が異なる場合、後で設定した内容が優先されます。

F.6 ソースビュー

ソースビューでは、HTML/JSPファイルをプレーンテキストとして編集します。

HTML/JSPファイルの表示形式は、以下の設定画面で設定します。

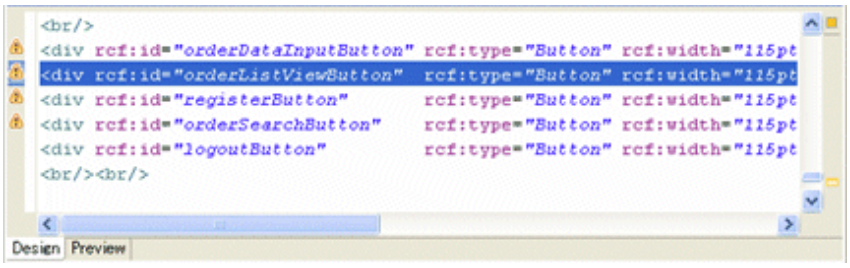
- HTMLファイルの場合
  [Preferences]ダイアログボックスの[Web] > [HTML Files] > [Editor]
- JSPファイルの場合
  [Preferences]ダイアログボックスの[Web] > [JSP Files] > [Editor]

ソースビューでは、以下の編集支援機能を利用することができます。

- コンテキストメニュー
- 入力支援/妥当性検証

以下に、ソースビューの画面例を示します。

図F.6 ソースビュー



F.6.1 ソースビューのコンテキストメニュー

ソースビューのコンテキストメニューのコマンドを以下の表に示します。

コマンド名	機能	キーボードショートカット
Undo	直前に行った動作を元に戻します。	Ctrl + Z
Revert File	前回保存した状態に戻します。	なし
Save	作業中のファイルを保存します。	Ctrl + S
Cut	選択したデータを切り取ってクリップボードに保存します。	Ctrl + X
Copy	選択したデータをコピーしてクリップボードに保存します。	Ctrl + C
Paste	クリップボードの内容を貼り付けます。 クリップボードにテキスト形式で保存されている場合に有効です。	Ctrl + V
Format	エディタの内容をフォーマットします。	Ctrl + Shift + F
Format Active Elements	アクティブ要素の内容をフォーマットします。	Ctrl + I
プロパティ	プロパティビューを表示します。	なし
Preferences	[HTMLファイル]または[JSPファイル]の設定ページを表示します。	なし

F.6.2 入力支援/妥当性検証

ソースビューでHTML/JSPファイルを編集する場合、以下の入力支援機能を利用することができます。

- HTML/JSPタグ、UJIタグ、UI部品タグ
  キーボードで「<」を押すことによって、入力可能なタグ名の一覧がリストに表示されます。
  また、タグの「<」と「>」の間にカーソルを置いて、Ctrl + Spaceを押すことによって、指定できるプロパティの一覧が入力候補としてリストに表示されます。

- JavaScript
  ファイルから読み込まれて実行時に使用できるJavaScriptの変数、関数、予約語の一覧がリストに表示されます。

- 通信処理のテンプレート
  入力候補として、通信処理のテンプレートがリストに表示されます。
  <script>タグ内では、キーボードで以下の文字列を入力し、Ctrl + Spaceを押します。

  - UjiRequest
    UjiRequest.send関数の記述例が入力候補として表示されます。

  - RcfRequest
    RcfRequest.send関数およびRcfRequest.invoke関数の記述例が入力候補として表示されます。

  - MuRequest
    MuRequest.send関数の記述例が入力候補として表示されます。

- styleタグ

  <style>タグ内では、キーボードで「.r」を入力し、Ctrl + Spaceを押すことによって、必要に応じて、入力可能な画面部品のスタイルのクラス名、スタイルプロパティ名、スタイルプロパティ値の一覧が入力候補としてダイアログにリスト表示されます。

  詳細は、「CSSの入力支援」を参照してください。

また、以下の妥当性検証機能により、記述に誤りがある可能性がある場合、エラーまたは警告が表示されます。

- HTML/JSPタグ、UJIタグ、UI部品タグ
  定義されていないプロパティを指定した場合や、プロパティに正しくない値を指定した場合、ソースビューおよび[Problems]ビューにエラーまたは警告が表示されます。

- JavaScript
  構文エラーがある場合や、未定義の関数や変数を使用した場合、ソースビューおよび[Problems]ビューにエラーまたは警告が表示されます。

Ajaxフレームワークアプリケーションで利用可能なJavaScript API、UI部品のJavaScript APIは実行時に動的に定義されるため、編集時は未定義になりエラーが表示されます。

ポイント

以下の方法で、JavaScriptに関するエラーの表示条件を変更できます。

1. [Window]メニューから[Preferences]を選択します。
2. [Preferences]ダイアログボックスの設定項目で、[JavaScript] > [Validator] > [Errors/Warnings]を選択します。
3. 必要に応じて修正します。

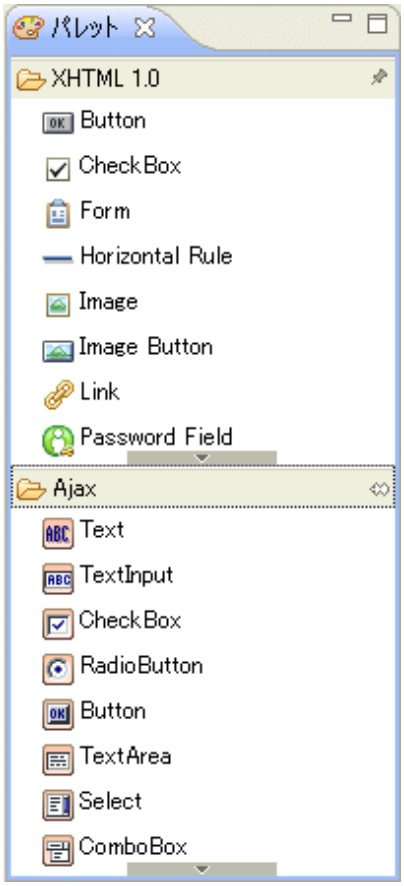
F.7 パレットビュー

パレットビューは、部品を設計ビューに配置するときに利用します。パレットビューで選択した部品を設計ビューに配置します。

パレット上の部品は、HTMLタグ(XHTML1.0)と画面部品(Ajax)の2つのカテゴリに分けて表示されます。

以下に、パレットビューの画面例を示します。

図F.7 パレットビュー



F.7.1 パレットビューのコンテキストメニュー

パレットビューのコンテキストメニューのコマンドを以下の表に示します。

コマンド名	機能	キーボードショートカット
Hide(注1)	パレットでカテゴリを非表示にします。	なし
Layout	部品のレイアウトを変更します。	なし

コマンド名	機能	キーボードショートカット
Use Large Icon	パレットで大きいアイコンを使用して部品を表示します。	なし
Customize	[Customize Palette]ダイアログボックスを表示し、パレット上のカテゴリをカスタマイズします。	なし
Settings	[Palette Settings]ダイアログボックスを表示し、パレット上の部品の表示形式を設定します。	なし
Pinned(注1)	ピン留めされた場合、そのカテゴリを閉じずに開いておきます。	なし

注1) [Hide]および[Pinned]コマンドは、カテゴリで右クリックした場合にだけ表示されます。

F.7.2　パレットビューのカテゴリ

パレットビューでは、以下のカテゴリ単位に部品が表示されます。

- XHTML1.0カテゴリ(HTMLタグ)
- Ajaxカテゴリ(画面部品)

ここでは、それぞれのカテゴリで表示される部品について説明します。

XHTML1.0カテゴリ

以下の表に、XHTML1.0カテゴリで表示される部品を示します。

表示名	HTMLタグ	説明
Button	input	type=submitを備えたinput要素はフォームを実行するようにユーザーエージェントに知らせるための入力オプション、ボタンを表します。
CheckBox	input	type=checkboxを備えたinput要素はboolean選択を表します。同じ名前を持った1セットの要素は多数のnの選択フィールドを表します。
Form	form	form要素は要素を組み立てるドキュメントに加えて、入力要素のシーケンスを含んでいます。
HorizontalRule	hr	hr要素はテキストのセクションの間のデバイダーです。通常は、全幅罫線か同等なグラフィックです。
Image	img	img要素はハイパーリンクでイメージかアイコンを参照します。
Image Button	input	type=imageを備えたinput要素は表示するイメージリソースを指定し、イメージから選択したピクセルのXとY座標を指定します。
Link	a	a要素はユーザーがドキュメントの内容をナビゲートすることを可能にします。
Password Field	input	type=passwordを備えたinput要素はパスワードが入力されるとともに、値が見えなくなる以外は、テキストフィールドと同じです。
Radio Button	input	type=radioを備えたinput要素はboolean選択を表します。同じ名前がある1セットの要素は多くの中から1つの選択フィールドを表します。
Select	select	select要素は値を列挙したリストへの形式フィールドを抑制します。
TextArea	textarea	textarea要素はマルチラインテキストフィールドを表します。
Text Field	input	type属性のデフォルト値はシングルラインテキストエントリフィールドを示すtextです。
Label	label	label要素はラベルを表します。
Div	div	div要素はブロック要素を表します。

Ajaxカテゴリ

以下の表に、Ajaxカテゴリで表示される部品を示します。

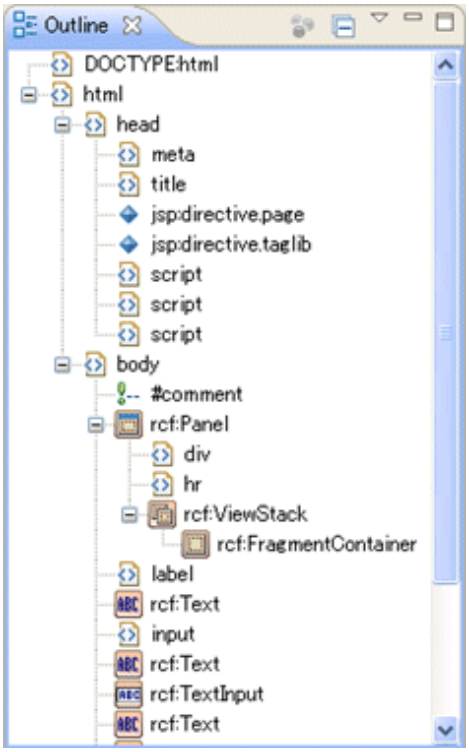
画面部品 (表示名)	説明
Text	テキストを表示する部品です。
TextInput	単一行のテキストを入力および編集するための部品です。
CheckBox	オンまたはオフの状態を表すチェックボックス部品です。
RadioButton	ラジオボタン部品です。
Button	ボタン部品です。
TextArea	単一行または複数行のテキストを、入力および編集するための部品です。
Select	単一選択および複数選択が可能な選択リスト部品です。
ComboBox	入力フィールドと選択リストから構成され、項目の選択および選択されている項目の表示を行う部品です。
DateInput	TextInputの一種であり、日付および時間データを、入力および編集するための部品です。
NumberInput	TextInputの一種であり、数値を入力および編集するための部品です。
MaskedTextInput	TextInputの一種であり、フォーマットパターンを設定して、穴埋め方式でテキストを入力および編集するための部品です。 (注) Internet Explorerでだけ利用できます。
MaskedDateInput	TextInputの一種であり、日時のフォーマットパターンを設定して、入力文字を数字だけに制約し、穴埋め方式でテキストを入力および編集するための部品です。 (注) Internet Explorerでだけ利用できます。
SelectList	単一選択および複数選択が可能な選択リスト部品です。
CheckList	チェックボックス付きのリスト部品です。
ViewContainer	汎用コンテナ部品です。
Panel	タイトル部、ボディ部から成り立っているタイトルバー付きコンテナ部品です。
ViewStack	同じ位置で表示切替えを行う場合に利用するコンテナ部品です。
TabPanel	タブにより表示切替えを行う場合に利用するコンテナ部品です。
FragmentContainer	ページ表示後、任意のタイミングで外部から画面情報を読み込み表示するためのコンテナ部品です。
TableView	2次元のデータを表形式で表示する部品です。
TableEdit	2次元のデータを表形式で表示し、編集することもできる部品です。
DataGrid	2次元のデータを表形式で表示し、編集することもできる部品です。テーブル内にチェックボックスやツリーなどの表示ができます。 (注) Internet Explorerでだけ利用できます。
Calendar	カレンダの表示と日付の選択を行う部品です。
CalendarButton	PopupCalendarを表示するボタン部品です。PopupCalendarと組み合わせて利用します。
TreeView	ツリー形式で項目リストを表示する部品です。
ScrapingView	Webアプリケーションからスクレイピングしたコンテンツを表示する部品です。

F.8 アウトラインビュー

アウトラインビューでは、画面上の部品がツリー形式で一覧表示されます。また、画面上の部品を一覧から選択することができます。

以下に、アウトラインビューの画面例を示します。

図F.8 アウトラインビュー



F.8.1　アウトラインビューのコンテキストメニュー

アウトラインビューでは、<body>タグ中のUI部品およびHTMLタグに対して、コンテキストメニューが表示されます。
アウトラインビューのコンテキストメニューのコマンドを以下の表に示します。

コマンド名	機能	キーボードショートカット
削除	選択したノードを削除します。	なし
モデルバインディング定義	モデルバインディングを定義します。	なし
イベント処理定義	イベント処理を定義します。	なし
機能付加部品バインディング定義	画面部品と機能付加部品のバインディングを定義します。	なし
プロパティ	選択したノードのプロパティを表示します。	なし

F.8.2　タグの表示形式

アウトラインビューでは、タグは、「アイコン + 表示名」の形式で表示されます。

HTMLタグの表示形式

HTMLタグの表示名は、タグ名(htmlなど)になり、htmlを示すアイコンに加えて表示されます。

JSPタグの表示形式

JSPタグ(JSPアクション)の表示名は、タグ名(jsp:attributeなど)になり、jspを示すアイコンに加えて表示されます。また、JSP directiveは「アイコン + "jsp:directive." + directive名」の形式で表示されます。

UJIタグの表示形式

UJIタグ(Apcoordinatorが提供するJSP用の拡張タグ)の表示名は、タグ名(uji:actionなど)になり、taglibを示すアイコンに加えて表示されます。ほかのTaglibファイルで定義するJSPタグも、UJIタグと同様に、taglibを示すアイコンで表示されます。

UI部品タグの表示形式

UI部品タグの表示名は、「rcf: + rcf:type属性で指定する名前」(rcf:Textなど)になり、各UI部品のアイコンに加えて表示されます。

未知タグの表示形式

未知のタグの表示名は、タグ名(mytagなど)になり、未知タグを示すアイコンに加えて表示されます。

F.9　プロパティビュー

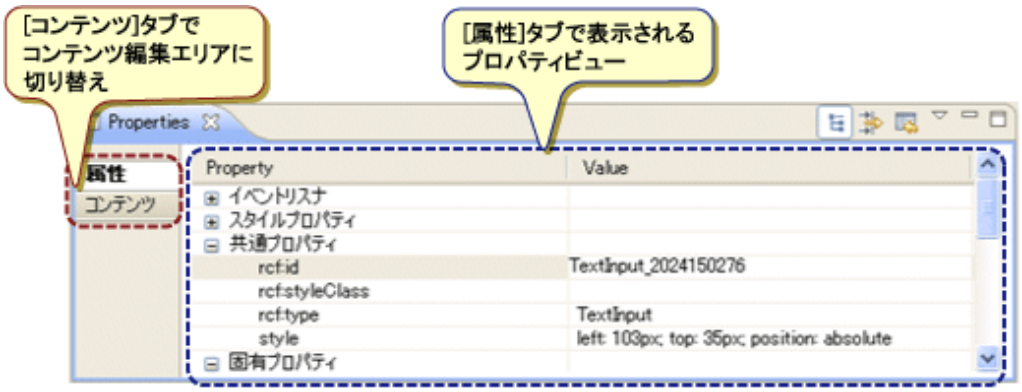
プロパティビューでは、設計ビュー、ソースビュー、またはアウトラインビューで選択されたタグや部品のプロパティが表示されます。また、表示されたプロパティを編集することができます。

プロパティビューでは、以下のように、タブによって表示内容を切り替えることができます。

- [属性]タブ: 選択されたタグや部品のプロパティの属性が表示されます。
- [コンテンツ]タブ : コンテンツ編集エリアに切り替わり、選択されたタグや部品のコンテンツ内容が表示されます。

以下に、プロパティビューの画面例を示します。

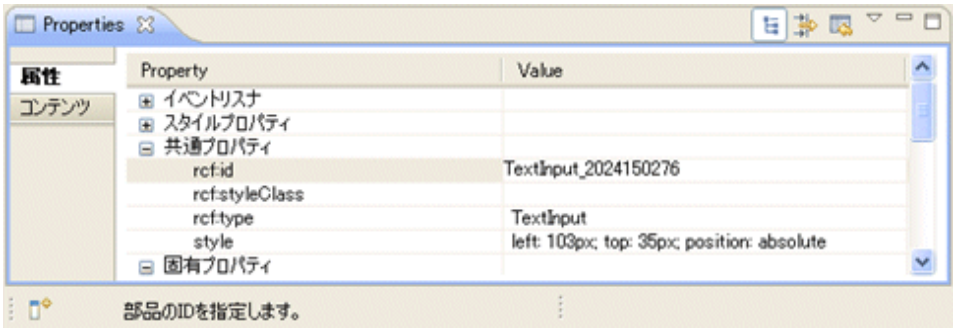
図F.9 プロパティビュー



F.9.1　属性の表示と編集

プロパティビューの[属性]タブでは、選択されたタグや部品のプロパティの属性と値が表示されます。また、表示された属性値を編集することができます。

以下に、プロパティビューの[属性]タブの画面例を示します。



プロパティビューの[属性]タブで表示される項目を以下に示します。

項目	説明
Property	プロパティは、以下のカテゴリ分けで階層化して表示します。 ・ 共通プロパティ 画面部品共通プロパティを表示します。 ・ 固有プロパティ UI部品に固有のプロパティを表示します。 ・ スタイルプロパティ スタイルのプロパティを表示します。 ・ イベントリスナ 使用できるイベントリスナを表示します。 ・ その他 上記以外のプロパティを表示します。
Value	プロパティの値を表示します。

なお、プロパティビューの[属性]タブでは、Eclipseが提供するコンテキストメニューが表示されます。

属性の表示形式

プロパティビューの[属性]タブでは、各プロパティは、以下の形式で表示されます。

表示形式	説明
テキスト入力	入力フィールドをテキストで入力できるようにします。
標準コンボボックス入力	入力フィールドをコンボボックスにし、値を選択できるようにします。

注意

プロパティビューで値を削除した場合、プロパティは削除されず、値に""が設定されます。
このため、省略値が空文字列("")以外のプロパティをプロパティビューで削除すると不当な値になります。

その結果、設計ビューでエラーが発生し、編集できなくなります。
この場合、以下のどれかの対処を実施してください。

a. 空となったプロパティを削除する

　― ソースビューでプロパティを削除する

　― プロパティビューのコンテキストメニューから[Restore Default Value]を選択する

b. プロパティ値に適切な値を指定する

ただし、プロパティが必須プロパティの場合、プロパティを削除するとエラーが発生します。対処b.を実施してください。

なお、プロパティの省略値については、「UI部品リファレンス」の各部品のプロパティの説明を参照してください。

F.9.2　コンテンツの表示と編集

プロパティビューの[コンテンツ]タブ をクリックすると、コンテンツ編集エリアに切り替わり、選択されたタグや部品のコンテンツ内容が表示されます。コンテンツ編集エリアでは、コンテンツ内容を編集することができます。

以下に、コンテンツ編集エリアの画面例を示します。

コンテンツ編集エリアでは、Eclipseが提供するコンテキストメニューが表示されます。

編集対象のHTMLタグ

コンテンツ編集エリアでは、以下のHTMLタグ(空要素)を除いた、HTMLタグを編集することができます。

- area
- base
- br
- col
- hr
- img
- input
- link
- meta
- param
- table

また、上記以外のHTMLタグを編集対象外にすることもできます。
編集対象外にするHTMLタグは、以下の手順で設定します。

1. [Window]メニューから[Preferences]を選択します。
2. [Preferences]ダイアログボックスの設定項目で、[Ajaxページエディタ] > [プロパティビュー]を選択します。
3. [コンテンツ タブ]の[コンテンツ編集不可とする要素]に編集対象外とするHTMLタグの要素名を入力します。
   複数指定する場合は、要素名をカンマ(,)で区切ります。

改行の入力

コンテンツ編集エリアでは、以下のどちらかのキーで改行を入力します。

- Ctrl + Enter
- Shift + Enter

上記のどちらのキーで改行を入力するかを選択することができます。デフォルトは、Ctrl + Enterです。
改行の入力キーは、以下の手順で設定します。

1. [Window]メニューから[Preferences]を選択します。
2. [Preferences]ダイアログボックスの設定項目で、[Ajaxページエディタ] > [プロパティビュー]を選択します。
3. [コンテンツ タブ]の[改行の入力キー]で、改行の入力キーを選択します。

特殊文字の変換

エスケープを設定すると、以下の特殊文字の変換が行われます。
ただし、すでにコンテンツ内容に値が設定されている場合は、特殊文字の変換は行われません。

元の文字	変換後の文字
"	&quot;
&	&amp;
<	&lt;
>	&gt;

エスケープは、以下の手順で設定します。

1. [Window]メニューから[Preferences]を選択します。
2. [Preferences]ダイアログボックスの設定項目で、[Ajaxページエディタ] > [プロパティビュー]を選択します。
3. [コンテンツ タブ]の[特殊文字の変換]をチェックします。

F.10 クライアントフレームワーク連携編集

Ajaxページエディタには、クライアントフレームワークの機能を利用するために、クライアントフレームワーク連携編集機能があります。
クライアントフレームワーク連携編集機能では、以下の機能を提供しています。

- モデルバインディング定義機能
  画面部品のプロパティの値に、モデルのデータやほかの画面部品のプロパティを選択して割り当てる機能です。
- イベント処理定義機能
  画面部品のイベント処理を選択して割り当てる機能です。
- 機能付加部品バインディング定義機能
  機能付加部品の機能を付加する対象の画面部品を選択して割り当てる機能です。

F.10.1 モデルバインディング定義

モデルバインディング定義では、[モデルバインディング定義]ダイアログボックスを利用して、画面部品のプロパティのバインディングを設定します。

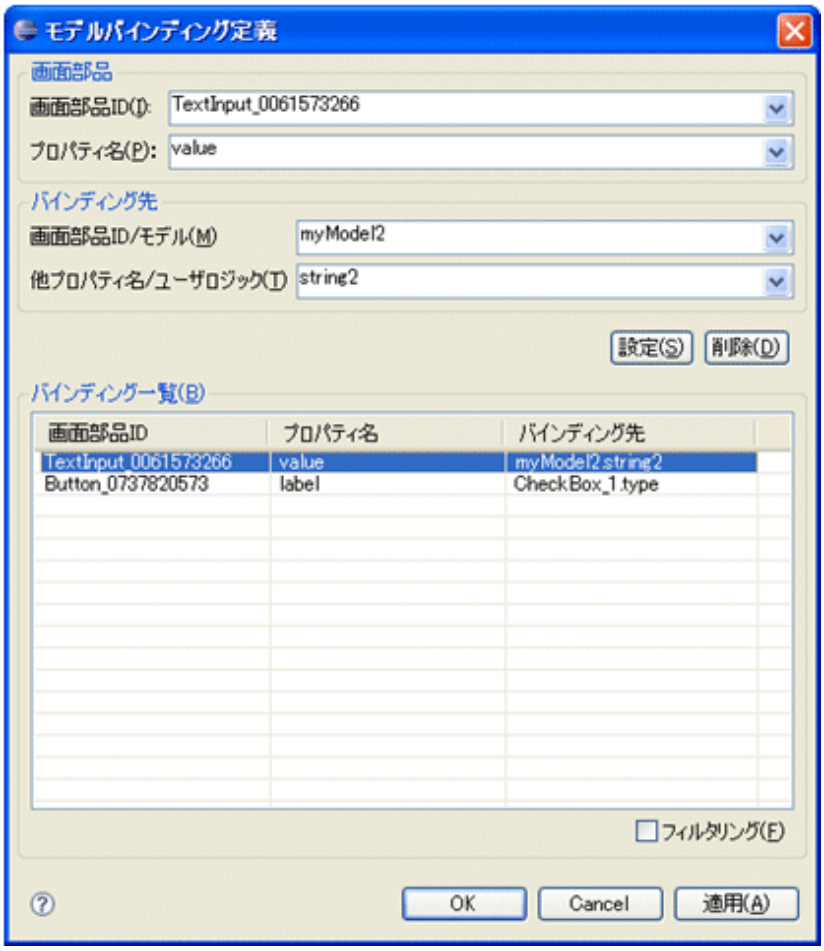
モデルバインディング定義で定義した内容は、ソースビューおよびプロパティビューに反映されます。

