目次

改訂情報

変更年月日	変更内容
2013-10-01	初版
2014-04-01	第２版　下記を追加・変更しました ■ 「*スクリプト開発モデルで業務処理を作成する*」 にコラムを追記しました。

はじめに

本書の目的

本書では intra-mart Accel Platform における Webサービス を、スクリプト開発モデルによって作成、提供する方法について説明します。

説明範囲は以下のとおりです。

- Webサービス・プロバイダ の作成方法とサンプルの解説
- Webサービス・クライアント の作成方法とサンプルの解説

対象読者

本書では次の利用者を対象としています。

- intra-mart Accel Platform の Webサービス を理解している
- Webサービス・プロバイダ の提供を行うアプリケーションの開発者
- Webサービス・クライアント を利用したアプリケーションの開発者

注意事項

1. Webサービス を利用するにあたり、いくつかの制限事項が存在します。制限事項についての詳細は 「リリースノート 制限事項」を参照してください。
2. 本書で解説するチュートリアルの資材は、一部「Webサービス Java開発プログラミングガイド」と重複しています。
   本書で作成した資材と「Webサービス Java開発プログラミングガイド」で作成した資材を同時にデプロイすると、サンプルが動作しないことがあります。

本書の構成

- *概要*

  スクリプト開発モデルにおける Webサービス の概要情報について説明します。

- *Web サービス・プロバイダ の作成*

  スクリプト開発モデルを利用して、 intra-mart Accel Platform 上に Webサービス をデプロイする方法について説明します。

- *Web サービス・クライアント の作成*

  スクリプト開発モデルを利用して、 intra-mart Accel Platform 上にデプロイされた Webサービス を利用する方法について説明します。

- *付録*

  Webサービス を開発する上で発生する問題と解決方法や、追加のサンプルコードなどを収録しています。

概要

Webサービスとは

本書では、「Webサービス」とは SOAP と WSDL を用いた Webサービス を指します。
Webサービス の詳細については、「Web サービス 認証・認可 仕様書」を参照してください。

スクリプト開発モデルで Webサービス を作成する利点

Webサービス を開発する際にスクリプト開発モデルで業務処理を記述することの利点は、以下の通りです。

- 既存のファンクションコンテナを Webサービス として再利用できます。
- 外部システムとの連携時に JavaScript が利用できます。
- 再デプロイやサーバの再起動をすることなく、業務処理の変更を即座に反映できます。

本書のチュートリアルを進める上での注意点

- チュートリアルでは e Builder を使用します。

  チュートリアルで作成する資材を開発・デプロイするために、e Builder を使用します。
  あらかじめご用意ください。

- JavaBean の簡単な知識が必要です。

  JavaBean とは、再利用可能なプログラムを可能にするために、ある機能をまとめた Java のクラスです。
  JavaBean では、一定の作法に従って記述することにより、JavaScript のオブジェクトのようなプロパティという概念を実現しています。
  JavaBean でプロパティを定義するためには、プロパティの値を格納する private のフィールドと、読み書きを行うアクセサメソッド (getter, setter) を用意します。