[モデルバインディング定義]ダイアログボックスの起動

[モデルバインディング定義]ダイアログボックスは、以下のどれかの方法で起動します。

- 設計ビューで画面部品を選択しないで、コンテキストメニューの[モデルバインディング定義]コマンドを選択します。
  [モデルバインディング定義]ダイアログボックスのバインディング一覧には、ソースコードに定義されているすべてのバインディング情報が表示されます。
- 設計ビューで画面部品を選択して、コンテキストメニューの[モデルバインディング定義]コマンドを選択します。
  [モデルバインディング定義]ダイアログボックスのバインディング一覧には、選択した部品のバインディング情報だけが表示されます。
- アウトラインビューで画面部品または機能付加部品を選択して、コンテキストメニューの[モデルバインディング定義]コマンドを選択します。
  [モデルバインディング定義]ダイアログボックスのバインディング一覧には、選択した部品のバインディング情報だけが表示されます。

以下に、[モデルバインディング定義]ダイアログボックスの画面例を示します。



以下の表に、[モデルバインディング定義]ダイアログボックスの項目を説明します。

項目	説明
画面部品	モデルバインディングを定義する対象の画面部品およびプロパティを指定します。
画面部品ID	モデルバインディングを定義する対象の画面部品IDを指定します。 設計ビューまたはアウトラインビューで部品を選択した場合は、初期値として、選択中の部品IDが表示されます。部品を選択していない場合は、何も表示されません。 部品IDは、選択リストから選択することができます。 なお、選択中の部品にIDが設定されていない場合は、画面部品IDはグレー表示され編集できません。
プロパティ名	画面部品のプロパティ名を指定します。 プロパティ名は、選択リストから選択することができます。
バインディング先	画面部品のバインディング先を指定します。
画面部品ID/モデル	バインディング先の画面部品IDまたはモデル名を指定します。 画面部品IDおよびモデル名は、選択リストから選択することができます。
他プロパティ名/ユーザーロジック	バインディング先の画面部品のプロパティ名またはモデルのユーザーロジックを指定します。 プロパティ名およびユーザーロジックは、選択リストから選択することができます。
[設定]ボタン	[画面部品]および[バインディング先]で指定した情報をバインディング一覧に登録します。

項目	説明
	ダイアログボックス上部の[画面部品]および[バインディング先]の4つの項目がすべて指定されると、[設定]ボタンはアクティブになります。
[削除]ボタン	バインディング一覧で選択した行を削除します。 バインディング一覧で行が選択されていると、[削除]ボタンはアクティブになります。
バインディング一覧	ソースコードに定義されているバインディング情報が一覧表示されます。 バインディング一覧の行を選択すると、その情報がダイアログボックス上部の[画面部品]および[バインディング先]の4つの項目に表示されます。
画面部品ID	モデルバインディングを定義する対象の画面部品IDが表示されます。 [画面部品ID]ヘッダをクリックすると、[画面部品ID]の内容に従ってソートすることができます。
プロパティ名	画面部品のプロパティ名が表示されます。 [プロパティ名]ヘッダをクリックすると、[プロパティ名]の内容に従ってソートすることができます。
バインディング先	バインディング先の画面部品IDと他プロパティ名、またはモデル名とユーザーロジックが、「.」で区切られて表示されます。 [バインディング先]ヘッダをクリックすると、[バインディング先]の内容に従ってソートすることができます。
フィルタリング	バインディング一覧に表示する情報にフィルタをかけます。 [フィルタリング]をチェックすると、[画面部品]の[画面部品ID]で指定された画面部品に関するバインディング情報だけが表示されます。 [フィルタリング]をチェックしないと、ソースコードに定義されているすべてのバインディング情報が表示されます。 なお、[画面部品]の[画面部品ID]が指定されていない場合は、[フィルタリング]をチェックしても、ソースコードに定義されているすべてのバインディング情報が表示されます。[画面部品ID]がグレー表示されている場合は、[フィルタリング]チェックボックスは無効となり、選択中の画面部品に関するバインディング情報だけが表示されます。
[OK]ボタン	バインディング一覧に表示されているすべての情報をソースコードに反映し、[モデルバインディング定義]ダイアログボックスを閉じます。
[Cancel]ボタン	バインディング一覧の変更内容を破棄し、[モデルバインディング定義]ダイアログボックスを閉じます。
[適用]ボタン	バインディング一覧に表示されているすべての情報をソースコードに反映します。[モデルバインディング定義]ダイアログボックスは閉じません。

モデルバインディング定義の新規追加

以下に、モデルバインディング定義を新規に追加する方法を説明します。

1. [モデルバインディング定義]ダイアログボックス上部の[画面部品]および[バインディング先]の4つの項目を指定します。
2. [設定]ボタンをクリックし、指定した情報をバインディング一覧に登録します。
3. バインディング一覧の内容を確認したあと、[OK]ボタンまたは[適用]ボタンをクリックし、バインディング一覧の情報をソースコードに反映します。

モデルバインディング定義の変更

以下に、モデルバインディング定義を変更する方法を説明します。

1. バインディング一覧から定義を変更する行を選択します。
   選択した行の設定内容は、[モデルバインディング定義]ダイアログボックス上部の[画面部品]および[バインディング先]の4つの項目に表示されます。

2. 表示された4つの項目のうち、変更する項目の値を修正し、[設定]ボタンをクリックします。
すでに同じ情報がバインディング一覧に登録されている場合は、[上書き確認]ダイアログボックスが表示されるので、[Yes]ボタンをクリックします。

ポイント

以下のどちらかの方法で、[上書き確認]ダイアログボックスを表示しないようにすることができます。

- [Preferences]ダイアログボックスの設定を変更します。
  1. [Window]メニューから[Preferences]を選択します。
  2. [Preferences]ダイアログボックスの設定項目で、[Ajaxページエディタ] > [モデルバインディング]を選択します。
  3. [上書き確認ダイアログを表示する]のチェックを外します。
- [上書き確認]ダイアログボックスの[常にプロンプトなしで終了]をチェックします。
[常にプロンプトなしで終了]をチェックすると、[Preferences]ダイアログボックスの[Ajaxページエディタ] > [モデルバインディング]の[上書き確認ダイアログを表示する]のチェックが外れます。

3. バインディング一覧の内容を確認したあと、[OK]ボタンまたは[適用]ボタンをクリックし、バインディング一覧の情報をソースコードに反映します。

モデルバインディング定義の削除

以下に、モデルバインディング定義を削除する方法を説明します。

1. バインディング一覧から定義を削除する行を選択します。
2. [削除]ボタンをクリックし、選択した情報をバインディング一覧から削除します。
3. バインディング一覧の内容を確認したあと、[OK]ボタンまたは[適用]ボタンをクリックし、バインディング一覧の情報をソースコードに反映します。

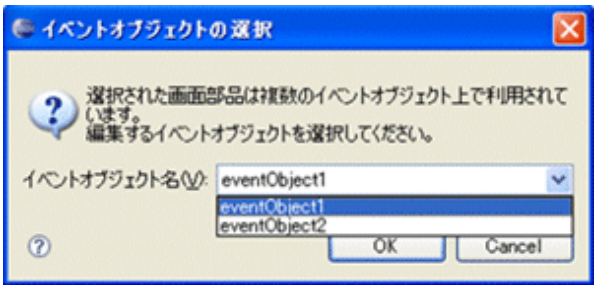
F.10.2 イベント処理定義

イベント処理定義では、[イベント処理定義]ダイアログボックスを利用して、画面部品にイベント処理を定義します。

設計ビューまたはアウトラインビューで画面部品を選択し、コンテキストメニューの[イベント処理定義]コマンドを選択すると、[イベント処理定義]ダイアログボックスが表示されます。

ポイント

選択された画面部品に複数のイベントオブジェクトが定義されている場合は、[イベント処理定義]ダイアログボックスの前に、以下に示す[イベントオブジェクトの選択]ダイアログボックスが表示されるので、編集するイベントオブジェクト名を選択します。



[イベント処理定義]ダイアログボックスでは、以下の方法でイベント処理を定義することができます。

- 画面部品に直接定義
画面部品のプロパティにイベント処理を直接定義します。

- イベントオブジェクトに定義
  イベントオブジェクトに対して画面部品とイベント処理の情報を定義します。

なお、選択した画面部品にIDが存在しない場合は、画面部品に直接定義する方法だけが選択できます。選択した画面部品にIDが存在する場合は、両方の定義方法を選択することができます。

イベント処理定義で定義した内容は、ソースビューおよびプロパティビューに反映されます。

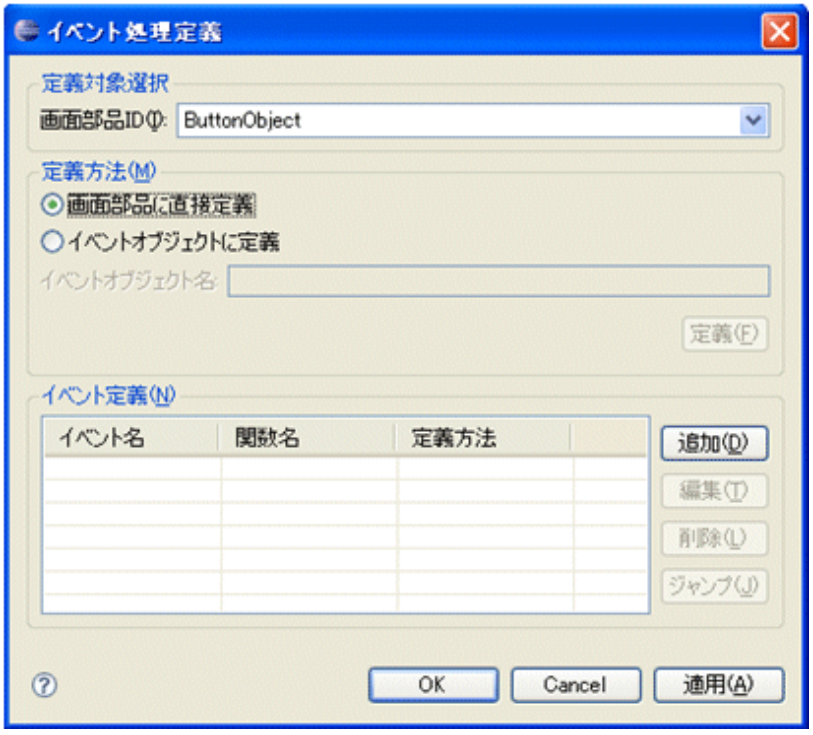
FragmentContainer部品で使用するHTML/JSPファイルでは、画面部品に直接定義する方法だけが選択できます。その場合、新しく定義した関数は、画面部品のプロパティに関数名が設定されるだけで、<script>タグ内にソースは生成されません。

ポイント

FragmentContainer部品で使用するHTML/JSPファイルかどうかは、[Ajaxページエディタ設定]ページの[編集時に指定したCSSファイル、JavaScriptファイルを読み込む]をチェックしているかどうかで判定します。

画面部品に直接定義

以下に、[イベント処理定義]ダイアログボックスで、イベント処理を画面部品に直接定義する例を示します。
[イベント処理定義]ダイアログボックスでは、イベント処理定義を編集および削除することもできます。



以下に、[イベント処理定義]ダイアログボックスの項目を説明します。

項目	説明
画面部品ID	イベント処理を定義する対象の画面部品IDを指定します。 初期値として、選択した画面部品のIDが表示されます。選択した画面部品にIDが設定されていない場合は、何も表示されません。 画面部品IDは、選択リストから選択することができます。
定義方法	イベント処理の定義方法を選択します。 ここでは、「画面部品に直接定義」を選択します。
イベント定義一覧	選択した画面部品に定義されているイベント定義が一覧表示されます。 表示される項目は、画面部品で発生するイベント名、イベントに対して呼び出される関数名、イベント処理の定義方法です。

項目	説明
[追加]ボタン	[イベント処理定義の追加]ダイアログボックスを表示し、ダイアログボックスで指定されたイベント名、関数名を追加します。
[編集]ボタン	[イベント処理定義の追加]ダイアログボックスを表示し、選択された行のイベント名、関数名をダイアログボックスで指定されたイベント名、関数名に変更します。
[削除]ボタン	選択された行を削除します。
[ジャンプ]ボタン	ソースビューのキャレットを選択された関数に位置付けます。
[OK]ボタン	イベント定義一覧に表示されているすべての情報をソースコードに反映し、[イベント処理定義]ダイアログボックスを閉じます。
[Cancel]ボタン	イベント定義一覧の変更内容を破棄し、[イベント処理定義]ダイアログボックスを閉じます。
[適用]ボタン	イベント定義一覧に表示されているすべての情報をソースコードに反映します。[イベント処理定義]ダイアログボックスは閉じません。

上記の[イベント処理定義]ダイアログボックスで[追加]ボタンをクリックすると、[イベント処理定義の追加]ダイアログボックスが表示されます。

[イベント処理定義の追加]ダイアログボックスで、以下の項目を指定します。

項目	説明
イベント名	画面部品で発生するイベント名が候補として表示されるので、その中から選択します。 なお、追加済みのイベント名は表示されません。
関数名	JavaScriptに定義されているすべてのfunction名(関数名)が候補として表示されるので、その中から選択します。 また、新規に作成する関数の名前を直接入力することもできます。

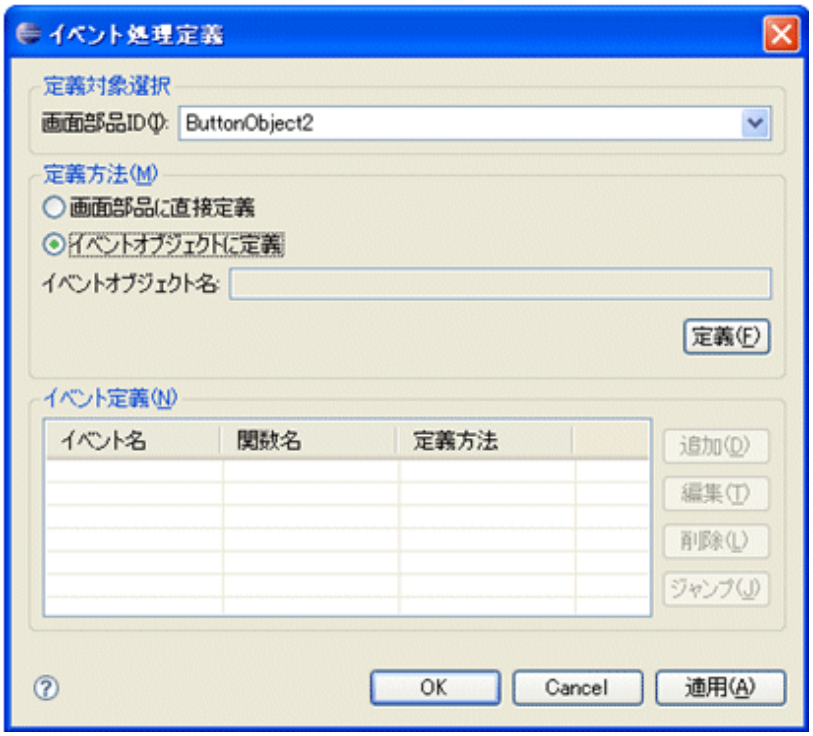
この例では、イベント名に「click」を、関数名に「search」を選択します。

上記の[イベント処理定義の追加]ダイアログボックスで[OK]ボタンをクリックすると、設定したイベント名、関数名および定義方法が[イベント処理定義]ダイアログボックスのイベント定義一覧に反映されます。

イベント定義一覧の内容を確認したあと、[OK]ボタンまたは[適用]ボタンをクリックし、イベント定義一覧の情報をソースに反映します。

イベントオブジェクトに定義

以下に、[イベント処理定義]ダイアログボックスで、イベントオブジェクトに対して画面部品とイベント処理の情報を定義する例を示します。[イベント処理定義]ダイアログボックスでは、イベント処理定義を編集および削除することもできます。



以下に、[イベント処理定義]ダイアログボックスの項目を説明します。

項目	説明
画面部品ID	イベント処理を定義する対象の画面部品IDを指定します。 初期値として、選択した画面部品のIDが表示されます。選択した画面部品にIDが設定されていない場合は、何も表示されません。 画面部品IDは、選択リストから選択することができます。
定義方法	イベント処理の定義方法を選択します。 ここでは、「イベントオブジェクトに定義」を選択します。
イベントオブジェクト名	選択した画面部品に定義されているイベントオブジェクト名が表示されます。選択した画面部品にイベントオブジェクトが定義されていない場合は、何も表示されません。 この項目は直接編集できません。
[定義]/[編集]ボタン	選択した画面部品にイベントオブジェクトが定義されていない場合は[定義]ボタンが、選択した画面部品にイベントオブジェクトが定義されている場合は[編集]ボタンがアクティブになります。 ・ [定義]ボタン [イベントオブジェクト定義]ダイアログボックスを表示し、イベントオブジェクトを新規に定義します。 ・ [編集]ボタン [イベントオブジェクト編集]ダイアログボックスを表示し、既存のイベントオブジェクトのイベントオブジェクト名やイベント登録名を更新します。
イベント定義一覧	選択した画面部品に定義されているイベント定義が一覧表示されます。表示される項目は、画面部品で発生するイベント名、イベントに対して呼び出される関数名、イベント処理の定義方法です。
[追加]ボタン	[イベント処理定義の追加]ダイアログボックスを表示し、ダイアログボックスで指定されたイベント名、関数名を追加します。

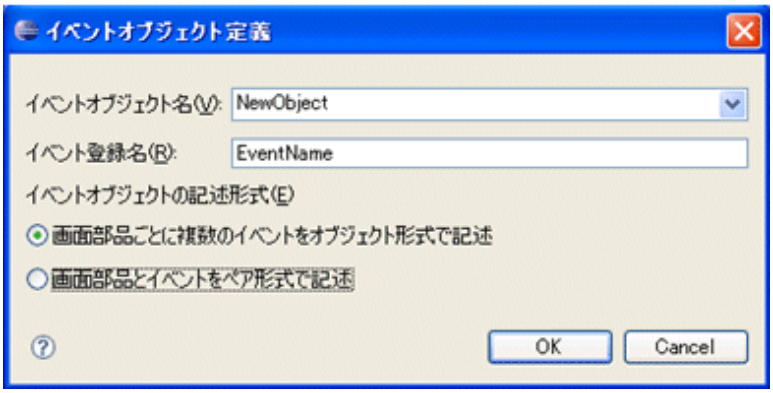
項目	説明
[編集]ボタン	[イベント処理定義の追加]ダイアログボックスを表示し、選択された行のイベント名、関数名をダイアログボックスで指定されたイベント名、関数名に変更します。
[削除]ボタン	選択された行を削除します。
[ジャンプ]ボタン	ソースビューのキャレットを選択された関数に位置付けます。
[OK]ボタン	イベント定義一覧に表示されているすべての情報をソースコードに反映し、[イベント処理定義]ダイアログボックスを閉じます。
[Cancel]ボタン	イベント定義一覧の変更内容を破棄し、[イベント処理定義]ダイアログボックスを閉じます。
[適用]ボタン	イベント定義一覧に表示されているすべての情報をソースコードに反映します。[イベント処理定義]ダイアログボックスは閉じません。

イベントオブジェクトに定義する場合、以下の2通りの方法があります。

- 画面部品ごとに複数のイベントを一括して定義
  1つのイベントオブジェクトに、「画面部品1:イベント1,イベント2,..., 画面部品2:イベント1,イベント3,...」のように情報を定義します。
- 画面部品とイベントの組ごとに定義
  1つのイベントオブジェクトに、「画面部品1_イベント1, 画面部品1_イベント2,...」のように情報を定義します。

画面部品ごとに複数のイベントを一括して定義

[イベント処理定義]ダイアログボックスで[定義]ボタンをクリックすると、[イベントオブジェクト定義]ダイアログボックスが表示されます。



[イベントオブジェクト定義]ダイアログボックスで、以下の項目を指定します。

項目	説明
イベントオブジェクト名	JavaScriptに定義されているイベントオブジェクト名が候補として表示されるので、その中から選択します。 また、イベントを登録するイベントオブジェクトの名前を直接入力することもできます。
イベント登録名	イベント登録名を直接入力して指定します。
イベントオブジェクトの記述形式	イベントオブジェクトの記述形式を選択します。 ここでは、「画面部品ごとに複数のイベントをオブジェクト形式で記述」を選択します。

[イベントオブジェクト定義]ダイアログボックスで[OK]ボタンをクリックすると、設定したイベントオブジェクト名が[イベント処理定義]ダイアログボックスに反映されます。

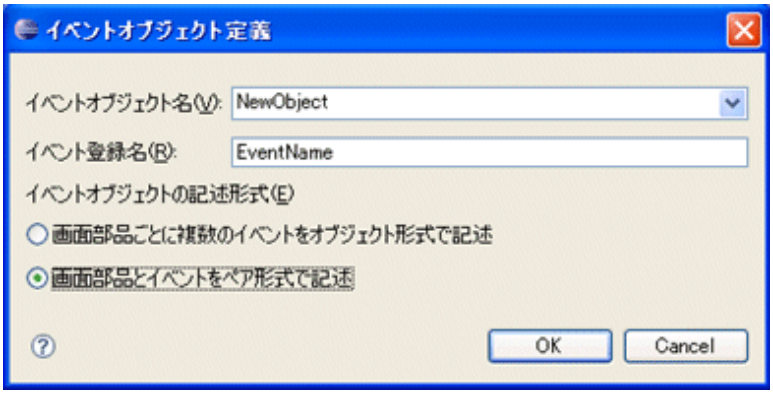
[イベント処理定義]ダイアログボックスで[追加]ボタンをクリックすると、[イベント処理定義の追加]ダイアログボックスが表示されます。
[イベント処理定義の追加]ダイアログボックスの指定項目は、画面部品に直接定義する場合と同じです。

[イベント処理定義の追加]ダイアログボックスで[OK]ボタンをクリックすると、設定したイベント名および関数名が[イベント処理定義]ダイアログボックスのイベント定義一覧に反映されます。
イベント処理定義は、繰り返して追加することができます。

イベント定義一覧の内容を確認したあと、[OK]ボタンまたは[適用]ボタンをクリックし、イベント定義一覧の情報をソースに反映します。

画面部品とイベントの組ごとに定義

[イベント処理定義]ダイアログボックスで[定義]ボタンをクリックすると、[イベントオブジェクト定義]ダイアログボックスが表示されます。



[イベントオブジェクト定義]ダイアログボックスで、以下の項目を指定します。

項目	説明
イベントオブジェクト名	JavaScriptに定義されているイベントオブジェクト名が候補として表示されるので、その中から選択します。 また、イベントを登録するイベントオブジェクトの名前を直接入力することもできます。
イベント登録名	イベント登録名を直接入力して指定します。
イベントオブジェクトの記述形式	イベントオブジェクトの記述形式を選択します。 ここでは、「画面部品とイベントをペア形式で記述」を選択します。

[イベント処理定義の追加]ダイアログボックスで[OK]ボタンをクリックすると、設定したイベントオブジェクト名が[イベント処理定義]ダイアログボックスに反映されます。

[イベント処理定義]ダイアログボックスで[追加]ボタンをクリックすると、[イベント処理定義の追加]ダイアログボックスが表示されます。
[イベント処理定義の追加]ダイアログボックスの指定項目は、画面部品に直接定義する場合と同じです。