  本書のスクリプト開発モデルにおける Web サービス化に関しては、以下の型とその固定長配列をプロパティとする JavaBean のみを扱います。

  - java.lang.String
  - java.lang.Double
  - java.lang.Boolean
  - java.util.Date

  JavaBean に関するイベント、および、アクセサメソッド以外のメソッドは扱いません。

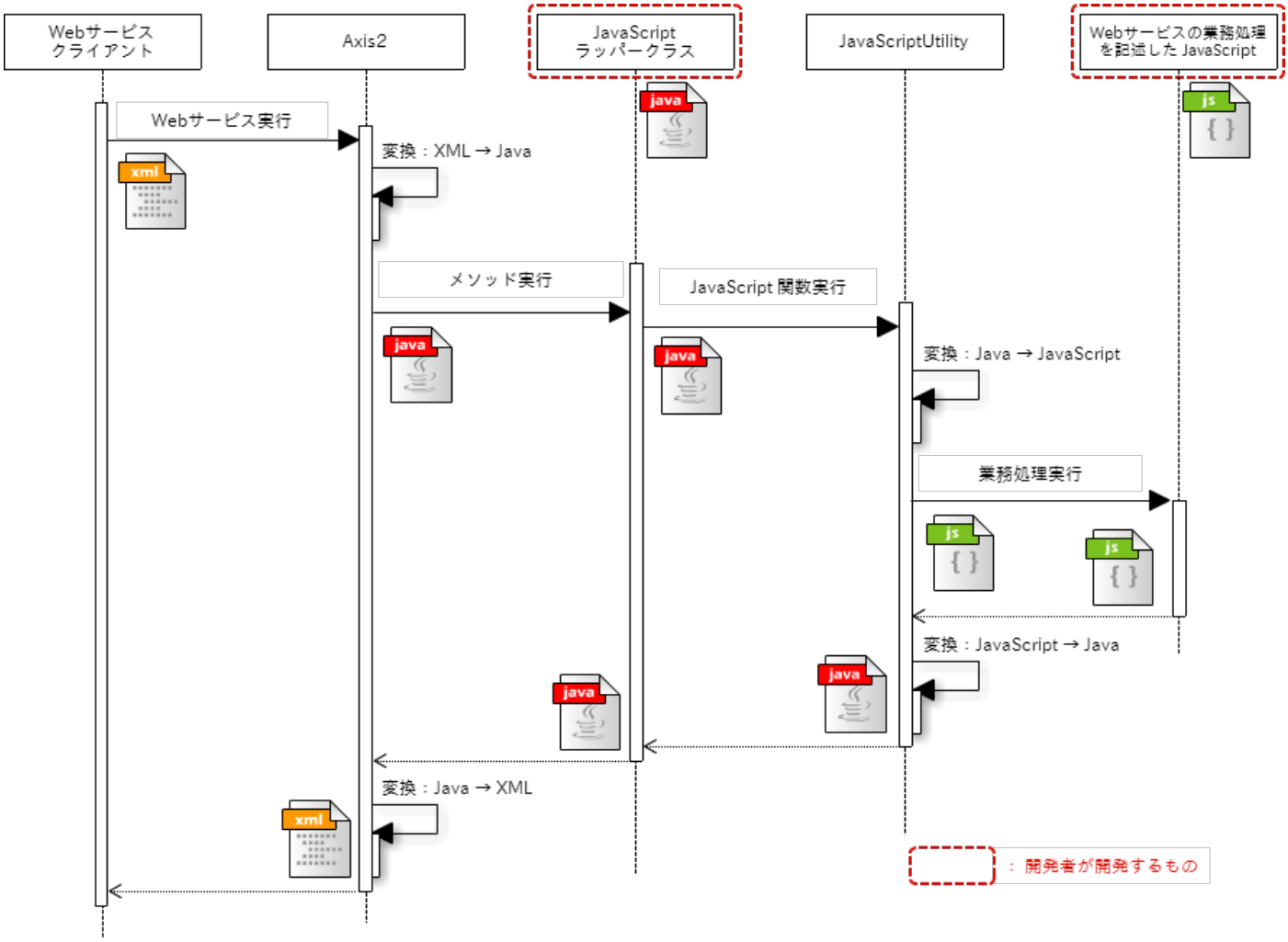
Webサービス・プロバイダ の作成

項目

作成手順の概要

この章では、スクリプト開発モデルのファンクションコンテナを Web サービスとして公開する手順を説明します。
JavaScript 関数の Web サービス化は、関数を実行するための Java クラス（以下、「JavaScript ラッパークラス」）を作成し、その Java クラスを Webサービス として公開することで実現します。

Web サービスとして公開された JavaScript 関数が実行されるまでの流れを以下に示します。



1. クライアントが、Web サービスの実行を要求します。
2. Web サービス実行エンジン「Apache Axis2」が、受け付けたリクエストに該当する JavaScript ラッパークラスのメソッドを呼び出します。
3. JavaScript ラッパークラスは、「JavaScriptUtility クラス」を利用して JavaScript 関数を実行します。
4. 以降、実行結果が適宜変換され、クライアントに返却します。

「JavaScriptUtility クラス」は、この「型情報クラス」を利用することで、Java 形式から JavaScript 形式への変換、および、その逆変換を自動的に行い、 JavaScript 関数を実行します。

Webサービス のデプロイを準備する

開発環境を用意する

最初に Web サービスを提供するための資材を作成する開発環境を用意します。
このチュートリアルでは e Builder を使用して、以下のプロジェクトを作成し、開発を行う手順を説明します。

グループID	mypackage （デフォルトの設定を使用）
バージョン	1.0.0 （デフォルトの設定を使用）
プロジェクト名	sample_provider

まず、e Builder、Resin、および、intra-mart Accel Platform をインストールして開発環境を構築してください。
インストール手順は「e Builder セットアップガイド 」を参照してください。

> **注意**
>
> intra-mart Accel Platform をインストールする際、ベースモジュールから「Webサービス 認証・認可」と「Webサービス 認証・認可クライアント」モジュールを選択してください。
> 選択しない場合、チュートリアルの Java コードでコンパイルエラーが発生します。

e Builder のインストールが完了したら、以下の手順に従って、「Module Project」でプロジェクトを作成します。
詳しくは「e Builder アプリケーション開発ガイドのモジュール・プロジェクト作成 」を参照してください。