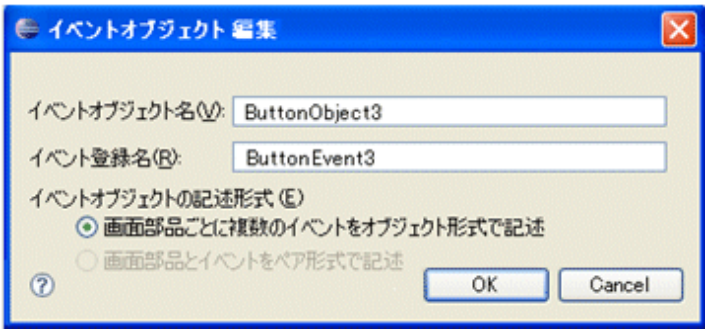
[イベント処理定義の追加]ダイアログボックスで[OK]ボタンをクリックすると、設定したイベント名および関数名が[イベント処理定義]ダイアログボックスのイベント定義一覧に反映されます。
イベント処理定義は、繰り返して追加することができます。

イベント定義一覧の内容を確認したあと、[OK]ボタンまたは[適用]ボタンをクリックし、イベント定義一覧の情報をソースに反映します。

ポイント

イベントオブジェクトの編集

[イベントオブジェクト編集]ダイアログボックスを利用して、既存のイベントオブジェクトのイベントオブジェクト名やイベント登録名を変更することができます。ただし、イベントオブジェクトの記述形式は変更できません。
[イベントオブジェクト編集]ダイアログボックスは、[イベント処理定義]ダイアログボックスの[定義方法]の[編集]ボタンをクリックすると表示されます。以下に、[イベントオブジェクト編集]ダイアログボックスの表示例を示します。



F.10.3 機能付加部品バインディング定義

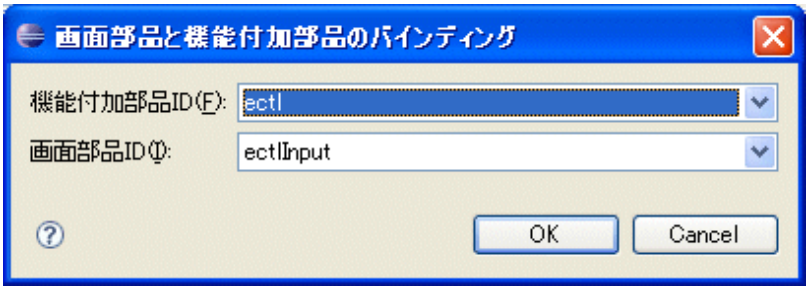
機能付加部品バインディング定義では、画面部品と機能付加部品のバインディングを定義します。

アウトラインビューでrcf:id属性が設定されている機能付加部品を選択し、コンテキストメニューの[機能付加部品バインディング定義]コマンドを選択すると、[画面部品と機能付加部品のバインディング]ダイアログボックスが表示されます。
[画面部品と機能付加部品のバインディング]ダイアログボックスでは、以下のバインディングを定義することができます。

- 画面部品と入力支援機能付加部品のバインディング
- 画面部品とフォーカス制御機能付加部品・グループ化機能付加部品のバインディング

画面部品と入力支援機能付加部品のバインディング

以下に、[画面部品と機能付加部品のバインディング]ダイアログボックスで、画面部品と入力支援機能付加部品のバインディングを定義する例を示します。
入力支援機能付加部品を選択すると、以下の形式のダイアログボックスが表示されます。
この例では、TextInput画面部品に、入力支援機能を付加するAutoCompleter機能付加部品をバインディングします。機能付加部品IDに「ectl」を、画面部品IDに「ectlInput」を選択します。



以下の表に、[画面部品と機能付加部品のバインディング]ダイアログボックスの項目を説明します。

項目	説明
機能付加部品ID	画面上に定義されている機能付加部品が候補として表示されるので、画面部品に付加したい機能付加部品のIDを選択します。
画面部品ID	画面上に定義されている機能付加対象の画面部品が候補として表示されるので、機能を付加したい画面部品のIDを選択します。

画面部品とフォーカス制御機能付加部品・グループ化機能付加部品のバインディング

以下に、[画面部品と機能付加部品のバインディング]ダイアログボックスで、画面部品とフォーカス制御機能付加部品・グループ化機能付加部品のバインディングを定義する例を示します。
フォーカス制御機能付加部品またはグループ化機能付加部品を選択すると、以下の形式のダイアログボックスが表示されます。
この例では、RadioButton画面部品をグループ化するために、グループ化機能を付加するRadioButtonGroup機能付加部品をバインディングします。機能付加部品IDに「rbGroup2」を選択し、画面部品IDに「radioButton21」、「radioButton22」を指定します。

以下の表に、[画面部品と機能付加部品のバインディング]ダイアログボックスの項目を説明します。

項目	説明
機能付加部品ID	画面上に定義されている機能付加部品が候補として表示されるので、画面部品に付加したい機能付加部品のIDを選択します。
画面部品ID	画面上に定義されている機能付加対象の画面部品が候補として表示されるので、機能を付加したい画面部品のIDを左のリストボックスで指定します。 画面部品IDを指定するには、[追加]/[削除]/[上へ]/[下へ]ボタンを利用します。
[追加]ボタン	右側のリストボックスで選択された画面部品を左側のリストボックスに移動します。
[削除]ボタン	左側のリストボックスで選択された画面部品を右側のリストボックスに移動します。
[上へ]ボタン	左側のリストボックスで選択された画面部品を上へ移動します。
[下へ]ボタン	左側のリストボックスで選択された画面部品を下へ移動します。

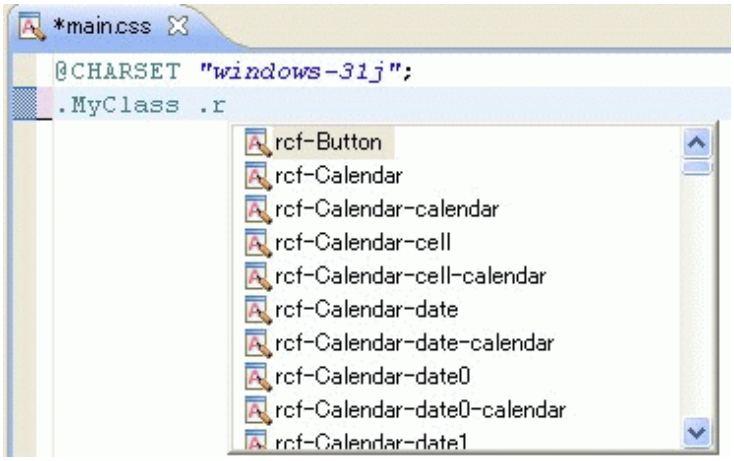
F.11 CSSファイル編集機能

Ajaxフレームワークには、CSSファイルをより簡単に編集できるための、Ajax CSSエディタを用いたCSSファイル編集機能があります。
CSSファイル編集機能では、CSSファイルの入力支援機能を提供しています。

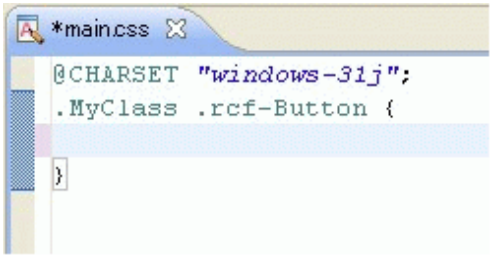
CSSの入力支援

CSSファイルをAjax CSSエディタで開き、下記のコマンドを入力することで画面部品のスタイルの設定に関する入力支援を利用することができます。

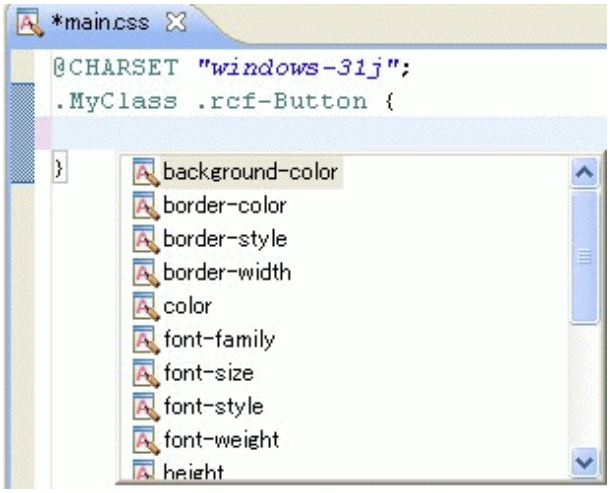
- クラス名
  キーボードで「.r」を入力し、Ctrl + Spaceを押すことによって、設定可能な画面部品のクラス名の一覧が入力候補として入力支援ダイアログにリスト表示されます。



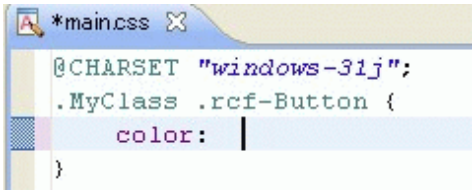
ここで任意のクラス名を指定することで、画面部品のクラス名と「{}」が入力されます。



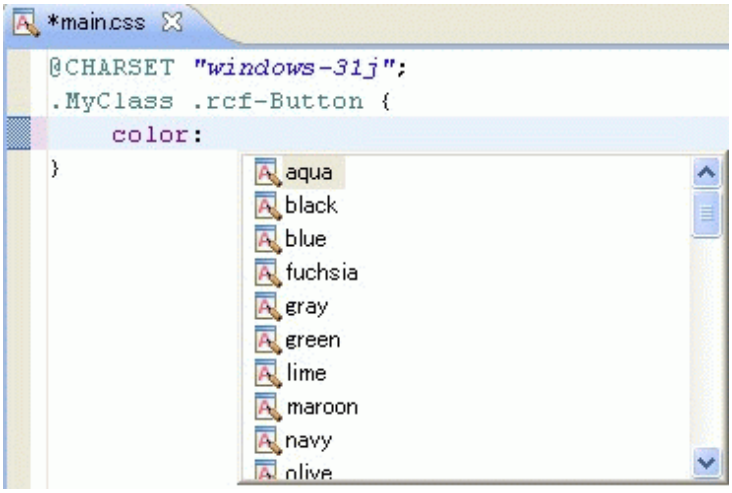
- スタイルプロパティ名
  画面部品のクラス名を指定したスタイルの宣言部({ }の中)でCtrl + Spaceを押すことによって、設定可能なスタイルプロパティ名の一覧が入力候補として入力支援ダイアログにリスト表示されます。



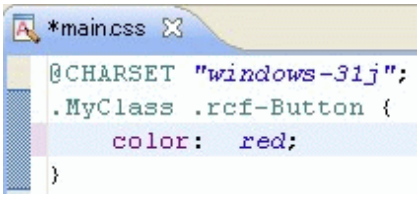
ここで任意のスタイルプロパティ名を指定することで、スタイルプロパティ名と「:」が入力されます。



- スタイルプロパティ値
  入力した「スタイルプロパティ名: 」の後でCtrl + Spaceを押すことによって、スタイルプロパティ名に設定可能な値の一覧が入力候補として入力支援ダイアログにリスト表示されます。



ここで任意の値を指定することで、スタイルプロパティの値と「;」が入力されます。



ポイント

スタイルプロパティに設定可能な値が数値の場合、入力支援ダイアログには設定可能な単位がリスト表示されます。

注意

- 画面部品の入力支援ダイアログ表示時に、ユーザーが入力した文字や値がダイアログの内容と一致する場合、ダイアログの内容は自動的にフィルタリングされ、入力候補を絞ることができます。ユーザーの入力した文字や値がダイアログの内容と異なる場合、入力支援ダイアログは自動的に閉じられます。
- 画面部品のクラス名、およびスタイルプロパティ名については、「UI部品リファレンス」の「第2章 画面部品」を参照してください。

付録G　スクレイピングツール

本付録では、スクレイピングツールについて説明します。
スクレイピングツールは、EclipseおよびInterstage Studioワークベンチのプラグインとして動作します。

注意

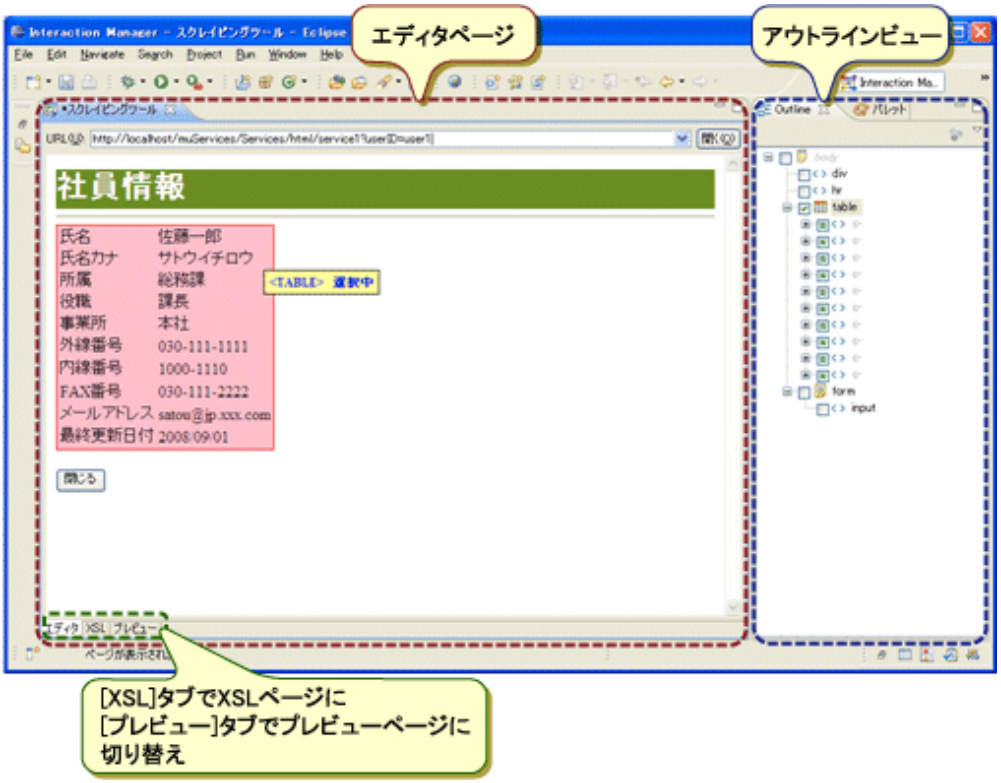
本付録では、Eclipseを使用した場合の操作方法を説明しています。Interstage Studioワークベンチを使用する場合は、本付録に記載されている英語のメニュー名やコマンド名などを日本語の名称に読み替えてください。

G.1　スクレイピングツールの概要

スクレイピングツールは、Ajaxフレームワークアプリケーションで利用するWebアプリケーションをGUIベースでスクレイピングし、XSLファイルを作成するツールです。

以下に、スクレイピングツールのビューの概要を説明します。

図G.1 スクレイピングツールのビュー

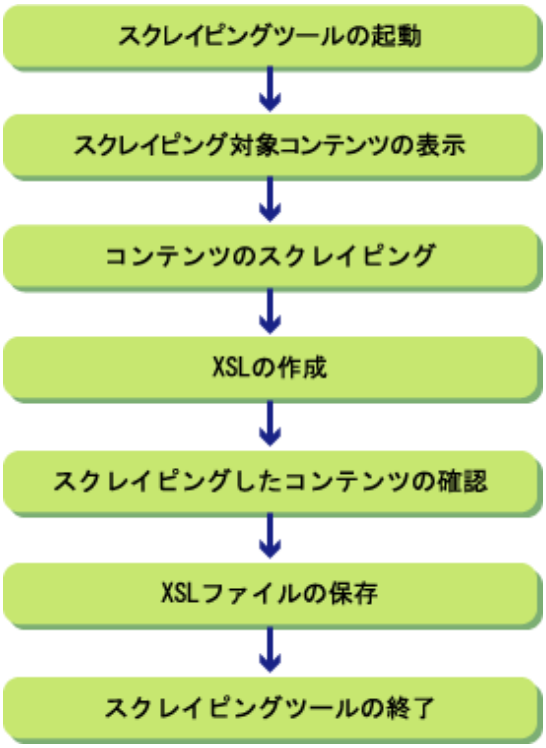


- エディタページ
  スクレイピング対象とするWebコンテンツをGUI操作で編集するページです。
- XSLページ
  エディタページの編集状態から作成されるXSLファイルを表示するページです。
- プレビューページ
  エディタページの編集情報に基づいて、スクレイピング結果を表示するページです。
- アウトラインビュー
  スクレイピング対象のコンテンツのHTML構成を表示するビューです。
  スクレイピングするタグを一覧から選択することもできます。

G.2 スクレイピングツールの操作手順

以下に、スクレイピングツールの操作手順を説明します。

図G.2 スクレイピングツールの操作手順



1. スクレイピングツールの起動
   マッシュアップフレームワークを利用するプロジェクトを選択し、コンテキストメニューの[スクレイピングツール]を選択して、スクレイピングツールを起動します。

2. スクレイピング対象コンテンツの表示
   スクレイピングツールのエディタページでWebコンテンツのURLを入力し、[開く]ボタンをクリックします。
   スクレイピングの対象コンテンツがHTML表示エリアに表示されます。

3. コンテンツのスクレイピング
   表示されているコンテンツから、スクレイピングしたいタグを選択します。
   任意の箇所をマウスカーソルでポイントすると、その位置のHTMLタグが[スクレイピング対象登録]メニューに表示されるので、スクレイピングする場合は、[スクレイピング対象登録]メニューをクリックします。解除する場合は、[スクレイピング対象登録]メニューを再クリックします。
   スクレイピングされたコンテンツは、背景色が変更されて表示されます。
   詳細は、「G.3 エディタページ」を参照してください。

4. XSLの作成
   必要なタグがスクレイピングできたら、[XSL]タブをクリックして、XSLページを表示します。このとき、XSLが作成されます。
   作成されたXSLが、XSLページに表示されます。
   詳細は、「G.4 XSLページ」を参照してください。
   なお、XSLページに表示されたXSLデータは、編集できません。

5. スクレイピングしたコンテンツの確認
   [プレビュー]タブをクリックして、プレビューページを表示します。
   スクレイピングしたWebコンテンツが表示されるので、スクレイピングしたコンテンツが正しいかを確認します。
   詳細は、「G.5 プレビューページ」を参照してください。
   スクレイピングをやり直す場合は、[エディタ]タブをクリックして、3. の操作に戻ります。HTML表示エリアには、コンテンツがスクレイピングされた状態で表示されます。

6. XSLファイルの保存
   コンテンツが確認できたら、プレビューページの[保存]ボタンをクリックします。

[保存]ダイアログボックスが表示されるので、保存先のプロジェクトとファイル名を指定して、保存します。
詳細は、「G.5 プレビューページ」を参照してください。
なお、ファイルの拡張子(.xsl)と保存場所(xslフォルダ)は、変更できません。

7. スクレイピングツールの終了
   ツールの閉じるボタンをクリックして、スクレイピングツールを終了します。
   ツールの終了時に編集内容が保存されていない場合は、確認のメッセージボックスが表示されるので、そのままツールを終了するか、終了をキャンセルするかを選択します。

注意

Webコンテンツが表示されない場合の対処
インターネット上のWebコンテンツをスクレイピングする場合、インターネットへの接続の設定が必要です。
スクレイピングツールのエディタページおよびプレビューページでWebコンテンツが表示されない場合は、以下の方法でインターネットへの接続を設定します。

1. [Window]メニューから[Preferences]を選択します。
2. [Preferences]ダイアログボックスの設定項目で、[General] > [Network Connections]を選択します。
3. [Manual proxy configuration]を選択し、環境に応じたプロキシサーバの設定を行います。

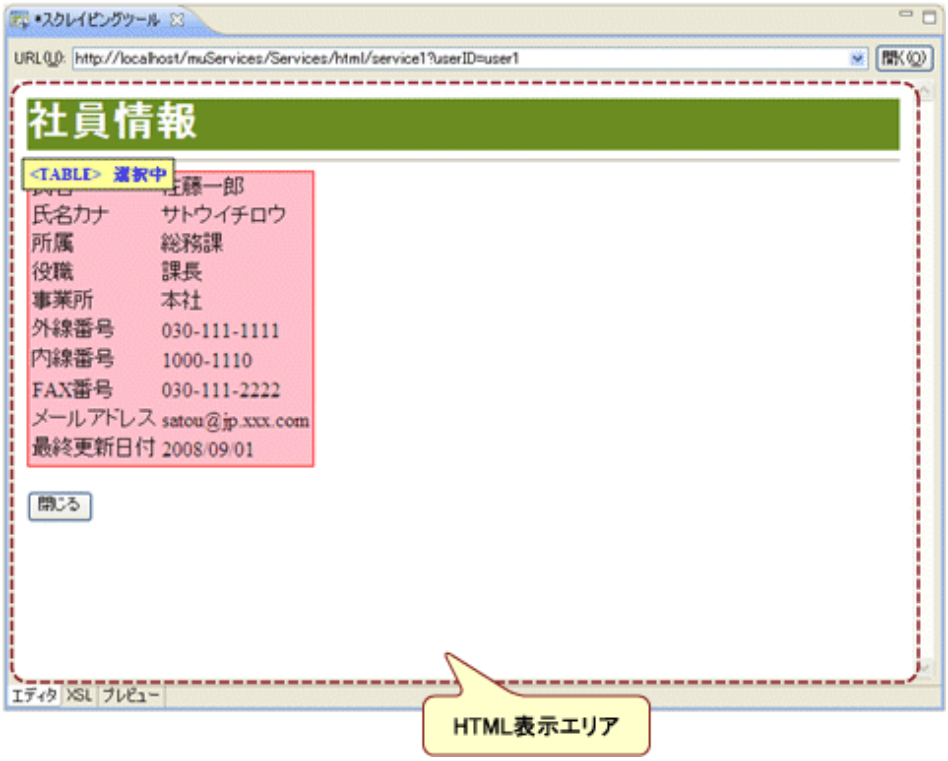
G.3 エディタページ

エディタページは、スクレイピングツールの起動時、および[エディタ]タブをクリックした場合に表示されます。

エディタページでは、表示されているコンテンツから、スクレイピングしたいタグを選択します。
コンテンツ上で任意の箇所をマウスカーソルでポイントすると、その位置のHTMLタグの[スクレイピング対象登録]メニューが「未選択」の状態で表示されるので、スクレイピングする場合は、[スクレイピング対象登録]メニューをクリックします。クリックすると、[スクレイピング対象登録]メニューが「選択中」の状態に切り替わり、スクレイピング対象の領域の背景色が変更されて表示されます。選択を解除する場合は、[スクレイピング対象登録]メニューを再クリックします。

以下に、エディタページの画面例を示します。

図G.3 エディタページ



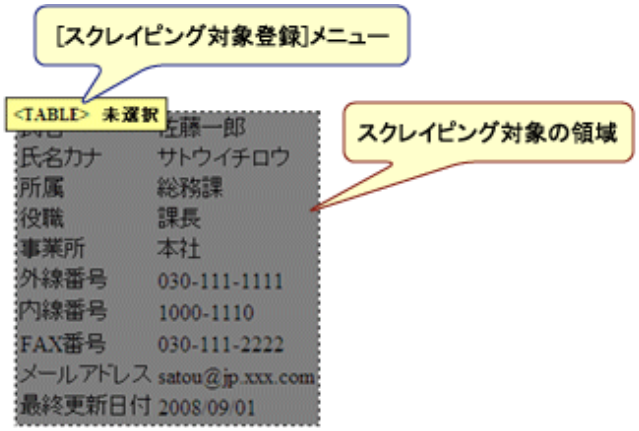
以下に、画面の項目について説明します。

項目	説明
URL	スクレイピング対象とするWebコンテンツのURLを指定します。 URLは、選択リストから選択することができます。選択リストには、過去にスクレイピングツールで表示したことのあるURLが新しい順に50件まで表示されます。選択したURLは、直接編集することができます。
[開く]ボタン	URLに指定したWebコンテンツを表示する場合にクリックします。
HTML表示エリア	スクレイピングの対象コンテンツがHTML表示エリアに表示されます。 コンテンツの文字コードは、UTF-8に変換されて表示されます。 なお、スクレイピングされたコンテンツの領域は、背景色が変更されて表示されます。

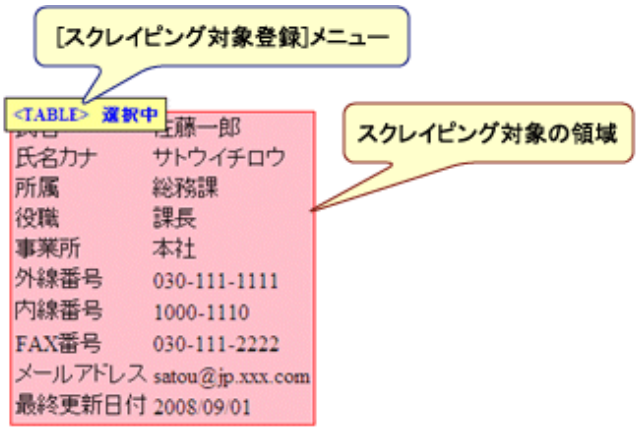
スクレイピング対象の領域について

スクレイピング対象の領域は、以下のように、枠線と背景色が変わります。

- スクレイピング対象として選択可能な領域をマウスオーバーした場合
  枠線:黒色の点線
  背景色:グレー



- スクレイピング対象として選択済みの領域の場合
  枠線:赤色の実線
  背景色:ピンク



- 選択できない領域の場合
  枠線:なし
  背景色:なし

注意

スクレイピング対象領域の下位階層で背景色が設定されている場合、その背景色が有効になります。

[スクレイピング対象登録]メニュー

[スクレイピング対象登録]メニューは、表示されているコンテンツのタグ領域をマウスオーバーすると表示されます。
[スクレイピング対象登録]メニューには、タグと選択状態が表示されて、ウィンドウをクリックするごとに、選択状態が切り替わります。
選択状態には、以下の2種類があります。

- 未選択 <TABLE> 未選択
  選択可能なタグで、未選択な場合、黒字で表示されます。また、ウィンドウの背景色は黄色になります。
- 選択中 <TABLE> 選択中 選択可能なタグで、選択中の場合、青字で表示されます。また、ウィンドウの背景色は黄色になります。

なお、マウスオーバーした領域よりも上位階層が選択中の場合、および選択不可能な領域をマウスオーバーした場合で、選択可能な上位階層の領域が存在する場合は、上位階層の情報がウィンドウに表示されます。

[認証]ダイアログボックス

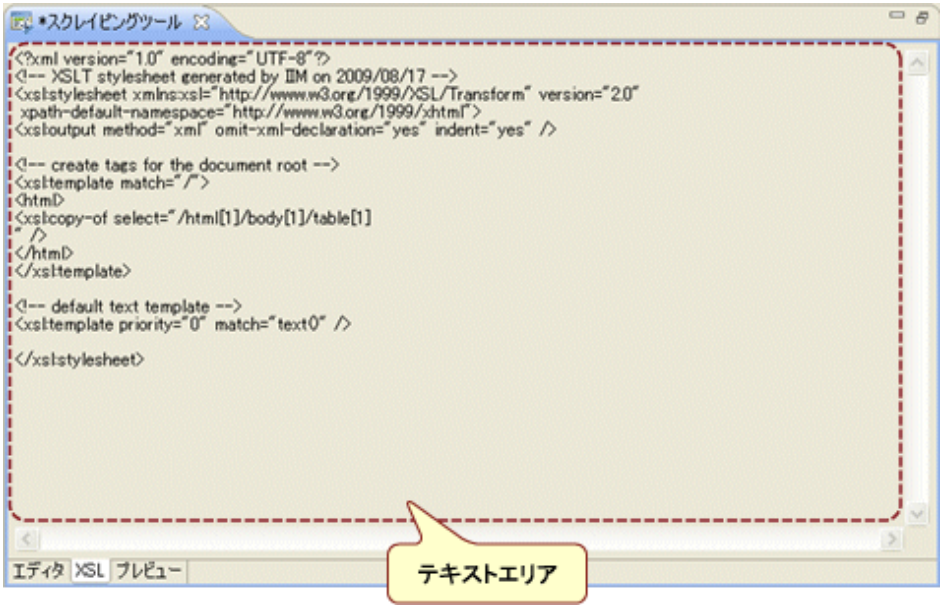
[認証]ダイアログボックスは、URLに基本認証を必要とするURLを入力し、[開く]ボタンをクリックした場合に表示されます。
[認証]ダイアログボックスでは、ユーザー名およびパスワードを指定します。

G.4 XSLページ

XSLページは、[XSL]タブをクリックした場合に表示されます。

以下に、XSLページの画面例を示します。

図G.4 XSLページ



以下に、画面の項目について説明します。

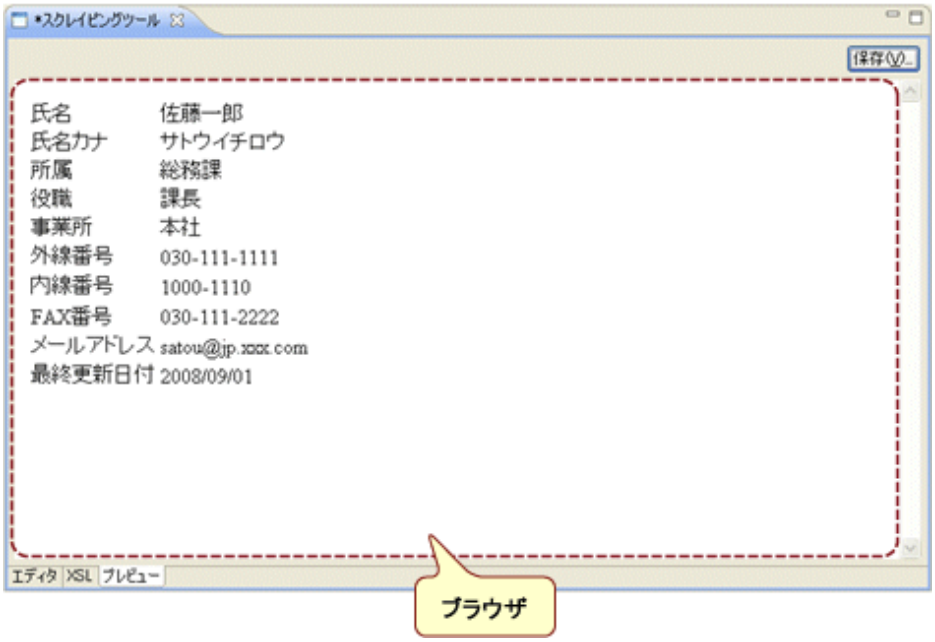
項目	説明
テキストエリア	エディタページの編集状態から作成されるXSLファイルが表示されます。 なお、表示されたXSLデータは、編集できません。

G.5 プレビューページ

プレビューページは、[プレビュー]タブをクリックした場合に表示されます。

以下に、プレビューページの画面例を示します。

図G.5 プレビューページ

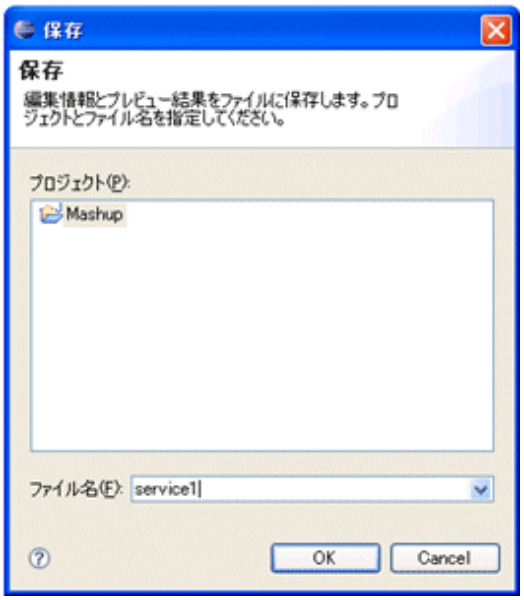


以下に、画面の項目について説明します。

項目	説明
ブラウザ	エディタページの編集情報に基づいて、スクレイピング結果が表示されます。 スクレイピング結果の文字コードは、UTF-8に変換されて表示されます。
[保存]ボタン	スクレイピングしたコンテンツをXSLファイルに保存する場合にクリックします。 クリックすると、[保存]ダイアログボックスが表示されます。 なお、エディタページで一度もスクレイピング対象を編集していない場合は、[保存]ボタンはアクティブになりません。

[保存]ダイアログボックス

[保存]ダイアログボックスは、プレビューページの[保存]ボタンをクリックした場合に表示されます。
以下に、[保存]ダイアログボックスの画面例を示します。



以下に、画面の項目について説明します。

項目	説明
プロジェクト	プロジェクトの一覧が表示されるので、保存先のプロジェクトを選択します。 なお、選択できるプロジェクトは1つだけです。
ファイル名	保存するファイル名を指定します。 ファイル名は、選択リストから選択することができます。選択リストには、[プロジェクト]で選択したプロジェクト内のXSLファイルが表示されます。 選択したファイル名は、直接編集することができます。 指定したファイル名には、拡張子「.xsl」が自動的に付加されます。 なお、拡張子は変更できません。

ポイント

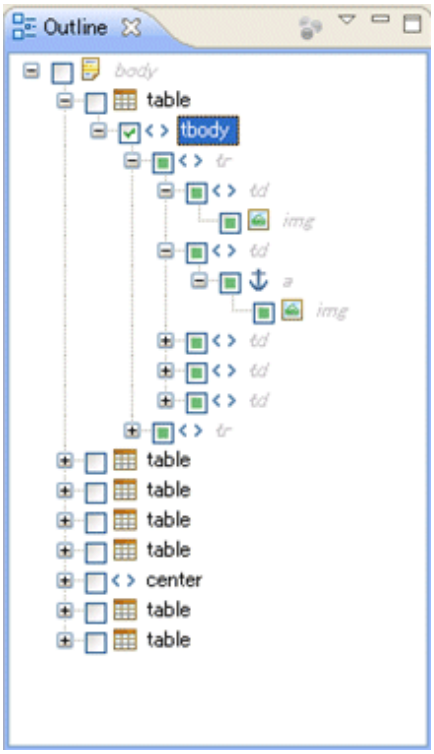
XSLファイルを保存すると、同じファイル名でスクレイピングしたコンテンツのXMLファイルも保存されます。
スクレイピングツールを終了したあとで、再度スクレイピングしたコンテンツを確認したい場合、保存されたXMLファイルを表示することで確認できます。また、XMLファイルのデータを利用することで、モックアップやAjaxページエディタでスクレイピングしたコンテンツを確認することができます。

G.6 アウトラインビュー

エディタページにスクレイピング対象のコンテンツが表示されると、コンテンツのHTML構成がアウトラインビューに表示されます。
アウトラインビューでは、スクレイピングするタグを一覧から選択することができます。

以下に、アウトラインビューの画面例を示します。

図G.6 アウトラインビュー



以下に、画面の項目について説明します。

項目	説明
チェックボックス	スクレイピングするタグのチェックボックスをチェックします。スクレイピングを解除する場合は、チェックを外します。 なお、チェックボックスの状態は、タグの選択状態によって、以下のように異なります。 ・ チェックなし、かつ、タグ名が黒の標準フォント スクレイピング対象として選択可能 ・ チェックあり、かつ、タグ名が黒の標準フォント スクレイピング対象として選択中で、選択の解除が可能 ・ チェックなし、かつ、タグ名がグレーの斜体フォント スクレイピング対象として選択不可 ・ チェックあり、かつ、タグ名がグレーの斜体フォント 上位階層のタグがスクレイピング対象として選択中 チェックボックスのチェック状態を変更した場合、その下位階層のチェック状態も同様に変更されます。
タグ名	タグ名(黒の標準フォント)を選択すると、エディタページに表示中のコンテンツでは、対応するタグの領域が枠線で囲まれ、背景色が変わります。

付録H　互換情報

本付録では、旧バージョンのAjaxフレームワークとの互換情報について説明します。

H.1　通信フレームワークの互換情報

ここでは、通信フレームワークを利用する場合の互換情報について説明します。

H.1.1　共通の互換情報

エラーオブジェクトのerrorCodeプロパティについて

通信フレームワークでエラーが発生した場合に、エラーコードを通知するエラーオブジェクト内のerrorCodeプロパティを数値で通知するように修正しました。

V9.0.1以前のバージョンでは、サーバが停止している場合は、errorCodeプロパティに「RCF0700」が通知されていましたが、ほかのエラーと同様に文字列「RCF」を取り除いた数値として通知するように修正しました。

V9.0.1以前と同じ動作にしたい場合は、RCF_configオブジェクトの変数connection_compatible_errorCodeRCF0700にtrueを指定してください。変数connection_compatible_errorCodeRCF0700にfalseを指定、または、変数connection_compatible_errorCodeRCF0700の指定を省略した場合は、修正後の動作となります。
RCF_configオブジェクトの詳細については、「2.3.2 Ajaxフレームワークの動作定義」を参照してください。

エラーオブジェクトのnumberプロパティについて

通信フレームワークでエラーが発生した場合のエラーコードは、エラーオブジェクト内のerrorCodeプロパティに通知するように統一されました。

V9.0.1以前のバージョンでは、UjiRequest.send関数のtimeoutプロパティで指定したタイムアウト時間を超過した場合は、エラーコードの参照に、numberプロパティを使用する必要がありましたが、errorCodeプロパティで参照できるように修正しました。

エラーオブジェクトのnumberプロパティは、アプリケーションの動作互換維持のための情報として、V9.0.1以前のバージョンと同様にタイムアウト時間を超過した場合にだけ通知します。エラーコードを参照する際には、errorCodeプロパティを使用してください。

web.xmlへのload-on-startupの記載について

ロードバランサを利用して、1つのクライアントからのリクエストを複数のサーバに振り分けている場合には、エントリサーブレットをサーブレットコンテナ起動時にロードする必要があります。
設定方法については、「5.3.2 web.xmlの設定」を参照してください。

H.1.2　Apcoordinator連携機能の互換情報

通信先のサーバが停止された場合の動作の違い

Ajaxフレームワークで作成したアプリケーションがブラウザ上で動作している場合、通信先のサーバが停止されたときにエラー発生時のコールバック関数(errorHandler)が呼び出されない問題を修正しています。

この修正により、V9.0のAjaxフレームワークで作成したアプリケーションを当該バージョンのAjaxフレームワークで動作させた場合、以下のように動作が変わる可能性があります。

- 通信先のサーバが停止された場合、エラー発生時のコールバック関数(errorHandler)の呼出しが行われるようになります。

V9.0.0のAjaxフレームワーク使用時と同じ動作にしたい場合は、RCF_configオブジェクトの変数apcConnection_compatibleに「9.0」を指定してください。変数apcConnection_compatibleに「9.0」以外の値を指定、または、変数apcConnection_compatibleの指定を省略した場合は、修正後の動作となります。
RCF_configオブジェクトの詳細は、「2.3.2 Ajaxフレームワークの動作定義」を参照してください。

H.2　クライアントフレームワークの互換情報

ここでは、クライアントフレームワークを利用する場合の互換情報について説明します。

H.2.1 UI部品の互換情報

UI部品の表示サイズの違い

V9.1以前のバージョンでは、Internet Explorerの場合、以下の部品に指定したサイズ(height)と表示された部品のサイズが異なっていました。

- TabPanel
  タブ表示位置によりheightより1～2px高く表示されます。
- TableView/TableEdit
  heightより(列ヘッダの高さ×(列ヘッダの行数-1))+2px分、高く表示されます。
- DataGrid
  heightより(列ヘッダの高さ×(列ヘッダの行数-1))+(フィルタ行の高さ×フィルタ行の行数)+2pxまたは3px分、高く表示されます。

本バージョンでは、上記部品が指定したサイズどおりに表示されるよう修正しています。

Internet Explorer 9 互換、Internet Explorer 10 互換、およびInternet Explorer 11 互換で、V9.1以前と同じ表示にしたい場合は、RCF_config オブジェクトの変数 UIComponent_compatible_viewsize に「 9.1.0 」を指定してください。変数UIComponent_compatible_viewsizeに「9.1.0」以外の値を指定、または、変数UIComponent_compatible_viewsizeの指定を省略した場合は、修正後の動作となります。
また、V9.0.1からバージョンアップした場合に、テーブル部品(TableView、TableEdit、およびDataGrid)の表示サイズを変更していないにもかかわらず、従来は表示されていなかった横スクロールバーが表示されることがことがあります。その場合には、RCF_configオブジェクトの変数UIComponent_compatible_viewsizeに「9.0.1」を指定してください。
RCF_configオブジェクトの詳細は、「2.3.2 Ajaxフレームワークの動作定義」を参照してください。

参考

Internet Explorerの各バージョンとInterstage Interaction Managerのサポートバージョンの対応は以下のとおりです。

ブラウザのバージョン	Interstage Interaction Managerのバージョン
Internet Explorer 9 互換	V9.1.1以降
Internet Explorer 10 互換	V10.0.0以降
Internet Explorer 11 互換	V10.1.0
Internet Explorer 9 標準	V10.0.1以降
Internet Explorer 10 標準	V10.0.1以降
Internet Explorer 11 標準	V10.1.2以降

プロパティの型チェック

一部のUI部品でプロパティの型チェックを強化しています。
本書に記載された型以外を指定していた場合、エラーとなります。その場合には、RCF_configオブジェクトの変数UIComponent_compatible_errorに「9.1.1」を指定してください。

ComboBox

- 非活性時のボタン
  部品が非活性の場合にボタンの表示がグレーになります。従来どおりの表示にしたい場合は、RCF_configオブジェクトの変数UIComponent_compatibleに以下を指定してください。

```
ComboBox: {
   buttonDisabled: "9.1.1"
}
```

DataGrid

- 文字列選択
  DataGrid上での文字列の選択操作ができるようになります。文字列の選択操作を抑止したい場合は、RCF_configオブジェクトの変数UIComponent_compatibleに以下を指定してください。

```
DataGrid: {
   selection: "9.0.1"
}
```

  また、DataGrid上で選択した文字列のCtrl+Cでのコピーが可能となりました。DataGrid上で選択した文字列のコピーを抑止したい場合は、文字列の選択操作を抑止して回避してください。

- セルのスタイル
  詳細行含むデータ行のセルに対し、white-spaceは有効になりません。

スタイルシートの適用順位

V9.1.1までは、コンテンツで定義しているスタイルシートよりも、Ajaxフレームワーク(UI部品)の初期値を定義したスタイルシートが優先される場合がありました。
Ajaxフレームワーク(UI部品)のスタイルシートは部品表示のベースであり、その上にコンテンツのスタイルを適用します。そのため、V10.0.0以降では、常にコンテンツで定義したスタイルシートが優先されるように修正しています。

V9.1.1までのスタイルシートの優先順位で利用する場合には、RCF_configオブジェクトの変数StyleSheet_compatibleに"9.1.1"を指定してください。

付録I　開発資産の移行

本付録では、旧バージョンのInterstage Interaction Manager 開発環境パッケージで開発した開発資産を、本バージョンの開発環境パッケージで取り扱うことのできるプロジェクトに移行するための手順について説明します。

I.1　開発資産の移行

ここでは、開発資産の移行について説明します。

I.1.1　Ajaxフレームワークプロジェクトの新規作成

移行先となるAjaxフレームワークプロジェクトを作成します。

以下に、Ajaxフレームワークプロジェクトの作成方法を示します。

- Interstage Studioワークベンチへ移行する場合
  「5.7 Interstage Studioワークベンチを利用した開発」を参照してください。
- Eclipseへ移行する場合
  「5.8 Eclipseを利用した開発」を参照してください。

注意

Interstage Studio ワークベンチは、デフォルトの設定として、アプリケーションの配備時にInterstage Java EE検証が行われるように設定されています。
この設定が行われている場合には、Ajaxフレームワークを利用したアプリケーションの配備時にエラーが表示されます。
これは、Ajaxフレームワークを利用したアプリケーションの実行時に必要なライブラリには、Ajaxフレームワークが使用しない機能も含まれており、これらの機能が必要とするライブラリがクラスパスに設定されていないためにエラーとなります。

Interstage Studio ワークベンチを使用して、Ajaxフレームワークを利用したアプリケーションを開発する場合は、プロジェクトのInterstage Java EE検証の設定について、以下の値となるように指定してください。

- [Interstage Java EE検証を自動的に実行する]が選択されている場合は、選択を解除して指定を取り外します。
- [JSPをチェックする]が選択されている場合は、選択を解除して指定を取り外します。
- [配備前にチェックする]が選択されている場合は、選択を解除して指定を取り外します。
- [差分ビルドではチェックしない]が選択されていない場合は、選択します。

開発資産を移行した場合、スクレイピングツールで過去に表示したことのあるURLの一覧は移行されません。また、マッシュアップ定義ファイルエディタでサービスに関する定義を行う際に表示される、[URL]および[XSL]項目の選択リストの情報は移行されません。

I.1.2　開発資産の移行手順

移行先となるAjaxフレームワークプロジェクトに、既存プロジェクトに格納されたアプリケーションの開発資産をインポートします。

以下に、開発資産のインポートの手順を示します。

- Interstage Studioワークベンチへ移行する場合
  以下に、Interstage Studioワークベンチへインポートする場合の手順を示します。
  1. [ファイル]メニューで[インポート]を選択します。
  2. [一般] > [ファイルシステム]を選択し、[次へ]ボタンをクリックします。
  3. 必要な資産を選択し、[終了]ボタンをクリックします。
- Eclipseへ移行する場合
  以下にEclipseへインポートする場合の手順を示します。
  1. [File]メニューで[Import]を選択します。

2. [General] > [File System]を選択し、[Next]ボタンをクリックします。

3. 必要な資産を選択し、[Finish]ボタンをクリックします。

ポイント

インポートが必要なアプリケーションの開発資産とは、ソースファイル、設定ファイル、画面定義ファイル、リソースファイルなど、アプリケーション開発で編集が必要なファイルおよびアプリケーション実行時に必要なファイルです。

I.2 HTML/JSPファイルの移行

ここでは、HTML/JSPファイルの移行について説明します。

Interstage Studioで作成したHTML/JSPファイルを移行する際には、Ajaxページエディタで問題なく取り扱えるようにするために、以下の手順で移行する必要があります。

1. Ajaxページエディタの設計ビューを使用して、既存のHTML/JSPファイルを開きます。
2. 定義されているAjaxフレームワークのUI部品、HTMLタグを選択し、マウスまたはカーソルキーを使用して、別の行桁位置に移動します。
3. 移動したUI部品、HTMLタグを、マウスまたはカーソルキーを使用して、元の行桁位置に戻します。

Ajaxページエディタの設計ビューでは、実行イメージを確認しながらWYSIWYGで業務画面を編集するために、HTML/JSPファイルに定義するAjaxフレームワークのUI部品、HTMLタグを絶対座標で取り扱います。

Ajaxページエディタの設計ビューで既存のHTML/JSPファイルを編集する場合、定義されているAjaxフレームワークのUI部品、HTMLタグを移動、リサイズするときに絶対座標を指定します。このとき、ブラウザによって絶対座標の再計算が行われ、絶対座標が指定されていないほかの部品の位置が移動する場合があります。
この部品の位置移動を回避するには、上記の手順でHTML/JSPファイルを移行してください。

参考

以下の場合には、上記の手順で既存のHTML/JSPファイルを移行する必要はありません。

- 定義されているAjaxフレームワークのUI部品、HTMLタグすべてについて、絶対座標が指定されている場合
- Ajaxページエディタの設計ビューを使用しない場合

付録J　エラーメッセージ

本付録では、Ajaxフレームワークが出力するメッセージとその対処について説明します。

J.1　メッセージの形式

ここでは、メッセージの形式について説明します。

[メッセージ番号] [メッセージ種別] [メッセージ本文]

メッセージ番号

各メッセージに一意に付加されたメッセージの識別番号です。

メッセージ種別

出力メッセージの原因レベルです。以下のレベルのメッセージが出力されます。

- E: エラー
- W: 警告
- I: 情報

メッセージ本文

メッセージ本文です。

J.2　クライアントのエラーメッセージ

ここでは、クライアントフレームワーク、通信フレームワークのクライアントJavaScript、およびマッシュアップフレームワークのクライアントJavaScriptが出力するエラーメッセージについて説明します。

J.2.1　UI部品に関するメッセージ

ここでは、UI部品に関するエラーメッセージについて説明します。

RCF10100

The component specified in event definition is not found. id=%1

メッセージの意味

イベントリスナの登録において、イベント定義に記述された部品が見つかりません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 部品ID

利用者の処置

イベント定義に記述された部品のIDに誤りがないかを確認してください。
また、部品の初期化が完了する前にその部品のイベントリスナを登録した場合も、本メッセージが出力されます。
以下を確認してください。

- FragmentContainerのフラグメントHTMLに記述されている部品の場合
  FragmentContainerの有効化が完了してから、イベントの登録を行っているかを確認してください。詳細は、「UI部品リファレンス」の「2.2.5 FragmentContainer」を参照してください。

- その他の部品の場合
  Ajaxフレームワークの初期化が完了してから、イベントの登録を行っているかを確認してください。

RCF10101

The event listener specified in event definition is invalid. id=%1 event=%2

メッセージの意味

イベントリスナの登録において、イベント定義に記述されたイベントリスナが正しくありません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 部品ID
%2: イベント名

利用者の処置

イベント定義に記述されたイベントリスナが関数であるかを確認してください。

RCF11000

Duplicate component id. id=%1

メッセージの意味

2つ以上のUI部品でIDが重複しています。

メッセージ種別

E: エラー

パラメーターの意味

%1: 重複しているID

利用者の処置

UI部品のIDが重複しないように修正してください。

RCF11001

Invalid attribute value. name=rcf:%1 value=%2

メッセージの意味

プロパティに指定された初期値に誤りがあります。
以下の原因が考えられます。

- プロパティのデータ型と指定された初期値のデータ型が異なる場合
- プロパティのデータ型がobject型で、初期値に指定されたobjectが存在しない場合
- 初期値に指定されたobjectが有効な型ではない、または有効な範囲ではない場合

メッセージ種別

E: エラー

パラメーターの意味

%1: 不正な値が指定されたプロパティ名
%2: 指定された値

利用者の処置

プロパティには、データ型が一致する初期値を指定してください。また、プロパティのデータ型がobject型の場合には、存在するobjectを指定するか、object内のデータを確認してください。

RCF11002

The property %1 is read only.