1. 「ファイル」ー「新規」ー「プロジェクト」をクリックします。
2. 「e Builder」ー「Module Project」を選択して「次へ」をクリックします。
3. プロジェクト名に「sample_provider」を入力して、「終了」をクリックします。

プロジェクトの作成が完了したら、intra-mart Accel Platform の API を使用できるようにするために、プロジェクトの設定を行います。
詳しくは「e Builder アプリケーション開発ガイドのモジュール開発の基本機能 」を参照してください。

1. プロジェクトを右クリックして「プロパティ」を選択します。
2. 「e Builder」ー「Module Assembly」を選択します。
3. Web アーカイブディレクトリに、war を展開してできたコンテキストパスと同名のフォルダを選択します。
4. リソース変更時の自動デプロイ先の一覧で、全てのチェックボックスを外します。
5. 「OK」をクリックします。

依存関係を解決する

プロジェクトの設定が完了したら、依存関係の修正を行います。
Webサービス・プロバイダ を作成するためには、「Webサービス 認証・認可」モジュールに依存する必要があります。

以下の手順に従って、プロジェクトの依存関係を修正します。

1. 作成したプロジェクトのルートディレクトリに配置されている module.xml をダブルクリックします。
2. 「依存関係」タブを開き、「追加」をクリックします。
3. 以下の内容を入力して、「OK」をクリックします。

ID	jp.co.intra_mart.im_ws_auth
バージョン	8.0.2

> **コラム**
>
> 基本的にバージョンはサポートが行われている番号を指定します。
> 使用したい API が他のバージョンに含まれている場合、そのバージョン番号を指定してください。

4. module.xml ファイルを保存した後、「module.xml」タブを開き、不要なタグ（`<tags>`）を除去します。
   最終的に以下のようなソースになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<module conf:schemaLocation="urn:intramart:jackling:toolkit:configurations configurations.xsd"
      xmlns="urn:intramart:jackling:module" xmlns:conf="urn:intramart:jackling:toolkit:configurations"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemeLocation="urn:intramart:jackling:module module.xsd">

   <id>mypackage.sample_provider</id>
   <version>1.0.0</version>
   <type>module</type>
   <name>${module.name}</name>
   <vendor>${module.vendor}</vendor>
   <description>${module.description}</description>

   <dependencies>
      <dependency>
         <module-id>jp.co.intra_mart.im_ws_auth</module-id>
         <verified-version min="8.0.2">8.0.2</verified-version>
      </dependency>
   </dependencies>
</module>
```

5. 「依存関係の階層」タブを開き、依存関係が解決されさまざまなモジュールが表示されていれば成功です。

スクリプト開発モデルで業務処理を作成する

プロジェクトの準備が完了したら、Web サービスとして公開する JavaScript を作成し、業務処理を記述します。
ここでは、サンプルとして `sample/web_service/provider/member_info_operator.js` を用意します。
この JavaScript ソース内に定義されている関数「add()」、「find()」、および、「findAll()」を Web サービスとして公開します。

member_info_operator.js は、以下の形式のメンバー情報を保存・検索します。

プロパティ	説明 (JavaScript 型)
id	メンバーID（String）
name	メンバー名（String）
age	年齢（Number）
married	既婚の場合 true、未婚の場合 false（Boolean）
birthDate	生年月日（Date）
children	子供情報（メンバー情報形式の配列）

member_info_operator.js ファイルを、プロジェクトの以下の場所に作成します。

- `src/main/jssp/src/sample/web_service/provider/member_info_operator.js`

member_info_operator.js のソースは以下の通りです。

```
var DOMAIN = 'sample_web_service';
var GROUP = 'sample_member_info';

/**
 * メンバー情報の追加
 */
function add(member) {
    var message = 'member_info_operator.js#add() が実行されました。';
    Debug.console(message, member);

    // アカウントコンテキストからアカウントの情報が取得できます。
    var loginUserCd = Contexts.getAccountContext().userCd;
    Debug.print('ユーザID「' + loginUserCd + '」でログインしています');

    // 「.(ドット)」でプロパティにアクセスすることも可能です。
    Debug.print(member.name + '\'s birthDate: ' + member.birthDate);

    // Parmanent に保存
    var permanent = new Permanent(DOMAIN, GROUP);
    permanent.set(member.id, member);
}

/**
 * メンバー情報の検索
 */
function find(id) {
    var message = 'member_info_operator.js#find() が実行されました';
    Debug.console(message, id);

    // Parmanent から読み込み
    var permanent = new Permanent(DOMAIN, GROUP);
    var member = permanent.get(id);
    return member;
}

/**
 * 全てのメンバー情報の検索
 */
function findAll() {
    var message = 'member_info_operator.js#findAll() が実行されました';
    Debug.console(message);

    var memberArray = new Array();