メッセージの意味

指定されたプロパティは読取専用のため、値を変更できません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 変更対象のプロパティ名

利用者の処置

読取専用のプロパティに対して、モデルバインディングやAPIなどを使って、値を変更しようとしていないかを確認してください。
プロパティの変更可否については、「UI部品リファレンス」の各部品のプロパティの説明を参照してください。

RCF11003

The property %1 is undefined.

メッセージの意味

UI部品の動作に必須のプロパティが定義されていません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 未定義のプロパティ名

利用者の処置

必須のプロパティが指定されているかを確認してください。
指定が必須のプロパティについては、「UI部品リファレンス」の各部品のプロパティの説明を参照してください。

RCF11004

The component described in the binding expression is not found. expression=%1

メッセージの意味

バインディング式で指定されたUI部品が存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 記述されたバインディング式

利用者の処置

バインディング式に誤りがないか、バインディング式で指定したUI部品が存在するかを確認してください。

RCF11005

The component specified in the target(s) property is not found. id=%1

メッセージの意味

targetまたはtargetsプロパティで指定されたIDを持つUI部品が存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 指定されたID

利用者の処置

指定したIDに誤りがないか、指定したUI部品が存在するかを確認してください。

RCF11006

The %1 parameter is not specified.

メッセージの意味

プロパティ名またはパスが指定されていません。

メッセージ種別

E: エラー

パラメーターの意味

%1: name または path

利用者の処置

プロパティをJavaScript APIで変更する際に、プロパティ名やパス名が正しく指定されているかを確認してください。

RCF11007

Wrong data type of the %1 property. expected type=%2. actual type=%3

メッセージの意味

値を変更しようとしたプロパティのデータ型と、指定された値のデータ型とが異なります。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名
%2: 期待されたデータ型
%3: 実際に指定された値のデータ型

利用者の処置

プロパティをJavaScript APIで変更している場合は、指定した値のデータ型がプロパティのデータ型と一致しているかを確認してください。
プロパティをモデルバインディングしている場合は、バインディング対象のデータ型がプロパティのデータ型と一致しているかを確認してください。
プロパティのデータ型については、「UI部品リファレンス」の各部品のプロパティの説明を参照してください。

RCF11008

The %1 property is not found.

メッセージの意味

指定されたプロパティが存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

プロパティをJavaScript APIで参照・更新する際に、指定したプロパティ名に誤りがないかを確認してください。

RCF11009

The component %1 cannot be declared as span element. id=%2

メッセージの意味

UI部品をタグで記述することはできません。

メッセージ種別

E: エラー

パラメーターの意味

%1: UI部品名
%2: UI部品のID

利用者の処置

タグを使用してUI部品を記述してください。
UI部品の記述形式については、「UI部品リファレンス」の各部品の記述形式を参照してください。

RCF11010

A part of property %1 is not updatable.

メッセージの意味

指定されたプロパティは部分更新できません。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

部分更新不可能なプロパティに対して部分更新していないかを確認してください。
プロパティをモデルバインディングしている場合、モデル側を部分更新してもこのエラーとなります。
プロパティの部分更新可/不可については、「UI部品リファレンス」の各部品のプロパティの説明を参照してください。

RCF11011

The rcf:id attribute value is empty string.

メッセージの意味

rcf:id属性の値が空文字列です。

メッセージ種別

E: エラー

利用者の処置

rcf:id属性に空文字列を指定することはできません。
以下のどちらかの方法で修正してください。

- IDを省略する場合
  rcf:id属性を削除する
- IDを指定する場合
  rcf:id属性に空文字列以外の文字列を指定する

RCF11012

Loading StyleSheet is not completed.

メッセージの意味

スタイルシートがロードできません。
以下のどちらかの原因が考えられます。

- <link>タグの記述に誤りがあるため、CSSファイルの読込みが完了しない
- Internet Explorer 9を使用している場合、かつ、ページ内の<link>タグおよび<style>タグの総数が32個以上ある場合、32個目以降の<link>タグや<style>タグは読み込まれないため、CSSファイルの読込みが完了しない

メッセージ種別

E: エラー

利用者の処置

原因に応じて、以下のように対処してください。

- <link>タグの記述に誤りがないかを確認してください。
- Internet Explorer 9を使用している場合は、ページ内の<link>タグおよび<style>タグの総数が32個未満になるように、ユーザーアプリケーションで記述する<link>タグおよび<style>タグの数を減らしてください。
  Ajaxフレームワークの画面部品では、内部で<link>タグを生成します。例えば、1画面に全画面部品を利用した場合、27個の<link>タグが生成されるため、ユーザーアプリケーションで記述できる<link>タグおよび<style>タグの最大個数は4になります。

RCF11013

Invalid rcf:id attribute value. value=%1

メッセージの意味

rcf:id属性の値が不正です。

メッセージ種別

E: エラー

パラメーターの意味

%1: rcf:id属性に指定した値

利用者の処置

rcf:id属性の値に不正な文字が指定されています。
指定したrcf:idの値が、以下の条件に一致しているかを確認してください。

- 最初の文字が半角英字になっていること
- 2番目以降の文字が半角英字、半角数字、アンダーバー、ハイフン、コロン、ピリオドのどれかであること

RCF11014

Specified component is not focusable.

メッセージの意味

指定された部品は、フォーカスを設定することができません。

メッセージ種別

E: エラー

利用者の処置

RCF.setFocusに、フォーカスを設定することができない部品のIDが指定されています。
指定したIDの部品を確認してください。

RCF11015

Specified component is not focusable because it is disabled.

メッセージの意味

指定された部品は、無効となっているため、フォーカスを設定することができません。

メッセージ種別

E: エラー

利用者の処置

指定した部品が無効となっている場合、フォーカスは設定できません。
RCF.setFocusに指定したIDの部品のenabledプロパティが、trueになっていることを確認してください。

RCF11016

Specified component is not focusable because it is hidden.

メッセージの意味

指定された部品は、表示されていないため、フォーカスを設定することができません。

メッセージ種別

E: エラー

利用者の処置

指定した部品が表示されていない場合、フォーカスは設定できません。
RCF.setFocusに指定したIDの部品が表示されていることを確認してください。

RCF12000

Specified object of "labelProvider" property does not have getLabel() function.

メッセージの意味

Text部品のlabelProviderプロパティに指定されたオブジェクトに、getLabel関数が定義されていません。

メッセージ種別

E: エラー

利用者の処置

labelProviderプロパティに指定したオブジェクトに、getLabel関数が実装されているかを確認してください。

RCF12100

Property "imeMode" cannot be changed while the limiter is enabled.

メッセージの意味

Limiter機能が付加されたTextInput部品では、imeModeプロパティは変更できません。

メッセージ種別

E: エラー

利用者の処置

Limiter機能を追加すると、IMEを使った入力はできません。IMEは常に無効です。
imeModeプロパティを変更しないように、アプリケーションを修正してください。

RCF12101

Specified object of "labelProvider" property does not have getLabel() function.

メッセージの意味

TextInput部品のlabelProviderプロパティに指定されたオブジェクトに、getLabel関数が定義されていません。

メッセージ種別

E: エラー

利用者の処置

labelProviderプロパティに指定したオブジェクトに、getLabel関数が実装されているかを確認してください。

RCF12500

The specified value of "%1" property is invalid.

メッセージの意味

TextArea部品のプロパティに指定した値が有効範囲外です。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

プロパティに指定した値が有効範囲に収まっているかを確認してください。

RCF12600

The data type of "%1" property is invalid.

メッセージの意味

Select部品において、値を変更しようとしたプロパティのデータ型と、指定された値のデータ型とが異なります。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

プロパティをJavaScript APIで変更している場合は、指定した値のデータ型がプロパティのデータ型と一致しているかを確認してください。
プロパティをモデルバインディングしている場合は、バインディング対象のデータ型がプロパティのデータ型と一致しているかを確認してください。
プロパティのデータ型については、「UI部品リファレンス」のSelect部品のプロパティの説明を参照してください。

RCF12602

The value property not defined at options[%1].

メッセージの意味

Select部品において、optionsプロパティに指定されたobjectにvalueが定義されていません。
optionsプロパティにobjectの配列を指定する場合には、各objectにvalueプロパティが定義されている必要があります。

メッセージ種別

E: エラー

パラメーターの意味

%1: valueが定義されていない項目のインデックス

利用者の処置

optionsに指定した配列内の各オブジェクトに、valueプロパティが存在しているかを確認してください。

RCF12603

Duplicate option value "%1" detected at options[%2].

メッセージの意味

Select部品において、optionsプロパティに指定されたvalueが重複しています。
optionsプロパティのvalueは一意でなくてはなりません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 重複しているvalue
%2: 重複を検出した項目のインデックス

利用者の処置

optionsにstringの配列を指定した場合は、配列中に重複した文字列がないかを確認してください。
optionsにobjectの配列を指定した場合は、配列中の各objectのvalueプロパティが重複していないかを確認してください。

RCF12604

Specified index of "%1" property(%2) is out of range.

メッセージの意味

Select部品において、selectedIndexプロパティまたはselectedIndexesプロパティに指定されたインデックスの値が有効な範囲から外れています。
selectedIndexプロパティとselectedIndexesプロパティの要素の値は、それぞれ、以下の範囲に収まっていなければなりません。

- selectedIndex
  -1 ≦ 指定値≦ (optionsプロパティの配列の長さ) -1
- selectedIndexes
  0 ≦ 指定値 ≦ (optionsプロパティの配列の長さ) -1

メッセージ種別

E: エラー

パラメーターの意味

%1: 異常なインデックスを検出したプロパティ名
%2: 異常と判断されたインデックス

利用者の処置

各プロパティに指定した値が有効範囲に収まっているかを確認してください。

RCF12605

Specified value of "%1" property(%2) is not found in values of "options" property.

メッセージの意味

Select部品において、selectedValueプロパティまたはselectedValuesプロパティに指定された値に誤りがあります。
selectedValueプロパティとselectedValuesプロパティの各要素の値は、optionsプロパティのどれかのvalueと一致していなければなりません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 異常な値を検出したプロパティ名
%2: 異常と判断された値

利用者の処置

各プロパティに指定した値が、optionsプロパティのvalueに含まれているかを確認してください。

RCF12606

Duplicate value in "%1" property(%2).

メッセージの意味

Select部品において、selectedIndexesプロパティまたはselectedValuesプロパティに重複した値が指定されました。
selectedIndexesプロパティとselectedValuesプロパティの各要素の値は、ほかの要素の値と重複してはなりません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 重複を検出したプロパティ名
%2: 重複している値

利用者の処置

エラーが発生したプロパティに、重複した値を指定していないかを確認してください。

RCF12800

Wrong data type in 'list' property. (index %1)

メッセージの意味

ComboBox部品において、listプロパティに指定した配列の要素のデータ型に誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: 異常を検出したインデックス

利用者の処置

listプロパティの配列の各要素にはstringを指定してください。

RCF12801

The object of "buttonImage" does not have "base" property.

メッセージの意味

ComboBox部品において、buttonImageプロパティに指定したオブジェクトに、baseが設定されていません。

メッセージ種別

E: エラー

利用者の処置

buttonImageプロパティには、baseに画像のURLを設定したオブジェクトを指定してください。

RCF12900

The input string does not follow the date format.

メッセージの意味

DateInput部品において、入力値が日付の書式に従っていません。
本メッセージは、デフォルトのconverterを使用してdateparseerrorイベントが発生した際に、DateParseEventのerrorプロパティで通知されます。

メッセージ種別

I: 情報

RCF12901

The input day is wrong.

メッセージの意味

DateInput部品において、入力された日の値に誤りがあります。
本メッセージは、デフォルトのconverterを使用してdateparseerrorイベントが発生した際に、DateParseEventのerrorプロパティで通知されます。

メッセージ種別

I: 情報

RCF12902

The input month is wrong.

メッセージの意味

DateInput部品において、入力された月の値に誤りがあります。
本メッセージは、デフォルトのconverterを使用してdateparseerrorイベントが発生した際に、DateParseEventのerrorプロパティで通知されます。

メッセージ種別

I: 情報

RCF12903

The input year is wrong.

メッセージの意味

DateInput部品において、入力された年の値に誤りがあります。
本メッセージは、デフォルトのconverterを使用してdateparseerrorイベントが発生した際に、DateParseEventのerrorプロパティで通知されます。

メッセージ種別

I: 情報

RCF12904

Conversion to Date object failed. (%1)

メッセージの意味

DateInput部品において、入力値のDateオブジェクトへの変換に失敗しました。
RCF12900～RCF12903以外の理由で変換に失敗した場合、本メッセージが出力されます。
本メッセージは、デフォルトのconverterを使用してdateparseerrorイベントが発生した際に、DateParseEventのerrorプロパティで通知されます。

メッセージ種別

I: 情報

パラメーターの意味

%1: 変換が失敗した原因となるエラーメッセージ

RCF13000

The input string does not follow the number format.

メッセージの意味

NumberInput部品において、入力値が数値の書式に従っていません。
本メッセージは、デフォルトのconverterを使用してnumberparseerrorイベントが発生した際に、NumberParseEventのerrorプロパティで通知されます。

メッセージ種別

I: 情報

RCF13001

Conversion to Number object failed. (%1)

メッセージの意味

NumberInput部品において、入力値のNumberオブジェクトへの変換に失敗しました。
RCF13000以外の理由で変換に失敗した場合、本メッセージが出力されます。
本メッセージは、デフォルトのconverterを使用してnumberparseerrorイベントが発生した際に、NumberParseEventのerrorプロパティで通知されます。

メッセージ種別

I: 情報

パラメーターの意味

%1: 変換が失敗した原因となるエラーメッセージ

RCF13010

Specified object of "converter" property does not have format() function.

メッセージの意味

NumberInput部品において、converterプロパティに指定されたobjectにformat関数が定義されていません。

メッセージ種別

E: エラー

利用者の処置

converterプロパティに指定したオブジェクトに、format関数が実装されているかを確認してください。

RCF13011

Specified object of "converter" property does not have parse() function.

メッセージの意味

NumberInput部品において、converterプロパティに指定されたobjectにparse関数が定義されていません。

メッセージ種別

E: エラー

利用者の処置

converterプロパティに指定したオブジェクトに、parse関数が実装されているかを確認してください。

RCF13100

The input string does not follow the format.

メッセージの意味

MaskedTextInput部品において、入力文字列が書式に従っていません。
本メッセージは、invalidvalueerrorイベントが発生した際に、InvalidValueErrorEventのerrorプロパティで通知されます。

メッセージ種別

I: 情報

RCF13200

The specified value of "date" property is not a valid date.

メッセージの意味

MaskedDateInput部品において、dateプロパティに指定された値が、Dateオブジェクトではないか、有効な範囲内ではありません。

メッセージ種別

E: エラー

利用者の処置

dateプロパティに指定した値が、有効な範囲内の値を持つDateオブジェクトであるかを確認してください。

RCF13201

The specified value of "%1" property is out of valid range.

メッセージの意味

MaskedDateInput部品において、プロパティに指定された値が有効な範囲内ではありません。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

エラーが発生したプロパティに指定した値が、有効な範囲内であるかを確認してください。

RCF13202

Invalid date format.

メッセージの意味

MaskedDateInput部品において、formatプロパティに指定された値が、正しいフォーマット指定文字列ではありません。

メッセージ種別

E: エラー

利用者の処置

formatプロパティに指定した文字列が、フォーマット指定の文字列として正しい形式であるかを確認してください。

RCF13300

The data type of "%1" property is invalid.

メッセージの意味

SelectList部品において、値を変更しようとしたプロパティのデータ型と、指定された値のデータ型とが異なります。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

プロパティをJavaScript APIで変更している場合は、指定した値のデータ型がプロパティのデータ型と一致しているかを確認してください。
プロパティをモデルバインディングしている場合は、バインディング対象のデータ型がプロパティのデータ型と一致しているかを確認してください。
プロパティのデータ型については、「UI部品リファレンス」のSelectList部品のプロパティの説明を参照してください。

RCF13301

Specified object of "labelProvider" property does not have getLabel() function.

メッセージの意味

SelectList部品のlabelProviderプロパティに指定されたオブジェクトに、getLabel関数が定義されていません。

メッセージ種別

E: エラー

利用者の処置

labelProviderプロパティに指定したオブジェクトに、getLabel関数が実装されているかを確認してください。

RCF13302

Specified index of "%1" property(%2) is out of range.

メッセージの意味

SelectList部品において、selectedIndexプロパティまたはselectedIndexesプロパティに指定されたインデックスの値が有効な範囲から外れています。
selectedIndexプロパティとselectedIndexesプロパティの要素の値は、それぞれ、以下の範囲に収まっていなければなりません。

- selectedIndex
  -1 ≦ 指定値 ≦ (optionsプロパティの配列の長さ) -1
- selectedIndexes
  0 ≦ 指定値 ≦ (optionsプロパティの配列の長さ) -1

メッセージ種別

E: エラー

パラメーターの意味

%1: 異常なインデックスを検出したプロパティ名
%2: 異常と判断されたインデックス

利用者の処置

各プロパティに指定した値が有効範囲に収まっているかを確認してください。

RCF13303

Specified object of "renderer" property does not have render() function.

メッセージの意味

SelectList部品において、rendererプロパティに指定されたオブジェクトにrender関数が定義されていません。

メッセージ種別

E: エラー

利用者の処置

rendererプロパティに指定したオブジェクトに、render関数が実装されているかを確認してください。

RCF13304

Duplicate value in "selectedIndexes" property(%1).

メッセージの意味

SelectList部品において、selectedIndexesプロパティに重複した値が指定されました。
selectedIndexesプロパティの各要素の値は、ほかの要素の値と重複してはなりません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 重複しているインデックス

利用者の処置

selectedIndexesプロパティに重複した値を指定していないかを確認してください。

RCF13400

The data type of "%1" property is invalid.

メッセージの意味

CheckList部品において、値を変更しようとしたプロパティのデータ型と、指定された値のデータ型とが異なります。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

プロパティをJavaScript APIで変更している場合は、指定した値のデータ型がプロパティのデータ型と一致しているかを確認してください。
プロパティをモデルバインディングしている場合は、バインディング対象のデータ型がプロパティのデータ型と一致しているかを確認してください。
プロパティのデータ型については、「UI部品リファレンス」のCheckList部品のプロパティの説明を参照してください。

RCF13401

Specified object of "labelProvider" property does not have getLabel() function.

メッセージの意味

CheckList部品のlabelProviderプロパティに指定されたオブジェクトに、getLabel関数が定義されていません。

メッセージ種別

E: エラー

利用者の処置

labelProviderプロパティに指定したオブジェクトに、getLabel関数が実装されているかを確認してください。

RCF13402

Specified index of "%1" property(%2) is out of range.

メッセージの意味

CheckList部品において、selectedIndexプロパティまたはselectedIndexesプロパティに指定されたインデックスの値が有効な範囲から外れています。
selectedIndexプロパティとselectedIndexesプロパティの要素の値は、それぞれ、以下の範囲に収まっていなければなりません。

- selectedIndex
  -1 ≦ 指定値 ≦ (optionsプロパティの配列の長さ) -1
- selectedIndexes
  0 ≦ 指定値 ≦ (optionsプロパティの配列の長さ) -1

メッセージ種別

E: エラー

パラメーターの意味

%1: 異常なインデックスを検出したプロパティ名
%2: 異常と判断されたインデックス

利用者の処置

各プロパティに指定した値が有効範囲に収まっているかを確認してください。

RCF13404

Duplicate value in "selectedIndexes" property(%1).

メッセージの意味

CheckList部品において、selectedIndexesプロパティに重複した値が指定されました。
selectedIndexesプロパティの各要素の値は、ほかの要素の値と重複してはなりません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 重複しているインデックス

利用者の処置

selectedIndexesプロパティに重複した値を指定していないかを確認してください。

RCF13700

No containers are specified as the child elements.

メッセージの意味

ViewStackまたはTabPanel部品において、子要素のコンテナ(ViewContainer、FragmentContainer)が指定されていません。

メッセージ種別

E: エラー

利用者の処置

子要素に、コンテナ(ViewContainerまたはFragmentContainer)が1個以上指定されているかを確認してください。

RCF13800

Invalid tabPosition "%1" is specified.

メッセージの意味

TabPanel部品において、tabPositionプロパティに指定された値に誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: 指定された値

利用者の処置

tabPositionプロパティに指定した値に誤りがないかを確認してください。

RCF13801

Invalid key specification for "%1".

メッセージの意味

TabPanel部品において、nextKeyプロパティまたはpreviousKeyプロパティに指定されたキー指定文字列に誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

指定したキー指定文字列に誤りがないかを確認してください。

RCF13802

The same key is assigned to "%1" and "%2".

メッセージの意味

TabPanel部品において、nextKeyプロパティとpreviousKeyプロパティに同じキーが指定されました。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名
%2: プロパティ名

利用者の処置

nextKeyプロパティとpreviousKeyプロパティには、異なるキーを指定してください。

RCF13900

FragmentContainer activation error: error state = %1.

メッセージの意味

FragmentContainer部品において、フラグメントHTMLの有効化処理に失敗しました。
本メッセージは、fragmenterrorが発生した際に、FragmentErrorEventのmessageプロパティで通知されます。

メッセージ種別

I: 情報

パラメーターの意味

%1: エラーが発生した時点のFragmentContainer部品の状態

利用者の処置

有効化処理の失敗時に何らかの処理を行いたい場合は、onFragmentErrorのイベントリスナに個別の処理を実装してください。

RCF13901

The timeout property is invalid.

メッセージの意味

FragmentContainer部品において、timeoutプロパティに指定された値に誤りがあります。
timeoutプロパティには、1以上のnumberを指定してください。単位はミリ秒です。

メッセージ種別

E: エラー

利用者の処置

timeoutプロパティに1以上のnumberが指定されているかを確認してください。

RCF13902

FragmentContainer:%1 cannot unload because the rcf:id attribute value of child element is not specified.

メッセージの意味

FragmentContainer部品において、フラグメントHTMLに記述されている部品のrcf:id属性が指定されていないため、FragmentContainerをアンロードできません。

メッセージ種別

E: エラー

パラメーターの意味

%1: FragmentContainerの部品ID

利用者の処置

当該FragmentContainerのフラグメントHTMLに記述されている部品に対して、rcf:idが指定されているかを確認してください。
また、当該FragmentContainerの下階層にFragmentContainerがある場合にも、下階層のFragmentContainerのフラグメントHTMLに記述されている部品に対してrcf:idが指定されているかを確認してください。

RCF13903

FragmentContainer:%1 fragment-HTML is not specified.

メッセージの意味

FragmentContainerのreplaceSrcメソッドにおいて、引数のフラグメントHTMLが指定されていません。

メッセージ種別

E: エラー

パラメーターの意味

%1: FragmentContainerの部品ID

利用者の処置

引数にフラグメントHTMLを指定してください。

RCF14000

The zIndex of new modal Window must be %1 and above.

メッセージの意味

すでに表示しているモーダルウィンドウより小さなzIndexのモーダルウィンドウを表示しようとしました。

メッセージ種別

E: エラー

パラメーターの意味

%1: 新しく表示するWindowに必要なzIndexの値

利用者の処置

zIndexの設定に誤りがないかを確認してください。
新しく表示するモーダルウィンドウは、すでに表示しているモーダルウィンドウよりzIndexを2以上大きな値に指定する必要があります。

RCF14001

Closed modal Window violated the order.

メッセージの意味

表示順を無視して、モーダルウィンドウが閉じられました。

メッセージ種別

E: エラー

利用者の処置

モーダルウィンドウは、表示した順に閉じられなければなりません。順序どおりに閉じられているかを確認してください。

RCF14100

Specified defaultColumnWidth is less than 10.

メッセージの意味

TableView部品において、defaultColumnWidthプロパティに10より小さい値が指定されています。

メッセージ種別

E: エラー

利用者の処置

defaultColumnWidthプロパティに指定した値を確認してください。

RCF14101

Only a binding expression can be specified for rcf:selectedRows attribute value.

メッセージの意味

TableView部品およびTableEdit部品において、rcf:selectedRows属性の値には、バインディング式以外は記述できません。

メッセージ種別

E: エラー

利用者の処置

rcf:selectedRows属性にバインディング式が記述されているかを確認してください。

RCF14102

Specified selectedRows is not empty array.

メッセージの意味

TableView部品またはTableEdit部品において、selectedRowsプロパティに空の配列([])以外の値が指定されています。

メッセージ種別

E: エラー

利用者の処置

selectedRowsプロパティをバインディングする場合、バインディングされる値が空の配列であるかを確認してください。

RCF14201

%1 cannot edit %2 value.

メッセージの意味

TableEdit部品において、編集を行う入力部品で編集できないデータ型の値を編集しようとしています。

メッセージ種別

E: エラー

パラメーターの意味

%1: 入力部品名
%2: データ型

利用者の処置

編集を行う入力部品ごとに、編集可能なデータ型は異なります。
ViewColumnの子要素に入力部品を指定していない場合、デフォルトはTextInputです。TextInputでは文字列以外は編集できませんので、列データに適切な入力部品を指定してください。ViewColumnの子要素に指定できる入力部品については、「UI部品リファレンス」の「2.3.4 ViewColumn」を参照してください。
また、ViewColumnの子要素に指定した入力部品を指定している場合は、列データのデータ型が適切かを確認してください。

RCF14202

Only a binding expression can be specified for rcf:selectedCell attribute value.

メッセージの意味

TableEdit部品において、rcf:selectedCell属性の値には、バインディング式以外は記述できません。

メッセージ種別

E: エラー

利用者の処置

rcf:selectedCell属性にバインディング式が記述されているかを確認してください。

RCF14203

Specified selectedCell is not null.

メッセージの意味

TableEdit部品において、selectedCellプロパティにnull以外の値が指定されています。

メッセージ種別

E: エラー

利用者の処置

selectedCellプロパティをバインディングする場合、バインディングされる値がnullであるかを確認してください。

RCF14300

The render function is not found in the renderer object.

メッセージの意味

ViewColumn部品において、rendererプロパティに指定されたオブジェクトに、render関数が定義されていません。

メッセージ種別

E: エラー

利用者の処置

rendererプロパティに指定したオブジェクトに、render関数が実装されているかを確認してください。

RCF14301

The compare function is not found in the comparator object.

メッセージの意味

ViewColumn部品において、comparatorプロパティに指定されたオブジェクトに、compare関数が定義されていません。

メッセージ種別

E: エラー

利用者の処置

comparatorプロパティに指定したオブジェクトに、compare関数が実装されているかを確認してください。

RCF14302

Duplicate ViewColumn's name. name=%1

メッセージの意味

TableView部品またはTableEdit部品の子要素として記述された複数のViewColumn部品において、nameが重複しています。

メッセージ種別

E: エラー

パラメーターの意味

%1: ViewColumnの名前

利用者の処置

TableView部品またはTableEdit部品の子要素として記述されたViewColumn部品で、nameが重複しないように修正してください。

RCF14303

Specified columnWidth is less than 10.

メッセージの意味

ViewColumn部品において、columnWidthプロパティに10より小さい値が指定されています。

メッセージ種別

E: エラー

利用者の処置

columnWidthプロパティに指定した値を確認してください。

RCF14500

The property selectedDates can contain one date at most if selectMode is SINGLE.

メッセージの意味

selectModeプロパティがSINGLEであるCalendar部品において、selectedDatesプロパティに複数の日付が指定されました。
selectModeプロパティがSINGLEの場合、selectedDatesプロパティには日付を1つしか指定できません。

メッセージ種別

E: エラー

利用者の処置

selectModeプロパティおよびselectedDatesプロパティの指定値を確認してください。

RCF14501

Invalid data in property "%1" or date object not specified.

メッセージの意味

Calendar部品において、specialDatesまたはdateToolTipsプロパティに指定された配列中のオブジェクトに誤りがあるか、そのオブジェクトのdateプロパティがDate型のオブジェクトではありません。
これらのプロパティに指定した配列中のオブジェクトには、Date型のオブジェクトを持つdateプロパティを定義する必要があります。

メッセージ種別

E: エラー

パラメーターの意味

%1: 誤りのあるプロパティ名

利用者の処置

エラーが発生したプロパティに指定している配列中の各オブジェクトがdateプロパティを持ち、かつDate型のオブジェクトが設定されていることを確認してください。

RCF14502

The property selectedDates cannot be changed, when "selectable" is false.

メッセージの意味

selectableプロパティがfalseであるCalendar部品において、selectedDatesプロパティを変更しようとしました。
selectableプロパティがfalseの場合、selectedDatesは変更できません。

メッセージ種別

E: エラー

利用者の処置

Calendar部品を日付選択可能な状態で動作させたいかどうかを確認し、選択可能としたい場合はselectableプロパティをtrueに指定してください。
選択不可としたい場合は、selectedDatesプロパティを変更しようとしている箇所がないかを確認してください。

RCF14503

Illegal value is specified for property "%1".

メッセージの意味

Calendar部品において、指定されたプロパティの値に誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

エラーが発生したプロパティに指定された値に誤りがないかを確認してください。

RCF14700

Specified target is not PopupCalendar.

メッセージの意味

CalendarButton部品において、targetプロパティに指定されたUI部品がPopupCalendar部品ではありません。

メッセージ種別

E: エラー

利用者の処置

targetプロパティに指定したIDを持つUI部品が、PopupCalendar部品かを確認してください。

RCF14801

The property of "data" property is invalid or the "name" property of "data" property is not found. (path=%1 property=%2)

メッセージの意味

TreeView部品において、dataプロパティのプロパティに誤りがあるか、nameプロパティが存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: ノードパス
%2: プロパティ名

利用者の処置

dataプロパティのプロパティを修正するか、nameプロパティを指定してください。

RCF14802

The value of "name" property is duplicated. (path=%1)

メッセージの意味

TreeView部品において、同じ階層でnameプロパティが重複しています。
nameプロパティは、同じ階層で一意でなくてはなりません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 重複しているノードパス

利用者の処置

同じ階層のnameプロパティに、重複した値を指定しないように修正してください。

RCF14804

The content of "data" property is invalid.

メッセージの意味

TreeView部品において、dataプロパティのノードデータが正しく設定されていません。

メッセージ種別

E: エラー

利用者の処置

dataプロパティに、ルートノードとその子ノードが正しく設定されているかを確認してください。

RCF14805

The nodePath is unselectable.

メッセージの意味

TreeView部品において、ノードパスが選択できません。

メッセージ種別

E: エラー

利用者の処置

ノードパスが選択可能になっているかを確認してください。

RCF14806

The %1 parameter is invalid.

メッセージの意味

TreeView部品において、パラメーターに誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: ノードパス

利用者の処置

ノードパスに誤りがないかを確認してください。

RCF14807

The nodePath is not found. (rcf:nodePath=%1)

メッセージの意味

TreeView部品において、ノードパスが存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: ノードパス

利用者の処置

ノードパスが存在するかを確認してください。

RCF14900

Specified defaultColumnWidth is less than 10(px).

メッセージの意味

DataGrid部品において、defaultColumnWidthプロパティに10(ピクセル)より小さい値が指定されています。

メッセージ種別

E: エラー

利用者の処置

defaultColumnWidthプロパティに指定した値を確認してください。

RCF14901

Only a binding expression can be specified for rcf:selectedRows attribute value.

メッセージの意味

DataGrid部品において、rcf:selectedRows属性の値には、バインディング式以外は記述できません。

メッセージ種別

E: エラー

利用者の処置

rcf:selectedRows属性にバインディング式が記述されているかを確認してください。

RCF14902

Specified selectedRows is not empty array.

メッセージの意味

DataGrid部品において、selectedRowsプロパティに空の配列([])以外の値が指定されています。

メッセージ種別

E: エラー

利用者の処置

selectedRowsプロパティをバインディングする場合、バインディングされる値が空の配列であるかを確認してください。

RCF14903

The specified value of "%1" property is invalid.

メッセージの意味

DataGrid部品のプロパティに指定した値が有効範囲外です。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

プロパティに指定した値が有効範囲に収まっているかを確認してください。

RCF14904

%1 cannot edit %2 value.

メッセージの意味

DataGrid部品において、編集を行う入力部品で編集できないデータ型の値を編集しようとしています。

メッセージ種別

E: エラー

パラメーターの意味

%1: 入力部品名
%2: データ型

利用者の処置

編集を行う入力部品ごとに、編集可能なデータ型は異なります。
ViewColumnGridの子要素に入力部品を指定していない場合、デフォルトはTextInputです。TextInputでは文字列以外は編集できませんので、列データに適切な入力部品を指定してください。ViewColumnGridの子要素に指定できる入力部品については、「UI

部品リファレンス」の「2.3.5 ViewColumnGrid」を参照してください。
また、ViewColumnGridの子要素に指定した入力部品を指定している場合は、列データのデータ型が適切かを確認してください。

RCF14906

Only a binding expression can be specified for rcf:selectedCell attribute value.

メッセージの意味

DataGrid部品において、rcf:selectedCell属性の値には、バインディング式以外は記述できません。

メッセージ種別

E: エラー

利用者の処置

rcf:selectedCell属性にバインディング式が記述されているかを確認してください。

RCF14907

Specified selectedCell is not null.

メッセージの意味

DataGrid部品において、selectedCellプロパティにnull以外の値が指定されています。

メッセージ種別

E: エラー

利用者の処置

selectedCellプロパティをバインディングする場合、バインディングされる値がnullであるかを確認してください。

RCF15000

The render function is not found in the renderer object.

メッセージの意味

ViewColumnGrid部品において、rendererプロパティまたはselectedRendererプロパティに指定されたオブジェクトに、render関数が定義されていません。
または、DataGrid部品において、selectedRendererプロパティに指定されたオブジェクトに、render関数が定義されていません。

メッセージ種別

E: エラー

利用者の処置

rendererプロパティまたはselectedRendererプロパティに指定したオブジェクトに、render関数が実装されているかを確認してください。

RCF15001

The compare function is not found in the comparator object.

メッセージの意味

ViewColumnGrid部品において、comparatorプロパティに指定されたオブジェクトに、compare関数が定義されていません。

メッセージ種別

E: エラー

利用者の処置

comparatorプロパティに指定したオブジェクトに、compare関数が実装されているかを確認してください。

RCF15002

Duplicate ViewColumnGrid's name. name=%1.

メッセージの意味

DataGrid部品の子要素として記述された複数のViewColumnGrid部品において、nameが重複しています。

メッセージ種別

E: エラー

パラメーターの意味

%1: ViewColumnGridの名前

利用者の処置

DataGrid部品の子要素として記述されたViewColumnGrid部品で、nameが重複しないように修正してください。

RCF15003

Specified columnWidth is less than 10(px).

メッセージの意味

ViewColumnGrid部品において、columnWidthプロパティに10(ピクセル)より小さい値が指定されています。

メッセージ種別

E: エラー

利用者の処置

columnWidthプロパティに指定した値を確認してください。

RCF15004

The fixable property cannot be declared the last ViewColumnGrid.

メッセージの意味

固定列は、ViewColumnGrid部品の最後には定義できません。

メッセージ種別

E: エラー

利用者の処置

ViewColumnGrid部品において、最後尾列が固定列でないかを確認してください。
DataGrid部品では、固定されていない列が1つ以上必要です。

RCF15005

Specified fixed columnWidth property is out of table's width.

メッセージの意味

DataGrid部品内において、指定された固定列のcolumnWidthプロパティがテーブルの表示範囲を超えています。

メッセージ種別

E: エラー

利用者の処置

固定列のcolumnWidthプロパティが、テーブルの表示範囲を超えていないかを確認してください。

RCF15006

Illegal value is specified for property. (id=%1 property=%2)

メッセージの意味

DataGrid部品において、指定されたプロパティの値に誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: 部品ID
%2: プロパティ名

利用者の処置

エラーが発生したプロパティに指定された値に誤りがないかを確認してください。

RCF15007

Invalid ViewColumnGrid's definition. name1=%1 name2=%2

メッセージの意味

指定されたViewColumnGridの定義内容に誤りがあります。
以下の原因が考えられます。

- 列ヘッダのcolumnSpanの定義が行データの境界をまたぐ定義であり、かつ、列ヘッダの境界と行データの境界が一致しない
- 行データのcolumnSpanの定義が列ヘッダの境界をまたぐ定義であり、かつ、行データの境界と列ヘッダの境界が一致しない
- 固定列と固定されていない列をまたいでcolumnSpanが設定されている

メッセージ種別

E: エラー

パラメーターの意味

%1: ViewColumnGrid(列ヘッダ)の名前
%2: ViewColumnGrid(行データ)の名前

利用者の処置

列ヘッダまたは行データのcolumnSpanの定義を見直してください。

RCF15100

Duplicate ViewColumnCheck component in DataGrid.

メッセージの意味

DataGrid部品内において、複数のViewColumnCheck部品が指定されています。

メッセージ種別

E: エラー

利用者の処置

DataGrid部品において、ViewColumnCheck部品は1つまでとしてください。

RCF15200

Duplicate ViewColumnTree component in DataGrid.

メッセージの意味

DataGrid部品内において、複数のViewColumnTree部品が指定されています。

メッセージ種別

E: エラー

利用者の処置

DataGrid部品において、ViewColumnTree部品は1つまでとしてください。

RCF16100

The object property is not specified.

メッセージの意味

Model部品において、objectプロパティが指定されていません。

メッセージ種別

E: エラー

利用者の処置

objectプロパティが指定されているかを確認してください。

RCF16101

The schema property is not specified.

メッセージの意味

Model部品において、schemaプロパティが指定されていません。

メッセージ種別

E: エラー

利用者の処置

schemaプロパティが指定されているかを確認してください。

RCF16102

Invalid path parameter.

メッセージの意味

Model部品において、validateProperty関数で指定されたpath引数に誤りがあります。

メッセージ種別

E: エラー

利用者の処置

validateProperty関数のpath引数に、null、undefined、または文字列以外のデータが指定されています。
path引数で指定した値に誤りがないかを確認してください。

RCF16103

Unable to find %1 in schema.

メッセージの意味

Model部品において、validateProperty関数のpath引数で指定されたプロパティがスキーマに存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

validateProperty関数のpath引数に指定した値、およびModelのschemaプロパティで指定したスキーマの内容に誤りがないかを確認してください。

RCF16104

Unable to find %1 in object.

メッセージの意味

Model部品において、validateProperty関数のpath引数で指定されたプロパティがモデルオブジェクトに存在しません。
または、Model部品において、validate関数実行時にスキーマ定義で指定された対象となるプロパティがモデルオブジェクトに存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

validateProperty関数実行時に発生した場合、path引数に指定した値、およびModelのobjectプロパティで指定したオブジェクトの内容に誤りがないかを確認してください。
validate関数実行時に発生した場合、スキーマ定義の内容に誤りがないか、およびモデルオブジェクトにそのプロパティが存在しているかを確認してください。

RCF16105

Array object cannot be specified to object property.

メッセージの意味

Model部品において、objectプロパティにはarray型のオブジェクトを指定できません。

メッセージ種別

E: エラー

利用者の処置

objectプロパティに指定したオブジェクトが配列でないことを確認してください。

RCF16190

Invalid schema at %1.

メッセージの意味

Model部品において、スキーマの定義に誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: スキーマの位置

利用者の処置

エラーメッセージで示された位置のスキーマの内容に誤りがないかを確認してください。

RCF16199

Validation failed. location=%1

メッセージの意味

Model部品において、バリデーションに失敗しました。
本メッセージは、検証エラーを示すメッセージとして、検証エラー情報のmessageで通知されます。

検証エラー情報は、以下から取得できます。

- Model部品のvalidateProperty関数およびvalidate関数の戻り値
- ValidationHelper部品のonValidationErrorイベントリスナで受け取るイベントのresultsプロパティ

メッセージ種別

I: 情報

パラメーターの意味

%1: スキーマの位置

RCF16200

The %1 parameter is invalid.

メッセージの意味

TreeModel部品において、パラメーターに誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: ノードパス

利用者の処置

ノードパスに誤りがないかを確認してください。

RCF16201

The nodePath is not found. (rcf:nodePath=%1)

メッセージの意味

TreeModel部品において、ノードパスが存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1:ノードパス

利用者の処置

ノードパスが存在するかを確認してください。

RCF16300

The %1 parameter is invalid.

メッセージの意味

MenuModel部品において、パラメーターに誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: ノードパス

利用者の処置

ノードパスに誤りがないかを確認してください。

RCF16301

The nodePath is not found. (rcf:nodePath=%1)

メッセージの意味

MenuModel部品において、ノードパスが存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: ノードパス

利用者の処置

ノードパスが存在するかを確認してください。

RCF17000

The component specified to "target" property is not TextInput. rcf:id="%1"

メッセージの意味

AutoCompleter部品において、targetプロパティに指定されたUI部品がTextInput部品ではありません。

メッセージ種別

E: エラー

パラメーターの意味

%1: targetプロパティに指定されたID

利用者の処置

targetプロパティに指定したIDを持つUI部品が、TextInput部品かを確認してください。

RCF17001

The component specified to "list" property is not SelectList. rcf:id="%1"

メッセージの意味

AutoCompleter部品において、listプロパティに指定されたUI部品がSelectList部品ではありません。

メッセージ種別

E: エラー

パラメーターの意味

%1: listプロパティに指定されたID

利用者の処置

listプロパティに指定したIDを持つUI部品が、SelectList部品かを確認してください。

RCF17002

The component specified to "list" property is not found. rcf:list="%1"

メッセージの意味

AutoCompleter部品において、listプロパティに指定されたUI部品が存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: listプロパティの値

利用者の処置

listプロパティに指定されたIDを持つSelectList部品が存在するかを確認してください。

RCF17003

Specified object of "completer" property does not have complete() function.

メッセージの意味

AutoCompleter部品において、completerプロパティに指定されたオブジェクトにcomplete関数が定義されていません。

メッセージ種別

E: エラー

利用者の処置

completerプロパティに指定したオブジェクトに、complete関数が実装されているかを確認してください。

RCF17100

Specified target does not exist. rcf:id="%1"

メッセージの意味

Limiter部品において、targetプロパティに指定されたUI部品が存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: targetプロパティに指定されたID

利用者の処置

targetプロパティに指定されたIDを持つUI部品が存在するかを確認してください。

RCF17101

Specified target is not controllable with Limiter. rcf:id="%1"

メッセージの意味

Limiter部品において、targetプロパティに指定されたUI部品は、Limiter部品の機能付加対象ではありません。

Limiter部品の機能付加対象は、以下のとおりです。

- TextInput
- ComboBox
- DateInput
- NumberInput

メッセージ種別

E: エラー

パラメーターの意味

%1: targetプロパティに指定されたID

利用者の処置

targetプロパティに指定されたUI部品が、Limiter部品の機能付加対象かを確認してください。

RCF17102

The target is controlled with another Limiter. rcf:id="%1"

メッセージの意味

Limiter部品において、targetプロパティに指定されたUI部品には、すでにほかのLimiter部品によって入力文字を制限されています。
1つのUI部品に対して、設定可能なLimiterは1つだけです。

メッセージ種別

E: エラー

パラメーターの意味

%1: targetプロパティに指定されたID

利用者の処置

targetプロパティに指定されたUI部品が、ほかのLimiter部品によって入力制限されていないかを確認してください。

RCF17200

The property "%1" has an invalid value "%2".

メッセージの意味

NumeralOnlyLimiter部品において、プロパティに指定された値に誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名
%2: 指定された値

利用者の処置

エラーの発生したプロパティの指定値に誤りがないかを確認してください。

RCF17201

The same character is assigned to properties "%1" and "%2".

メッセージの意味

NumeralOnlyLimiter部品において、複数のプロパティに対して同じ文字が割り当てられています。

メッセージ種別

E: エラー

パラメーターの意味

%1、%2: 同じ文字が割り当てられたプロパティ名

利用者の処置

各プロパティには、個別の文字を割り当てるようにしてください。

RCF17300

No types are specified.

メッセージの意味

EnableCharTypeLimiter部品において、typesプロパティが指定されていません。

メッセージ種別

E: エラー

利用者の処置

typesプロパティの指定に誤りがないかを確認してください。

RCF17301

Invalid type "%1" in types property.

メッセージの意味

EnableCharTypeLimiter部品において、typesプロパティに指定された値に誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: エラーを検出した値

利用者の処置

typesプロパティの指定に誤りがないかを確認してください。

RCF17500

Specified target does not exist. rcf:id="%1"

メッセージの意味

FocusManager部品において、targetsプロパティに指定されたUI部品が存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 見つからなかったUI部品のID

利用者の処置

targetsプロパティに指定されたIDを持つUI部品が存在するかを確認してください。

RCF17501

Specified target is not controllable with FocusManager. rcf:id="%1"

メッセージの意味

FocusManager部品において、targetsプロパティに指定されたUI部品は、フォーカス制御の対象ではありません。

フォーカス制御可能なUI部品は、以下のとおりです。

- フォーム部品(Textは除く)
- TabPanel
- TableView
- TableEdit
- DataGrid
- Calendar
- CalendarButton
- TreeView
- ScrapingView(コンテンツ内は除く)
- CheckBoxGroup
- RadioButtonGroup

メッセージ種別

E: エラー

パラメーターの意味

%1: フォーカス制御不可能なUI部品のID

利用者の処置

targetsプロパティに指定されたIDを持つUI部品がフォーカス制御対象かを確認してください。

RCF17502

Duplicated targets. rcf:id="%1"

メッセージの意味

FocusManager部品において、targetsプロパティに指定されたIDが重複しています。

メッセージ種別

E: エラー

パラメーターの意味

%1: 重複しているID

利用者の処置

targetsプロパティに指定された値に重複がないかを確認してください。

RCF17503

Invalid key specification for "%1".