    // Parmanent から読み込み
    var permanent = new Permanent(DOMAIN, GROUP);
    var allKeys = permanent.keys();
    if (allKeys == null) {
        var soapFault = new SOAPFault('Parmanent にデータが登録されていません');
        // SOAPFault をスロー(ここで処理が終了します)
        soapFault.throwFault();
    }

    var max = allKeys.length;
    for (var idx = 0; idx < max; idx++) {
        var key = allKeys[idx];
        // Parmanent から読み込み
        var member = permanent.get(key);
        memberArray.push(member);
    }

    return memberArray;
}
```

> **コラム**
>
> メンバー情報の保存は Permanent API を利用します。
>
> SOAPFault オブジェクトをスローすることで、オブジェクトに設定した内容を Webサービス・クライアント に返信することができます。
> SOAPFault オブジェクトの詳細は、「API リストの SOAPFault オブジェクト」を参照してください。

型情報クラスを作成する

Web サービスの業務処理の作成が完了したら、JavaScript 関数の引数、および、返却値のオブジェクト変換に必要な「型情報クラス」を作成します。
「型情報クラス」とは、JavaScript オブジェクトのプロパティ構成を JavaBean で表現した単純なクラスです。
ここでは、member_info_operator.js で利用するメンバー情報の形式を JavaBean として作成します。

以下の手順に従って、e Builder で型情報クラスを作成します。

1. プロジェクトを右クリックして、「新規」ー「クラス」を選択します。
2. 以下の内容を入力して、「OK」をクリックします。

パッケージ	sample.web_service.provider
名前	Member

3. Member.java がプロジェクトの `src/main/java` 配下に作成されます。

```
package sample.web_service.provider;

public class Member {

}
```

> **注意**
>
> Web サービスとして公開するメソッドの引数、および、返却値に、継承関係を持つクラスを使用しないでください。
> 継承関係を持ったクラスを使用すると、Web サービスのクライアント側でエラーの原因になります。
> これは、Java オブジェクトが XML に変換される際、XML 名前空間がサブクラスで統一される ADB（Axis Data Binding）の仕様による制限です。
>
> 例えば、以下の SubModel が ParentModel の子クラスとして定義されている場合、SubModel は Web サービスとして公開するメソッドの引数、および、返却値として使用できません。
>
> - sample.foo.ParentModel
> - sample.bar.SubModel

4. メンバー情報の各プロパティを保持するための private 変数を定義します。

```
package sample.web_service.provider;

import java.util.Date;

public class Member {
    private String id;

    private String name;

    private Double age;

    private Boolean married;

    private Date birthDate;

    private Member[] children;
}
```

注意

プロパティの型が JavaScript の Number 型の場合、型情報クラスでは Double 型を使用してください。
上記サンプルでは、変数「age」が該当します。

Java と JavaScript の相互変換のため、使用できる型が限定されています。
詳細は「API リストの JavaScriptUtility クラス」に記載されている「javaBeanToJS」および「jsToJavaBean」を参照してください。

5. 各プロパティのアクセサメソッド（getter, setter）を追加します。

コラム

以下の手順に従って操作すると、簡単にアクセサメソッドを追加できます。

1. 変数を選択後、右クリックして「ソース」ー「Getter および Setter の生成」を選択します。
2. 「すべて選択」をクリックして、「OK」ボタンをクリックします。

```
package sample.web_service.provider;

import java.util.Date;

public class Member {
    private String id;

    private String name;

    private Double age;

    private Boolean married;

    private Date birthDate;

    private Member[] children;

    public Double getAge() {
        return age;
    }

    public Date getBirthDate() {
        return birthDate;
    }

    public Member[] getChildren() {
        return children;
    }

    public String getId() {
        return id;
    }

    public Boolean getMarried() {
        return married;
    }

    public String getName() {
        return name;
    }

    public void setAge(final Double age) {
        this.age = age;
    }

    public void setBirthDate(final Date birthDate) {
        this.birthDate = birthDate;
    }

    public void setChildren(final Member[] children) {
        this.children = children;
    }

    public void setId(final String id) {
        this.id = id;
    }

    public void setMarried(final Boolean married) {
        this.married = married;
    }

    public void setName(final String name) {
        this.name = name;
    }
}
```

JavaScript ラッパークラスを作成する

型情報クラスの作成が完了したら、JavaScript 関数を実行するための JavaScript ラッパークラスを作成します。
ここでは、member_info_operator.js の関数「add()」、「find()」、および、「findAll()」を実行する Java クラスを作成します。

以下の手順に従って、e Builder で型情報クラスを作成します。

1. プロジェクトを右クリックして、「新規」ー「クラス」を選択します。
2. 以下の内容を入力して、「OK」をクリックします。

パッケージ	sample.web_service.provider
名前	MemberInfoOperatorService

3. MemberInfoOperatorService.java がプロジェクトの src/main/java 配下に作成されます。

```
package sample.web_service.provider;

public class MemberInfoOperatorService {

}
```

4. JavaScript 関数の呼び出し処理を追加します。

> **コラム**
>
> JavaScript 関数の呼び出しには、jp.co.intra_mart.jssp.util.JavaScriptUtility クラスの executeFunction() および executeVoidFunction() メソッドを利用します。
> このメソッドを利用することで、関数のパラメータが Java 形式から JavaScript 形式へ自動的に変換されます。
> 同様に、関数の実行結果も JavaScript 形式から Java 形式へ自動的に変換されます。
>
> 変換後の Java クラスを配列で指定する場合は、固定長配列で指定します。
> 例えば、Member クラスの配列を変換後のクラスに指定する場合は、以下のようになります。
> `final Member[] members = (Member[]) JavaScriptUtility.executeFunction(pagePath, functionName, Member[].class);`

```
package sample.web_service.provider;

import jp.co.intra_mart.foundation.web_service.auth.WSUserInfo;
import jp.co.intra_mart.jssp.util.JavaScriptUtility;

import org.apache.axis2.AxisFault;

public class MemberInfoOperatorService {
   public Boolean add(final WSUserInfo wsUserInfo, final Member member) throws AxisFault {
      try {
         final String pagePath = "sample/web_service/provider/member_info_operator";
         final String functionName = "add";
         JavaScriptUtility.executeVoidFunction(pagePath, functionName, member);
         return true;
      } catch (final Exception ex) {
         throw AxisFault.makeFault(ex);
      }
   }

   public Member find(final WSUserInfo wsUserInfo, final String id) throws AxisFault {
      try {
         final String pagePath = "sample/web_service/provider/member_info_operator";
         final String functionName = "find";
         final Member member = (Member) JavaScriptUtility.executeFunction(pagePath, functionName, Member.class, id);
         return member;
      } catch (final Exception ex) {
         throw AxisFault.makeFault(ex);
      }
   }

   public Member[] findAll(final WSUserInfo wsUserInfo) throws AxisFault {
      try {
         final String pagePath = "sample/web_service/provider/member_info_operator";
         final String functionName = "findAll";
         final Member[] members = (Member[]) JavaScriptUtility.executeFunction(pagePath, functionName,
Member[].class);
         return members;
      } catch (final Exception ex) {
         throw AxisFault.makeFault(ex);
      }
   }
}
```

services.xml を作成する

JavaScript ラッパークラスの作成が完了したら、Web サービスの設定ファイル「services.xml」を用意します。

services.xml ファイルを、プロジェクトの以下の場所に作成します。

- src/main/webapp/WEB-INF/services/sample_member_info/META-INF/services.xml

services.xml のソースは以下の通りです。

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceGroup>
  <service name="SampleMemberInfoOperatorService">
    <parameter name="ServiceClass">sample.web_service.provider.MemberInfoOperatorService</parameter>
    <module ref="im_ws_auth"/>
    <messageReceivers>
      <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-only"
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver" />
      <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
    </messageReceivers>

    <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>

    <operation name="add">
      <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>
    </operation>

    <operation name="find">
      <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>
    </operation>

    <operation name="findAll">
      <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>
    </operation>

  </service>
</serviceGroup>
```

コラム

各関数に、アクセス権限を設定するための認可の「リソースURI」を設定します。
最初はサンプルの動作を確認するために、未認証・認証済みを問わず、どのユーザでもアクセス可能なリソースURIを割り当てます。

- service://intra-mart.jp/public-resources/welcome-to-intramart

実際に Web サービスを業務利用する場合は、個別の「リソースURI」を用意してください。
権限設定の詳細は、後述の「*認可を利用してアクセス権限を設定する* 」を参照してください。

Webサービス をデプロイする

資材をデプロイする

以上で Web サービスを提供するための資材が完成しました。
次に作成したモジュールをユーザモジュールとして取り込み、war を作成して Resin にデプロイします。

以下の手順に従って、e Builder でユーザモジュールを作成します。

1. プロジェクトを右クリックして、「エクスポート」を選択します。
2. 「e Builder」ー「imm file」を選択して、「次へ」をクリックします。
3. 出力先フォルダに任意の場所を選択して、「終了」をクリックします。
4. しばらくすると、出力先フォルダに sample_provider-1.0.0.imm ファイルが作成されます。

次に、以下の手順に従って、e Builder で imm ファイルをユーザモジュールとして取り込みます。

1. e Builder で環境構築時に利用したプロジェクトの「juggling.im」を開きます。
2. 「ユーザモジュール」タブを開き、右上の「モジュールを追加します」アイコンをクリックします。
3. e Builder で作成した sample_provider-1.0.0.imm ファイルを選択して開きます。