メッセージの意味

FocusManager部品において、nextKeyプロパティまたはpreviousKeyプロパティに指定されたキー指定文字列に誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名

利用者の処置

指定したキー指定文字列に誤りがないかを確認してください。

RCF17504

The same key is assigned to "nextKey" and "previousKey".

メッセージの意味

FocusManager部品において、nextKeyプロパティとpreviousKeyプロパティに同じキーが指定されました。

メッセージ種別

E: エラー

利用者の処置

nextKeyプロパティとpreviousKeyプロパティには、異なるキーを指定してください。

RCF17600

Specified target is not CheckBox. rcf:id="%1".

メッセージの意味

CheckBoxGroup部品において、targetsプロパティに指定されたUI部品がCheckBox部品ではありません。

メッセージ種別

E: エラー

パラメーターの意味

%1: エラーを検出したUI部品のID

利用者の処置

targetsプロパティに指定したIDを持つUI部品がCheckBox部品かを確認してください。

RCF17601

Each CheckBox must have a non-empty "value" property. rcf:id="%1".

メッセージの意味

CheckBoxGroup部品において、グループ化するCheckBox部品にvalueプロパティが定義されていません。
グループ化するCheckBox部品には、空文字列以外の値が指定されている必要があります。

メッセージ種別

E: エラー

パラメーターの意味

%1: エラーを検出したCheckBox部品のID

利用者の処置

グループ化するCheckBox部品に、valueプロパティが指定されているかを確認してください。

RCF17602

Each CheckBox must have a unique "value" property. nonunique rcf:value="%1".

メッセージの意味

CheckBoxGroup部品において、グループ化するCheckBox部品のvalueプロパティが重複しています。
グループ化するCheckBox部品のvalueプロパティは、グループ内で一意でなければなりません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 重複しているvalueプロパティの値

利用者の処置

グループ化するCheckBox部品に、一意のvalueプロパティが指定されているかを確認してください。

RCF17603

Duplicate value in "selectedValues" property(%1).

メッセージの意味

CheckBoxGroup部品において、selectedValuesプロパティに重複した値が指定されました。
selectedValuesプロパティの各要素の値は、ほかの要素の値と重複してはなりません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 重複している値

利用者の処置

selectedValuesプロパティに重複した値を指定していないかを確認してください。

RCF17604

Specified value of "selectedValues[%1]" property(%2) is not found in values of targets.

メッセージの意味

CheckBoxGroup部品において、selectedValuesプロパティに指定された値に該当するCheckBox部品がグループ内に存在しませんでした。
selectedValuesプロパティの各要素の値は、グループ内のCheckBox部品のvalueプロパティのどれかに一致している必要があります。

メッセージ種別

E: エラー

パラメーターの意味

%1: エラーを検出したインデックス
%2: エラーを検出した値

利用者の処置

selectedValuesプロパティに指定した値が、グループ内のCheckBox部品のvalueプロパティと一致しているかを確認してください。

RCF17700

Specified target is not RadioButton. rcf:id="%1"

メッセージの意味

RadioButtonGroup部品において、targetsプロパティに指定されたUI部品がRadioButton部品ではありません。

メッセージ種別

E: エラー

パラメーターの意味

%1: エラーを検出したUI部品のID

利用者の処置

targetsプロパティに指定したIDを持つUI部品がRadioButton部品かを確認してください。

RCF17701

Each RadioButton must have a non-empty "value" property. rcf:id="%1"

メッセージの意味

RadioButtonGroup部品において、グループ化するRadioButton部品にvalueプロパティが定義されていません。
グループ化するRadioButton部品には、空文字列以外の値が指定されている必要があります。

メッセージ種別

E: エラー

パラメーターの意味

%1: エラーを検出したRadioButton部品のID

利用者の処置

グループ化するRadioButton部品に、valueプロパティが指定されているかを確認してください。

RCF17702

Each RadioButton must have a unique "value" property. nonunique rcf:value="%1"

メッセージの意味

RadioButtonGroup部品において、グループ化するRadioButton部品のvalueプロパティが重複しています。
グループ化するRadioButton部品のvalueプロパティは、グループ内で一意でなければなりません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 重複しているvalueプロパティの値

利用者の処置

グループ化するRadioButton部品に、一意のvalueプロパティが指定されているかを確認してください。

RCF17704

Specified value of "selectedValue" property(%1) is not found in values of targets.

メッセージの意味

RadioButtonGroup部品において、selectedValueプロパティに指定された値に該当するRadioButton部品がグループ内に存在しませんでした。
selectedValueプロパティの値は、グループ内のRadioButton部品のvalueプロパティのどれかに一致している必要があります。

メッセージ種別

E: エラー

パラメーターの意味

%1: エラーを検出した値

利用者の処置

selectedValueプロパティに指定した値が、グループ内のRadioButton部品のvalueプロパティと一致しているかを確認してください。

RCF18801

The property of "data" property is invalid or the "name" property of "data" property is not found. (path=%1 property=%2)

メッセージの意味

ContextMenu部品において、dataプロパティのプロパティに誤りがあるか、nameプロパティが存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: ノードパス
%2: プロパティ名

利用者の処置

dataプロパティのプロパティを修正するか、nameプロパティを指定してください。

RCF18802

The specified value of "name" property is duplicated. (path=%1)

メッセージの意味

ContextMenu部品において、同じ階層でnameプロパティが重複しています。
nameプロパティは、同じ階層で一意でなくてはなりません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 重複しているノードパス

利用者の処置

同じ階層のnameプロパティに、重複した値を指定しないように修正してください。

RCF18804

The content of "data" property is invalid.

メッセージの意味

ContextMenu部品において、dataプロパティのノードデータが正しく設定されていません。

メッセージ種別

E: エラー

利用者の処置

dataプロパティに、ルートノードとその子ノードが正しく設定されているかを確認してください。

RCF18806

The %1 parameter is invalid.

メッセージの意味

ContextMenu部品において、パラメーターに誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: ノードパス

利用者の処置

ノードパスに誤りがないかを確認してください。

RCF18807

The nodePath is not found. (rcf:nodePath=%1)

メッセージの意味

ContextMenu部品において、ノードパスが存在しません。

メッセージ種別

E: エラー

パラメーターの意味

%1: ノードパス

利用者の処置

ノードパスが存在するかを確認してください。

RCF18850

The specified argument is invalid. (%1)

メッセージの意味

不正な引数が指定されました。

メッセージ種別

E: エラー

パラメーターの意味

%1: 引数名

利用者の処置

引数の値を確認してください。

J.2.2　通信フレームワーク(JavaScript)に関するメッセージ

ここでは、通信フレームワークのクライアントJavaScript、およびマッシュアップフレームワークのクライアントJavaScriptに関するエラーメッセージについて説明します。

RCF20000

Illegal eventLogLevel is specified. eventLogLevel="%1"

メッセージの意味

Ajaxフレームワークの動作定義を行うRCF_configオブジェクトに、不正なeventLogLevelが指定されました。

メッセージ種別

E: エラー

パラメーターの意味

%1: エラーを検出したeventLogLevelの指定値

利用者の処置

eventLogLevelには、「ERROR」、「WARN」、「INFO」、「TRACE」のどれかの値を指定してください。

RCF20001

Illegal request object. type="%1"

メッセージの意味

通信時のデータ内に、functionまたはundefinedのオブジェクトが定義されています。

メッセージ種別

E: エラー

パラメーターの意味

%1: 不正なデータ

利用者の処置

UjiRequest、RcfRequest、RcfObjectConverter.encodeで、functionまたはundefinedのプロパティをオブジェクトから取り除いてください。

RCF20002

Timeout occurred.

メッセージの意味

サーバとの通信時にタイムアウトが発生しました。

メッセージ種別

E: エラー

利用者の処置

タイムアウトに対する処理を記述する場合は、UjiRequestまたはRcfRequestの通信設定オブジェクト(option)にerrorHandlerプロパティを定義してください。

RCF20003

Illegal option object.

メッセージの意味

通信設定オブジェクト(option)に、不正なオブジェクトが定義されています。

メッセージ種別

E: エラー

利用者の処置

通信設定オブジェクト(option)に、正しいオブジェクトを定義してください。

RCF20004

Illegal URL. url="%1"

メッセージの意味

通信設定オブジェクト(option)に、不正なURLが定義されています。

メッセージ種別

E: エラー

パラメーターの意味

%1: 不正なURL

利用者の処置

通信設定オブジェクト(option)のurlプロパティに、正しい値を定義してください。

RCF20005

Too long queryString is specified. URL length="%1"

メッセージの意味

URLの文字列長が2083バイトを超えました。

メッセージ種別

E: エラー

パラメーターの意味

%1: URLの文字列長

利用者の処置

以下の点に注意して、URLの文字列長が2083バイト以下になるように、クエリ文字列を設定してください。

- URL文字列は、以下の形式の文字列長です。
  http://hostname[:port]/pathname[jsessionid][?query]
- queryには、キーと値を区切る「=」が含まれます。
- 複数のクエリ文字列を設定した場合、キーとキーを区切る「&」が文字列数に含まれます。
- URLエンコードの対象となる文字列が設定された場合、URLエンコード処理後の文字列数となります。例えば、文字コード「UTF-8」の実行環境において、文字列「あ」が設定された場合は、9バイト(%E3%81%82)として処理されます。

RCF20006

Illegal queryString object is specified. key="%1", value="%2"

メッセージの意味

クエリ文字列の値にString以外のオブジェクトが指定されました。

メッセージ種別

E: エラー

パラメーターの意味

%1: クエリ文字列に設定されたキー
%2: キー(%1)に設定された値の型

利用者の処置

クエリ文字列から不正なデータを取り除いてください。

RCF20007

Illegal argument is specified. key="%1", value="%2"

メッセージの意味

RCF.addQueryStringのパラメーターに不適切な値が指定されました。

メッセージ種別

E: エラー

パラメーターの意味

%1: keyパラメーターに指定された型(nullが指定された場合はnull)
%2: valueパラメーターに指定された型

利用者の処置

RCF.addQueryStringに指定したパラメーターについて、以下を確認してください。

- パラメーターには、Stringオブジェクトを指定します。
- keyパラメーターにnullを指定することはできません。

RCF20008

Illegal argument is specified. url="%1"

メッセージの意味

RCF.encodeURLのパラメーターに不適切な値が指定されました。

メッセージ種別

E: エラー

パラメーターの意味

%1: urlパラメーターに指定された型(nullが指定された場合はnull)

利用者の処置

RCF.encodeURLに指定したパラメーターについて、以下を確認してください。

- パラメーターには、Stringオブジェクトを指定します。
- urlパラメーターにnullを指定することはできません。

RCF20009

Illegal argument is specified. queryString="%1"

メッセージの意味

RCF.setQueryStringのパラメーターに不適切な値が指定されました。

メッセージ種別

E: エラー

パラメーターの意味

%1: queryStringパラメーターに指定された型(nullが指定された場合はnull)

利用者の処置

RCF.setQueryStringに指定したパラメーターについて、以下を確認してください。

- パラメーターには、連想配列オブジェクトを指定します。
- queryStringパラメーターにnullを指定することはできません。

RCF20010

Illegal requestParams object.

メッセージの意味

クライアント通信APIで、リクエストパラメーターオブジェクト(requestParams)に、不正なオブジェクトが定義されています。

メッセージ種別

E: エラー

利用者の処置

リクエストパラメーターオブジェクト(requestParams)に、正しい値を定義してください。

RCF20011

The error occurred in the application. function="%1" cause="%2"

メッセージの意味

アプリケーションのコールバック関数で例外が発生しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: コールバック関数の名前(callback、errorHandler、preInvoke、postInvokeのどれか)
%2: 例外のメッセージ

利用者の処置

例外の原因を特定し、必要に応じてコールバック関数を修正してください。

J.3 サーバのエラーメッセージ

ここでは、通信フレームワークのサーバ機能、およびマッシュアップフレームワークのサーバ機能が出力するエラーメッセージについて説明します。

利用者の処置に記載しているApcoordinatorのマニュアルについては、Interstage Application Serverのマニュアルを参照してください。

J.3.1 初期化処理に関するメッセージ

ここでは、通信フレームワークの初期化処理に関するエラーメッセージについて説明します。

RCF0001

[%1] is not supported.

メッセージの意味

要求された機能はサポートされていません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 指定機能名 (stub、invoke)

利用者の処置

利用している製品のバージョンおよびサポートしている機能を確認してください。クライアントに設定した機能と製品のバージョンを合わせてください。

RCF0002

[%1] is not supported.

メッセージの意味

規定外の機能要求です。

メッセージ種別

E: エラー

パラメーターの意味

%1: 指定機能名

利用者の処置

利用している製品のバージョン、サポートしている機能、およびクライアントに設定したURLプロパティの設定を確認してください。クライアントに設定した機能と製品のバージョンを合わせてください。

RCF0003

Apcoordinator is unavailable.

メッセージの意味

Apcoordinator連携を実行するための環境が整っていません。

メッセージ種別

E: エラー

利用者の処置

uji.jarをクラスパスに設定してください。

RCF0004

Invalid URI is specified. (URI = %1)

メッセージの意味

指定したURIに誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: URI

利用者の処置

利用している製品のバージョンおよびサポートしている機能を確認してください。クライアントのURLプロパティに利用可能なURLを設定してください。

RCF0005

Illegal initialization parameter is specified. (name= %1, value=%2)

メッセージの意味

初期化パラメーターの指定に誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: 初期化パラメーター名
%2: 初期化パラメーターの設定値

利用者の処置

エラーメッセージに表示されている初期化パラメーターを正しく設定してください。

J.3.2　環境定義ファイルに関するメッセージ

ここでは、環境定義ファイルに関するエラーメッセージについて説明します。

RCF0101

Failed to read file. (%1)

メッセージの意味

分割された定義ファイルが格納されているフォルダ内に、定義ファイルがありません。または、不正なファイルまたはサブフォルダが存在します。

メッセージ種別

E: エラー

パラメーターの意味

%1: 読込み対象のファイル名

利用者の処置

エラーメッセージに表示されているファイルを確認してください。分割機能を利用した場合、フォルダには1つ以上の環境定義ファイルだけを格納することができます。

RCF0102

"%1" tag is duplicated. (%2, %3)

メッセージの意味

環境定義ファイルの設定情報が重複しています。

メッセージ種別

E: エラー

パラメーターの意味

%1: 重複しているタグ名
%2: ファイル名:行番号
%3: ファイル名:行番号

利用者の処置

エラーメッセージに表示されている環境定義ファイルのタグを確認してください。重複しているタグのどちらか一方を削除してください。

RCF0103

Failed to read %1.

メッセージの意味

環境定義ファイルの読込みでエラーが発生しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: 読込みエラーとなったファイル名

利用者の処置

読込みエラーとなったファイルが、環境定義ファイルとして正しい構成となっていることを確認してください。このエラーメッセージの前後に出力されているエラーメッセージがあれば、そのメッセージに従って対処してください。

RCF0104

"%1" tag is not found.

メッセージの意味

環境定義ファイルに必要なタグが記述されていません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 記述が必要なタグ名

利用者の処置

設定が不足しているタグを定義してください。

RCF0105

Illegal element name is specified: %1.

メッセージの意味

環境定義ファイルに不要なタグが記述されています。

メッセージ種別

E: エラー

パラメーターの意味

%1: 設定不要なタグ名

利用者の処置

不要なタグを削除してください。

RCF0106

Illegal content is specified in %1 element: %2.

メッセージの意味

環境定義ファイルの定義に誤りがあります。タグの値に不正な文字列が設定されています。

メッセージ種別

E: エラー

パラメーターの意味

%1: タグ名
%2: 設定データ

利用者の処置

エラーメッセージに表示されているタグの値を正しく設定してください。

RCF0107

Illegal version number. (%1:%2,%3:%4)

メッセージの意味

環境定義ファイルの分割機能を利用した環境定義ファイルバージョン情報に相違があります。

メッセージ種別

E: エラー

パラメーターの意味

%1: ファイル名1:行番号
%2: 設定バージョン1
%3: ファイル名2:行番号
%4: 設定バージョン2

利用者の処置

すべての環境定義ファイルに同一のバージョンを設定してください。

RCF0108

No xml reader.

メッセージの意味

org.xml.sax.XMLReaderのインスタンス生成に失敗しました。

メッセージ種別

E: エラー

利用者の処置

SAXパーサが利用可能となるように、運用環境のクラスパスを確認してください。

RCF0109

Illegal DataBean class is specified in %1 element: %2.

メッセージの意味

不正なデータBeanクラスが指定されています。

メッセージ種別

E: エラー

パラメーターの意味

%1: タグ名
%2: データBeanのクラス名

利用者の処置

データBeanのクラスを確認してください。
データBeanのJavaクラスは、以下の条件を満たしている必要があります。

- WebアプリケーションのCLASSPATHに含まれるクラスであること
- パブリックのデフォルトコンストラクタを持つこと
- Ajaxフレームワークが提供するクラス以外であること
- com.fujitsu.uji.DataBeanクラスを継承していること
- JavaBeans形式であること

J.3.3 Apcoordinator連携機能に関するメッセージ

ここでは、Apcoordinator連携機能に関するエラーメッセージについて説明します。

RCF0201

Data bean is not found. (%1)

メッセージの意味

beanIdで指定されたデータBeanは定義ファイルに設定されていません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 指定されたbeanId

利用者の処置

環境定義ファイルにデータBeanの定義を追加してください。

RCF0202

Invalid scope "%1" is specified.

メッセージの意味

定義ファイルで指定したデータBeanのscopeに誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: 環境定義ファイルで設定されたscope

利用者の処置

環境定義ファイルのscopeを変更してください。

RCF0203

Apcoordinator で設定したエラーメッセージ

メッセージの意味

ビジネスクラスの呼出しに失敗しました。

メッセージ種別

E: エラー

利用者の処置

運用ログに出力されるスタックトレースの情報を確認してください。「UJI」で始まるエラーコードについては、「Apcoordinator メッセージ集」を参照してください。

RCF0204

Unknown result :%1.

メッセージの意味

Apcoordinator連携で使用するデータBeanは、ResponseBeanクラスのインスタンスである必要があります。

メッセージ種別

E: エラー

パラメーターの意味

%1: データBeanとして利用したクラス

利用者の処置

データBeanクラスにResponseBeanインターフェースを実装してください。

RCF0205

受け取った例外のメッセージ

メッセージの意味

Apcoordinator連携のレスポンス書込み中に例外が発生しました。

メッセージ種別

E: エラー

利用者の処置

運用ログに出力されるスタックトレースの情報を確認してください。

RCF0206

受け取った例外のメッセージ

メッセージの意味

Apcoordinatorクラスへのアクセスで例外が発生しました。

メッセージ種別

E: エラー

利用者の処置

運用ログに出力されるスタックトレースの情報を確認してください。「UJI」で始まるエラーコードについては、「Apcoordinator メッセージ集」を参照してください。

RCF0207

Failed to instantiate bean: %1.

メッセージの意味

データBeanを生成しようとしましたが、IOExceptionが発生しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: データBeanのクラス名

利用者の処置

データBeanが壊れている可能性があります。データBeanがロード可能であることを確認してください。

RCF0208

Failed to instantiate bean: %1.

メッセージの意味

データBeanを生成しようとしましたが、クラスが見つかりません。

メッセージ種別

E: エラー

パラメーターの意味

%1: データBeanのクラス名

利用者の処置

環境定義ファイルで設定したデータBeanクラスがクラスパスに設定されていることを確認してください。

RCF0209

Data bean is not found. (%1)

メッセージの意味

beanIdにデータBeanは対応付けられていません。

メッセージ種別

E: エラー

パラメーターの意味

%1: 指定されたbeanId

利用者の処置

環境定義ファイルにbeanIdに対するデータBeanが定義されているか、ビジネスメソッドでbeanIdに対するsetResponseBeanメソッドが実行されているかを確認してください。

J.3.4　イベントログ機能に関するメッセージ

ここでは、イベントログ機能に関するエラーメッセージについて説明します。

RCF0401

Log is not output. (logLevel = %1)

メッセージの意味

指定されたログレベルでは、イベントログに出力されません。

メッセージ種別

W: 警告

パラメーターの意味

%1: ログレベル

利用者の処置

サーバに設定した初期化パラメーター(logLevel)と、クライアントに設定したログレベル(eventLogLevel)の値を確認してください。
サーバに設定するログレベルは、クライアントの設定値と同じか低いレベルを設定してください。

以下の表に、クライアントのログレベル(eventLogLevel)とサーバ側の設定値(logLevel)との対応関係を示します。

クライアント設定文字列	サーバ設定数値
TRACE	20
INFO	10
WARN	7
ERROR	3

RCF0402

Invalid log level is specified. (logLevel = %1)

メッセージの意味

規定外のログレベルが指定されました。

メッセージ種別

W: 警告

パラメーターの意味

%1: ログレベル

利用者の処置

正しいログレベルを送信してください。

RCF0403

受け取った例外のメッセージ

メッセージの意味

クライアントへの応答メッセージを書込み中に入出力例外が発生しました。

メッセージ種別

E: エラー

利用者の処置

運用ログに出力されるスタックトレースの情報を確認してください。

RCF0404

受け取った例外のメッセージ

メッセージの意味

クライアントからのリクエストパラメーターの取得に失敗しました。

メッセージ種別

E: エラー

利用者の処置

運用ログに出力されるスタックトレースの情報を確認してください。「UJI」で始まるエラーコードについては、「Apcoordinator メッセージ集」を参照してください。

RCF0405

Cannot set log level: %1.

メッセージの意味

初期化パラメーターlogLevelの指定に誤りがあります。

メッセージ種別

W: 警告

パラメーターの意味

%1: ログレベル

利用者の処置

初期化パラメーター(logLevel)を正しく設定してください。

J.3.5 エントリサーブレットに関するメッセージ

ここでは、エントリサーブレットに関するエラーメッセージについて説明します。

RCF0500

受け取った例外のメッセージ

メッセージの意味

エントリサーブレットの実行中にInvokeException以外の例外が発生しました。

メッセージ種別

E: エラー

利用者の処置

運用ログに出力されるスタックトレースの情報を確認してください。「UJI」で始まるエラーコードについては、「Apcoordinator メッセージ集」を参照してください。

J.3.6 データ型変換機能に関するメッセージ

ここでは、データ型変換機能に関するエラーメッセージについて説明します。

RCF0600

Converter is not found for java type: %1.

メッセージの意味

指定された型に対応するコンバータが定義されていません。

メッセージ種別

E: エラー

パラメーターの意味

%1: Java型

利用者の処置

コンバータを環境定義ファイルに設定してください。

RCF0601

Structural depth of the object tree exceeds a regulation value.
: depth = %1

メッセージの意味

オブジェクトツリーの構造の深さが規定値を超えています。

メッセージ種別

E: エラー

パラメーターの意味

%1: ツリーのパス

利用者の処置

JavaScriptのオブジェクトの構造を変更して、規定値以内の深さになるように修正してください。

RCF0610

[up] Cannot refer javascript type [%1] as java [%2]. path = %3

メッセージの意味

上りデータの型とデータBeanの型に相違があります。

メッセージ種別

E: エラー

パラメーターの意味

%1: クライアントで指定した型
%2: データBeanで定義した型
%3: ツリーのパス:データBeanのクラス

利用者の処置

クライアントで定義したデータ型とデータBeanのプロパティを合わせてください。

RCF0611

[up] Conversion error is occurred. (number = %1 : type = %2)

メッセージの意味

上りデータ(number型)の変換でエラーが発生しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: クライアントで設定した数値
%2: コンバータのクラス名

利用者の処置

クライアントから送信する数値が変換できるようにコンバータを設定するか、コンバータが変換可能な数値をクライアントから送信してください。

RCF0612

[up] Cannot convert real number '%1' to integral number. Type = %2

メッセージの意味

上りデータ(number型)を整数型に変換できませんでした。

メッセージ種別

E: エラー

パラメーターの意味

%1: クライアントで設定したデータ
%2: コンバータのクラス名

利用者の処置

クライアントから送信するデータを、数値に変換できる値にしてください。

RCF0613

[up] The length of javascript string for java Character (or char) must be 1.