> **コラム**
>
> 依存関係が不足している場合、上側にエラーメッセージが表示されます。
> この場合、エラーメッセージをクリックして不足しているモジュールを追加してください。

4. juggling.im を保存して、環境構築時と同じ手順で war を作成し、Resin にデプロイします。
5. Resin を再起動します。

次に、以下の手順に従って、テナント環境セットアップを実施します。

1. システム管理画面を開き、システム管理者でログインします。
   http://<HOST>:<PORT>/<CONTEXT_PATH>/system/login
2. 「テナント環境セットアップ」をクリックします。

> **コラム**
>
> 「テナント環境は最新です。セットアップが必要なモジュールはありません。」が表示されている場合は、以降の操作は不要です。

3. 続けて「テナント環境セットアップ」をクリックします。
4. 確認メッセージで「決定」をクリックします。

デプロイされていることを確認する

Resin の再起動とテナント環境セットアップが完了したら、作成した Web サービスがデプロイされていることを確認します。

以下の手順に従って、正常にデプロイされていることを確認します。

1. 以下の URL にアクセスします。
   http://<HOST>:<PORT>/<CONTEXT_PATH>/services/listServices
2. 「SampleMemberInfoOperatorService」に「Service Status : Active」の文字列と、各関数名が表示されていることが確認できれば成功です。

認可を利用してアクセス権限を設定する

ここまでのチュートリアルでは、 Web サービスに割り当てた認可のリソースURIは、未認証・認証済みを問わず、どのユーザでもアクセス可能なものを割り当てました。

- service://intra-mart.jp/public-resources/welcome-to-intramart

Web サービスを正式版として提供する際は、Web サービスのアクセス権限を細かく設定できるようにするために、認可リソースを登録します。
Web サービスの場合、認可リソースのキーとなる「リソースURI」のスキームは、通常「service」を利用します。
例えば、以下のようにリソースURI を定義します。

- service://sample_provider/web_service/member_info_operator

Web サービスの各関数に対して個別に権限設定を分けて管理したい場合、各関数にリソースURI を定義します。
例えば、サンプルの Web サービスで公開した「add()」、「find()」、「findAll()」のそれぞれで権限設定を分けたい場合は、３つのリソースを作成するためにそれぞれ「リソースURI」を定義します。

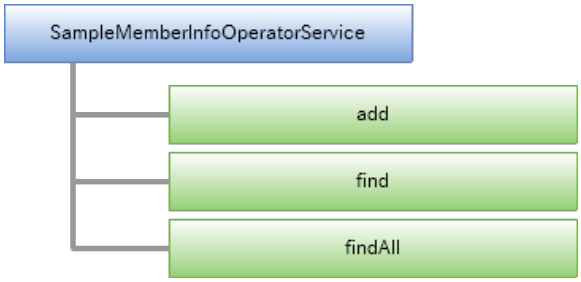
- service://sample_provider/web_service/member_info_operator/add
- service://sample_provider/web_service/member_info_operator/find
- service://sample_provider/web_service/member_info_operator/findAll

認可リソースを作成する

権限設定を行うための「リソースURI」が決定したら、テナント環境セットアップ資材を作成します。
必要な資材は以下の通りです。

- 認可リソース設定ファイル
- テナント環境セットアップを実施するためのセットアップ設定ファイル

この章では、認可に以下の構成でリソースを登録します。



> **コラム**
>
> 認可リソースは、テナント環境セットアップの資材からではなく、テナント管理機能の「認可設定画面」から登録することもできます。
> Web サービスの開発中に認可リソースを頻繁に変更する可能性がある場合は、認可設定画面から設定すると便利です。
> 認可設定画面を操作する方法についての詳細は、「テナント管理者操作ガイドの認可を設定する 」の「リソースを追加する」を参照してください。

それぞれ必要な資材を、プロジェクトの以下の場所に作成します。 サンプルの資材内容は以下の通りです。

- 認可リソース設定ファイル

  src/main/storage/system/products/import/basic/sample_provider/sample_provider-authz-resource.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.intra-mart.jp/authz/imex/resource">

  <authz-resource id="sample_provider-service"
uri="service://sample_provider/web_service/member_info_operator">
    <display-name>
      <name locale="ja">SampleMemberInfoOperatorService</name>
    </display-name>
    <parent-group id="web-services" />
  </authz-resource>

  <authz-resource uri="service://sample_provider/web_service/member_info_operator/add">
    <display-name>
      <name locale="ja">add</name>
    </display-name>
    <parent-group id="sample_provider-service" />
  </authz-resource>

  <authz-resource uri="service://sample_provider/web_service/member_info_operator/find">
    <display-name>
      <name locale="ja">find</name>
    </display-name>
    <parent-group id="sample_provider-service" />
  </authz-resource>

  <authz-resource uri="service://sample_provider/web_service/member_info_operator/findAll">
    <display-name>
      <name locale="ja">findAll</name>
    </display-name>
    <parent-group id="sample_provider-service" />
  </authz-resource>

</root>
```

コラム

Web サービス自身と、各関数に割り当てた「リソースURI」分、認可リソースを作成します。

Web サービスに関係する認可リソースを登録するための親リソースグループ「web-services」が初期状態で用意されています。
そのため、サンプルの Web サービスのトップ階層となる「SampleMemberInfoOperatorService」の親リソースグループを「web-services」に設定します。

- テナント環境セットアップを実施するためのセットアップ設定ファイル

src/main/conf/products/import/basic/sample_provider/import-sample_provider-config-1.xml

```
<import-data-config xmlns="http://intra_mart.co.jp/system/service/provider/importer/config/import-data-config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://intra_mart.co.jp/system/service/provider/importer/config/import-data-config import-data-
config.xsd">

  <tenant-master>
    <authz-resource-file>products/import/basic/sample_provider/sample_provider-authz-resource.xml</authz-
resource-file>
  </tenant-master>

</import-data-config>
```

services.xml を変更する

セットアップ資材の作成が完了したら、Web サービスの各関数にアクセスするための権限設定ファイル（services.xml）を変更します。

- src/main/webapp/WEB-INF/services/sample_member_info/META-INF/services.xml

service://intra-mart.jp/public-resources/welcome-to-intramart の部分を、定義した「リソースURI」に書き換えます。
services.xml のソースは以下の通りです。

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceGroup>
  <service name="SampleMemberInfoOperatorService">
    <parameter name="ServiceClass">sample.web_service.provider.MemberInfoOperatorService</parameter>
    <module ref="im_ws_auth"/>
    <messageReceivers>
      <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-only"
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver" />
      <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
    </messageReceivers>

    <parameter name="authz-uri">service://sample_provider/web_service/member_info_operator</parameter>

    <operation name="add">
      <parameter name="authz-uri">service://sample_provider/web_service/member_info_operator/add</parameter>
    </operation>

    <operation name="find">
      <parameter name="authz-uri">service://sample_provider/web_service/member_info_operator/find</parameter>
    </operation>

    <operation name="findAll">
      <parameter name="authz-uri">service://sample_provider/web_service/member_info_operator/findAll</parameter>
    </operation>

  </service>
</serviceGroup>
```

> **コラム**
>
> services.xml の設定内容については、「Web サービス 認証・認可 仕様書の services.xml について」もあわせて参照してください。

資材を再デプロイする

Web サービスの設定ファイルの更新が完了したら、再デプロイを行います。
「*資材をデプロイする* 」の手順に従って、 sample_provider-1.0.0.imm を再デプロイしてください。

> **注意**
>
> デプロイ後、システム管理画面から、テナント環境セットアップを必ず実行してください。

認可で権限設定を行う

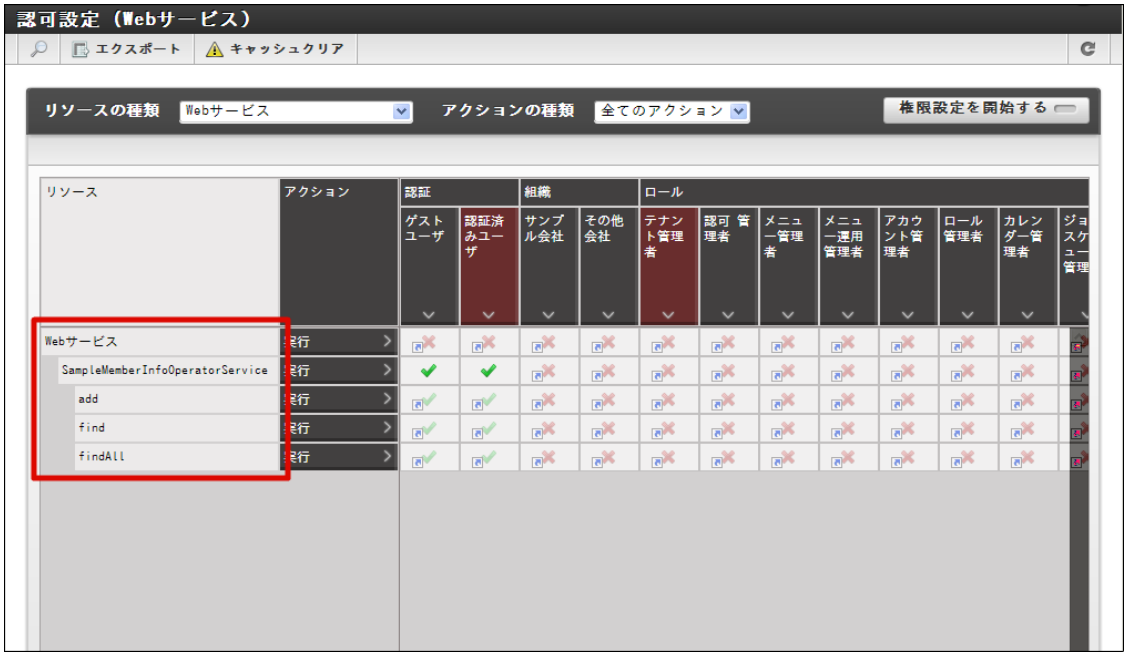
再デプロイが完了したら、認可設定画面を開いて実際に Web サービスに対してアクセス権限を設定します。