メッセージの意味

CharacterConverterで利用する文字列は、長さが1である必要があります。

メッセージ種別

E: エラー

利用者の処置

クライアントで設定するデータの文字列長を1にしてください。

RCF0614

Bean property is not found. name = "%1" : bean = %2

メッセージの意味

上りデータに対応するプロパティがデータBeanに定義されていません。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名
%2: データBeanクラス名

利用者の処置

上りデータに対応するプロパティをデータBeanに追加してください。

RCF0615

[up] Number format error is occurred. name = "%1" value = "%2"

メッセージの意味

上りデータの変換(BigDecimal、BigInteger)でフォーマットエラーが発生しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名
%2: 変換対象文字列

利用者の処置

クライアントで設定するデータをBigDecimal、BigIntegerで変換可能な値に変更してください。

RCF0616

The setter method for "%1" is not found in %2.

メッセージの意味

データBeanのプロパティに対応するsetterメソッドが定義されていません。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名
%2: データBeanのクラス名

利用者の処置

データBeanにsetterメソッドを追加してください。

RCF0617

An exception occurred in the setter method: class = %1, method = %2, type of the argument = %3

メッセージの意味

データBeanのsetterメソッドで例外が発生しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: データBeanのクラス名
%2: メソッド名
%3: パラメーターの型

利用者の処置

運用ログに出力される例外情報を確認し、例外情報に出力されたエラーの原因を取り除いてください。

RCF0618

The getter method for "%1" is not found in %2.

メッセージの意味

データBeanのプロパティに対応するgetterメソッドが定義されていません。

メッセージ種別

E: エラー

パラメーターの意味

%1: プロパティ名
%2: データBeanのクラス名

利用者の処置

データBeanにgetterメソッドを追加してください。

RCF0619

An exception occurred in the getter method: class = %1, method = %2

メッセージの意味

データBeanのgetterメソッドで例外が発生しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: データBeanのクラス名
%2: メソッド名

利用者の処置

運用ログに出力される例外情報を確認し、例外情報に出力されたエラーの原因を取り除いてください。

RCF0621

An exception occurred in the Array.set method: class = %1

メッセージの意味

データBeanのArray.setメソッドで例外が発生しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: データBeanのクラス名

利用者の処置

送信データ(上りデータ)には、サーバ側のデータBeanに保持されている配列数よりも小さい配列数を持つデータを指定してください。

RCF0650

[introspect]Exception is occurred while creating the bean. : Bean class = %1

メッセージの意味

データBeanのインスタンス生成中に例外が発生しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: データBeanのクラス名

利用者の処置

利用するデータBeanのクラスをクラスパスに追加してください。

RCF0680

***受け取った例外のメッセージ*(%1)**

メッセージの意味

環境定義ファイルで定義されたクラスのロードに失敗しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: 環境定義ファイル名:行番号

利用者の処置

環境定義ファイルのクラス名を正しいクラス名に変更してください。

RCF0690

Invalid number format. %1

メッセージの意味

上りデータの変換で数値に変換できませんでした。

メッセージ種別

E: エラー

パラメーターの意味

%1: 数値変換の対象となる文字列

利用者の処置

クライアントで設定するデータを数値変換できる値にしてください。

RCF0699

IOException is occurred while writing the response data. code = %1

メッセージの意味

下りデータの書込み中にIOExceptionが発生しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: エラーコード

利用者の処置

運用ログに出力されるスタックトレースを確認してください。

J.3.7　通信に関するメッセージ

ここでは、通信に関するエラーメッセージについて説明します。

RCF0700

An illegal response was received : %1.

メッセージの意味

異常なレスポンスデータを受信しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: URL

利用者の処置

サーバがタイムアウトしたか、または起動していません。サーバがタイムアウトしている場合は、サーバ側の処理を見直してください。
サーブレット連携機能の簡易通信方式を使用している場合は、ビジネスロジックの処理でcom.fujitsu.interstage.rcf.http.RcfServletHelperクラスのsetRequestBeanメソッドまたはsetExceptionメソッドのどちらかを呼び出す必要があります。呼び出していない場合は、どちらかを呼び出すように処理を修正してください。
クライアント通信APIを使用している場合は、マッシュアッププロキシ側の設定を見直してください。

J.3.8　マッシュアップに関するメッセージ

ここでは、マッシュアップに関するエラーメッセージについて説明します。

RCF0801

Mashup Proxy failed in the start. %1

メッセージの意味

マッシュアッププロキシの起動に失敗しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: 起動失敗の原因となる詳細メッセージ

利用者の処置

詳細メッセージから原因を特定し、修正してください。
以下の表に、詳細メッセージを説明します。

詳細メッセージ	説明
Authentication class is not found.	マッシュアップ定義ファイルのsecurity要素の値がデフォルトから変更されていないかを確認してください。
An illegal class was specified for authentication class.	
It failed in making the directory for the log.	ログディレクトリの作成に失敗しました。 マッシュアップ定義ファイルで指定したログ出力ディレクトリに書込み権限があるか、または同じ名前のファイルが存在していないかを確認してください。
The file already exists.	
It failed in the reading of muf.xml.	スタックトレースの情報から原因を特定し、修正してください。
Failed in the initialization of the network.	

詳細メッセージ	説明
An illegal data was specified for muf_admin tag.	また、当該メッセージの直前に以下のメッセージが出力されている場合は、マッシュアップ定義ファイルの該当の行を確認してください。 Mashup Proxy: ERROR: muf.xml line:"行数" column:"カラム番号" "xmlライブラリからのメッセージ"

RCF0802

It failed in the output of the log.

メッセージの意味

ログの出力に失敗しました。

メッセージ種別

E: エラー

利用者の処置

スタックトレースの情報から原因を特定し、修正してください。

RCF0803

The specified service is not found.

メッセージの意味

サービスが見つかりません。

メッセージ種別

E: エラー

利用者の処置

サービスを登録するか、クライアント通信APIで指定しているサービスIDの値を修正してください。

RCF0804

The request excluding MuRequest is not accepted.

メッセージの意味

クライアント通信API以外からのアクセスを受け付けました。

メッセージ種別

E: エラー

利用者の処置

マッシュアッププロキシの呼び出し方が不正です。
クライアントアプリケーションを見直してください。

RCF0805

Service returned the error response. %1

メッセージの意味

呼び出したサービスからエラーが返却されました。

メッセージ種別

E: エラー

パラメーターの意味

%1: サービスから返却されたエラーの詳細メッセージ

利用者の処置

詳細メッセージから原因を特定し、修正してください。
以下の表に、詳細メッセージを説明します。

詳細メッセージ	説明
It failed in the getting of the content.	コンテンツの取得に失敗しました。サービス提供元を確認してください。 以下の原因が考えられます。 ・ サーバが見つからない ・ 返却されたコンテンツが空
It failed in the connection.	
There is no response from the service.	
It failed in the conversion of the content.	コンテンツの内容の解析に失敗しました。提供されるコンテンツの内容を確認してください。 以下の原因が考えられます。 ・ RESTアダプタから正しくないXMLが返却された ・ HTMLの内容が複雑なため、またはHTMLとして認識できないため、XHTMLへの正規化で失敗した
The content is too complex or not html.	
The error occurred by conversion from html to xml.	
Illegal URL is set.	サービスのURLの値が正しくありません。
The specified XSL file is not found.	サービスのXSLで設定されているファイルが存在しません。
A basic authentication is necessary for the access of this service.	サービスまたはプロキシサーバに認証が必要です。すでに認証情報を設定している場合は、IDとパスワードが正しいかを確認してください。
A proxy authentication is necessary for the access of this service.	
It failed in the analysis of the XSL file.	XSLファイルの内容が正しくありません。XSLファイルを修正するか、スクレイピングツールでXSLファイルを再作成してください。
Illegal type is set.	クライアント通信APIで指定しているtypeの値が正しくありません。

RCF0806

It failed in the encryption.

メッセージの意味

暗号化の処理に失敗しました。

メッセージ種別

E: エラー

利用者の処置

環境を見直してください。
暗号化ライブラリが使えない場合は、認証情報の暗号化を使用しないように設定してください。

J.3.9　サーブレット連携機能に関するメッセージ

ここでは、サーブレット連携機能に関するエラーメッセージについて説明します。

RCF0901

Response has already been committed.

メッセージの意味

レスポンスはすでにコミットされています。

メッセージ種別

E: エラー

利用者の処置

コミットが1回になるようにビジネスロジックを修正してください。

RCF0902

%1 is null.

メッセージの意味

パラメーターにnullが指定されました。

メッセージ種別

E: エラー

パラメーターの意味

%1: パラメーター名

利用者の処置

パラメーターに正しい値を指定し、nullが指定されないように修正してください。

RCF0904

User's application threw the exception. method=%1 cause=%2

メッセージの意味

アプリケーションのコールバックで例外が発生しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: 例外が発生したメソッド
%2: 例外のメッセージ

利用者の処置

運用ログに出力されるスタックトレースの情報を確認してください。例外の原因を特定し、必要に応じてコールバックを修正してください。

RCF0905

User's application returned null. method=%1

メッセージの意味

アプリケーションのコールバックがnullを返しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: nullを返却したメソッド

利用者の処置

nullが返却された原因を特定し、必要に応じてコールバックを正しく修正してください。

RCF0906

%1 is illegal. value=%2

メッセージの意味

クライアントから送信されたデータに誤りがあります。

メッセージ種別

E: エラー

パラメーターの意味

%1: データのパラメーター名
%2: データの文字列表記

利用者の処置

ビジネスロジックの実行処理(クライアントの処理)で指定したデータ(オブジェクト)を確認し、正しいデータをサーバに送信するように修正してください。

RCF0907

Business logic threw the exception. exception=%1 message=%2

メッセージの意味

アプリケーションのビジネスロジックが、クラスRcfServletHelperのメソッドsetExceptionに例外を設定しました。

メッセージ種別

E: エラー

パラメーターの意味

%1: 例外のクラス
%2: 例外のメッセージ(例外のメッセージがnullの場合は「message=」以降の文字列は出力されません。)

利用者の処置

ビジネスロジックがsetExceptionに設定した例外の情報は、エラーオブジェクトのcauseプロパティに設定されています。causeプロパティの値を確認して例外の原因を特定し、必要に応じてビジネスロジックを修正してください。

RCF0908

The Ajax framework has not been initialized.

メッセージの意味

Ajaxフレームワークが初期化されていません。

メッセージ種別

E: エラー

利用者の処置

HTML/JSPファイルにAjaxフレームワークの初期化処理が正しく記述されているかを確認してください。Ajaxフレームワークの初期化については、「2.3.3 Ajaxフレームワークの初期化処理」を参照してください。

用語集

Ajax(Asynchronous JavaScript + XML)

JavaScriptのXMLHttpRequestオブジェクトを利用して非同期にサーバと通信し、ページ全体を読み込まずにページの必要な部分だけを書き換えるWebアプリケーションの開発技術です。

Ajaxフレームワーク環境定義ファイル

Ajaxフレームワークの通信フレームワークの動作環境を設定するXML形式のファイルです。

Ajaxページエディタ

Ajaxフレームワークアプリケーションで利用する画面をGUIベースで編集するツールです。
EclipseおよびInterstage Studio ワークベンチのプラグインとして動作します。

Apcoordinator

Java(TM) 2 Platform, Enterprise Edition(J2EE) およびJava(TM) Platform, Enterprise Edition(Java EE)に従ったアプリケーションの構築を支援するアプリケーションフレームワーク製品です。

Apcoordinator連携機能

Webブラウザ上のJavaScriptアプリケーションから、Apcoordinatorを利用する機能です。

Eclipse

オープンソースとして公開されている統合開発環境(IDE)です。
Eclipseのプラグイン機能を使用することで、開発環境を拡張することが可能です。

HTTPリクエスト

WebブラウザからWebサーバに対して処理や動作を要求する送信です。HTTPプロトコルに従って送信されます。

HTTPレスポンス

Webサーバからブラウザに送られるリクエストの処理結果です。

Interstage

富士通の提供するアプリケーションサーバ製品のファミリー名です。

Interstage Application Server

富士通の提供するアプリケーションサーバ製品です。

Interstage Studio

Ajaxフレームワークアプリケーション(Ajaxフレームワークを利用したWebアプリケーション)を開発するときに利用する、コンポーネント指向のJava統合開発環境です。アプリケーションを効率的に開発するための各種デザインツール、開発支援ツール、コンポーネント部品が組み込まれています。

Interstage Studioワークベンチ

Interstage Studioが提供するワークベンチの1つで、Java EEに準拠したアプリケーションの開発に用います。

J2EE(Java 2 Enterprise Edition)

プログラミング言語「Java 2」の機能セットの1つで、企業の業務システムやElectronic Commerce(電子商取引)などで使われるサーバに必要な機能をまとめたものです。標準機能セットのJava 2 Standard Edition(J2SE)に、サーバ用のAPIや諸機能を付加したものといえます。

Java EE

Java Platform, Enterprise Editionの略で、Java Platform, Standard Edition(Java SE)の拡張機能の形態で提供されるJavaの企業の大規模システム(サーバ用途)向けの機能セットです。

JSON(JavaScript Object Notation)

JavaScriptから派生した、データ交換を行うためのデータ記述形式の1つで、テキストベースのデータフォーマットです。

JSP(JavaServer Pages)

Javaによる動的Webページです。JSPタグやJSPスクリプトレット記述により、サーバでの動作を記述します。

REST(Representational State Transfer)

HTTPプロトコルのGETやPOSTメソッドによるリクエストに対して、XMLで応答を返すようなシステムです。
Webサービスのうち、REST形式で動作するサービスをRESTサービスと呼びます。

RIA(Rich Internet Application)

Java applet、Ajaxなどを用いたユーザーインターフェースによって、単純なHTMLで記述されたページに比べて操作性や表現力に優れたWebアプリケーションのことをいいます。

SSL(Secure Sockets Layer)

Netscape Communications社が提唱した、インターネット上で情報をのぞき見されないようにするためのネットワークセキュリティ方式です。Webサーバへの接続時に、まず相手の認証や使用する暗号などを確認し、相互に認証したあとで送受信を行うという2階層になっています。

UI部品

クライアントで使用するAjaxフレームワークの部品群の総称です。UI部品には、以下の3種類があります。

- 画面部品
- 機能部品
- 機能付加部品

Webアプリケーション

HTMLファイル、イメージファイル、サーブレット、JSPファイルなどのWebリソースと、Webアプリケーション環境定義ファイルから構築します。機能を、1つのWebアプリケーションのパッケージ単位として開発できます。

Webサービス

HTTPなどのインターネット関連技術を応用して、Web上で公開されているサービスです。

WYSIWYG(What You See Is What You Get)

出力イメージ(文書や画面)を確認しながら編集できるという概念を表します。

XHTML(Extensible HyperText Markup Language)

HTMLをXMLの文法で定義し直したマークアップ言語です。

XML(Extensible Markup Language)

文書、データの意味や構造を記述するための汎用的なマークアップ言語です。

XSL(Extensible Stylesheet Language)

XMLのスタイルシートを記述する言語です。

XSLT(XML Stylesheet Language Transform)

XMLの構造を別の形式に変形するための変換ルールを記述するものです。

XSLTプロセッサ

XSLTの定義に従って、XML形式のデータをHTML形式やほかのドキュメント形式へと変換するものです。
スクレイピングツールで利用しています。

アクションイベント

画面部品が操作されたときに発生するイベントです。

アクセスログ機能

ブラウザ上のJavaScriptアプリケーションからWebサービスへのアクセスログを収集し、出力する機能です。

アダプタ

様々なWebサービスとのアクセスを制御する部品の総称です。マッシュアップフレームワークでは、RESTサービスやHTMLサービスのためのアダプタを提供しています。

アプリケーションサーバ

アプリケーションの実行環境と運用の管理機能を提供するソフトウェアです。

イベント

画面上の操作に対して発生する通知のことです。ユーザーが画面部品に対して操作したときに発生するイベントと、画面そのものに対して操作したときに発生するイベント(グローバルイベント)の2種類があります。

イベントオブジェクト

イベントが発生してリスナが呼ばれた場合、リスナ関数の引数に渡されるものです。主なイベントオブジェクトに、アクションイベントとプロパティチェンジイベントがあります。

イベントハンドリング機能

画面に対するイベントリスナを管理し、イベントの発生時に適切なユーザーロジックを呼び出す機能です。クライアントフレームワークの機能の1つです。

イベントリスナ

イベントの受信を行うためのメソッドが定義されたインターフェースです。

イベントログ機能

通信フレームワークのクライアント側のJavaScriptから、サーバ側のログを出力する機能です。

画面部品

入力フィールド、チェックボックス、テーブルなど、画面に表示される機能を提供する部品です。

簡易通信方式

サーブレット連携機能において、アプリケーションを容易に構築するために利便性を重視した通信の方式です。アプリケーションは、JavaScriptアプリケーションとビジネスロジック間の通信にサーブレット連携機能のAPIを使用します。JavaScriptのオブジェクトとビジネスロジックのJavaBeanは、サーブレット連携機能の中で相互変換されます。

疑似要素

要素や属性等の文書構造で示される以外の箇所で、スタイルの適用が可能な要素です。

機能付加部品

入力フィールドへの入力時に文字列を補完するオートコンプリーション機能、フォーカス制御機能など、画面部品に対して機能を付加する部品です。

機能部品

データモデルを定義するなど、画面内部に機能をもつ部品です。

クライアント通信API

ブラウザ上のJavaScriptアプリケーションから、Webサービスを呼び出す機能です。

クライアントフレームワーク

Webブラウザ上のJavaScriptで動作するAjaxフレームワークの機能です。クライアントフレームワークには、以下の4つの機能があります。

- UI部品(画面部品、機能部品、機能付加部品)
- モデルバインディング機能
- イベントハンドリング機能
- グローバルイベント制御機能

クロスドメイン制約

JavaScriptのXMLHttpRequestオブジェクトを利用してサーバと通信する場合、セキュリティを確保するため、JavaScriptを読み込んだページがあるサーバ以外にはアクセスできません。この制約をクロスドメイン制約と呼びます。

グローバルイベント

ファンクションキーを押した場合など、画面そのものに対して操作したときに発生するイベントのことです。

グローバルイベント制御機能

画面そのものに対して操作したときに発生するイベントを制御する機能です。F1～F12までのファンクションキーに対して、keydownおよびkeyupのイベントを検出することが可能です。クライアントフレームワークの機能の1つです。

サービス管理機能

ブラウザ上のJavaScriptアプリケーションからアクセスするWebサービスを管理する機能です。

サービス認証機能

Webサービスにアクセスする際に、認証情報を付加する機能です。

サーブレット

Webサーバ上で動作するJavaアプリケーションです。ダイナミックな処理結果を返却するアプリケーションの開発に適しています。

サーブレット連携機能

ブラウザ上のJavaScriptアプリケーションから、サーブレットを利用する機能です。

サロゲートペア

従来のUnicode(UTF-16)では未使用だった0xD800～0xDBFF(1024通り)を上位サロゲート、0xDC00～0xDFFF(1024通り)を下位サロゲートと規定し、上位サロゲート＋下位サロゲートの32ビットで1文字を表現する方法です。

自動脱出機能

最大文字数を超えて文字列が入力されたときに、次の画面部品に自動的にフォーカスが移動する機能です。
自動脱出機能を利用するには、対象部品に最大入力文字数、およびフォーカスの移動先を指定しておく必要があります。

スクレイピング機能

各サービスから取得したデータのうち、実際に必要なデータだけを切り出す機能です。

スクレイピングツール

Ajaxフレームワークアプリケーションで利用するWebアプリケーションをGUIベースでスクレイピングし、XSLファイルを作成するツールです。
EclipseおよびInterstage Studioワークベンチのプラグインとして動作します。

セキュリティ機能

リクエストの正当性を確認する機能です。

セキュリティロール

Interstage管理コンソールを使用するユーザーに割り当てられた権限です。
セキュリティロールによって、ユーザーをグループ分けすることができます。

セション

サーバとクライアントとの接続を表します。クライアントごとに1つのセションが作成されます。

遅延読込み

画面の一部分を別に用意しておき、ユーザーが画面を操作している間に、裏で画面情報を読み込むことをいいます。

通信フレームワーク

Webブラウザで動作するJavaScriptアプリケーションとサーバ側のJavaで記述されたビジネスロジックを呼び出すためのAjaxフレームワークの機能です。通信フレームワークには、以下の機能があります。

- Apcoordinator連携機能
- サーブレット連携機能
- データ型変換機能
- イベントログ機能

データBean

クライアントとビジネスクラスの間のデータ受渡しを行うJavaBeans形式のクラスです。

データBean共有

AjaxフレームワークアプリケーションとApcoordinatorアプリケーションとで、データBeanを共有するための機能です。データBeanを共有することにより、Apcoordinatorアプリケーションで利用するデータBeanの情報を、Ajaxフレームワークアプリケーションで更新することができます。

データ型変換機能

クライアントのJavaScriptとサーバのJava間での双方向オブジェクト変換機能です。通信フレームワークでは、メソッド実行前後に、データ型変換機能を利用してオブジェクト変換を行います。

データ形式変換機能

アダプタから受け取ったXML情報をクライアントのJavaScriptで使えるように、JSON形式に変換する機能です。

汎用通信方式

サーブレット連携機能において、アプリケーションの自由度を重視した通信の方式です。アプリケーションは、JavaScriptのオブジェクトとビジネスロジックのJavaBeanとの変換にサーブレット連携機能のAPIを使用します。JavaScriptアプリケーションとビジネスロジック間の通信は、WebブラウザとWebサーバの機能を使用して実装します。

ビジネスクラス

ビジネスロジックを記述するJavaのクラスをビジネスクラスと呼びます。

ビジネスメソッド

ビジネスクラス内に記述されたメソッドで、クライアントから送信されたデータを処理します。

ビジネスロジック

アプリケーションが実現する業務処理そのものをビジネスロジックと呼びます。

非同期通信

送信側と受信側とでデータ送出のタイミングを合わせる必要がなく、任意のタイミングでデータを送信する通信方式です。

プロパティチェンジイベント

モデルの値が変更されたときに発生するイベントです。

マッシュアップ

複数の異なるWebサービスやアプリケーションを組み合わせて、1つの新しいサービスを形作ることです。

マッシュアップ定義ファイル

マッシュアップフレームワークの動作環境を設定するXML形式のファイルです。

マッシュアップフレームワーク

WebブラウザとWebサービスの通信を中継するAjaxフレームワークの機能です。複数のWebサービスを組み合わせたWebアプリケーションの作成が可能となります。
マッシュアップフレームワークには、以下の機能があります。

- クライアント通信API
- マッシュアッププロキシ
- アダプタ

マッシュアッププロキシ

クライアント通信APIからの呼出しに応じて、様々なWebサービスから情報を取得する機能です。
マッシュアッププロキシでは、Webサービスを呼び出すために以下の機能を提供します。

- サービス管理機能
- サービス認証機能
- データ形式変換機能
- セキュリティ機能
- アクセスログ機能

モデルオブジェクト

ユーザーアプリケーションが利用するデータを定義したものです。モデルオブジェクトはモデル定義部品で定義します。

モデルバインディング機能

画面部品から入力されたデータをモデルオブジェクトに反映したり、モデルオブジェクト内のデータが更新されたときに画面部品に値を反映したりする機能です。クライアントフレームワークの機能の1つです。

ユーザー認証

ユーザーを、ユーザーID/パスワードや証明書により判定し確認する処理です。

ユーザーロジック

サーバとの通信量を最適化するためにJavaScriptで実装したアプリケーションです。

ログ出力レベル

ログのレベルは、ログの詳細度を表します。レベルには、高い順から、TRACE、INFO、WARN、ERRORの4種類があります。特定のレベル以下のログだけを出力するように設定して、不要なログを抑止することができます。

ワークベンチ

EclipseやInterstage Studioが提供する、アプリケーションを開発するための統合開発環境です。プロジェクトの定義から実行まで、すべての開発作業を行うことができます。

索 引

[ま]

[や]

[ら]