以下の手順に従って、アクセス権限を設定します。

1. 一般利用者のログイン画面を開き、テナント管理者でログインします。
   http://<HOST>:<PORT>/<CONTEXT_PATH>/login
2. サイトマップを開き、「テナント管理」カテゴリから「認可」をクリックします。

3. 「リソースの種類」から「Web サービス」を選択します。

4. 認可設定のグリッドに「SampleMemberInfoOperatorService」とその配下に「add」、「find」、および、「findAll」が表示されていることを確認します。



5. 「add」、「find」、および、「findAll」に対して、任意の対象者条件に権限を設定します。

> **コラム**
>
> 権限の設定方法についての詳細は、「テナント管理者操作ガイドの認可を設定する 」を参照してください。
>
> 実際に設定したアクセス権限通りに動作するかどうかを確認するためには、「Webサービス・クライアント」を用意する必要があります。
> 「*Webサービス・クライアント の作成*」ではスクリプト開発モデルを使用して Webサービス・クライアント を作成する方法を記載していますので、参照してください。

Webサービス・クライアント の作成

項目

作成手順の概要

この章では、Web サービスとして公開された関数を、スクリプト開発モデルから利用する手順を説明します。
スクリプト開発モデルでは、Web サービスを呼び出すための API 「SOAPClient オブジェクト」が用意されています。
SOAPClient オブジェクトを利用することにより、XML や Java を意識することなく、Web サービスを呼び出すことができます。

SOAPClient オブジェクトを利用した Web サービスの呼び出しは、以下の３つの手順で実現できます。

1. WSDL を指定して SOAPClient オブジェクトのインスタンスを生成します。
2. SOAPClient オブジェクトの引数に渡す WSUserInfo オブジェクトを生成します。
3. SOAPClient オブジェクトを利用して Web サービスにアクセスします。

このチュートリアルでは、「*Webサービス・プロバイダ の作成*」で解説されている Web サービスが呼び出されるまでを解説します。

> **コラム**
>
> SOAPClient オブジェクトを利用するための設定が用意されています。
> SOAPClient の設定についての詳細は、「設定ファイルリファレンスの SOAPClient オブジェクトの設定」を参照してください。

Webサービス へアクセスする画面を作成する

開発環境を用意する

Webサービス・プロバイダ の「*開発環境を用意する*」と同様に開発環境（e Builder、Resin、および、intra-mart Accel Platform）をインストールします。
このチュートリアルでは、以下のプロジェクトを作成し、開発を行う手順を説明します。

グループID	mypackage （デフォルトの設定を使用）
バージョン	1.0.0 （デフォルトの設定を使用）
プロジェクト名	sample_client

e Builder のインストールが完了したら、以下の手順に従って、「Module Project」でプロジェクトを作成します。
詳しくは「e Builder アプリケーション開発ガイドのモジュール・プロジェクト作成」を参照してください。

1. 「ファイル」ー「新規」ー「プロジェクト」をクリックします。
2. 「e Builder」ー「Module Project」を選択して「次へ」をクリックします。
3. プロジェクト名に「sample_client」を入力して、「終了」をクリックします。

プロジェクトの作成が完了したら、intra-mart Accel Platform の API を使用できるようにするために、プロジェクトの設定を行います。

詳しくは「e Builder アプリケーション開発ガイドのモジュール開発の基本機能 」を参照してください。

1. プロジェクトを右クリックして「プロパティ」を選択します。
2. 「e Builder」ー「Module Assembly」を選択します。
3. Web アーカイブディレクトリに、war を展開してできたコンテキストパスと同名のフォルダを選択します。
4. リソース変更時の自動デプロイ先の一覧で、全てのチェックボックスを外します。
5. 「OK」をクリックします。

依存関係を解決する

プロジェクトの設定が完了したら、依存関係の修正を行います。
Webサービス・プロバイダ にアクセスするためには、「Webサービス 認証・認可クライアント」モジュールに依存する必要があります。

以下の手順に従って、プロジェクトの依存関係を修正します。

1. 作成したプロジェクトのルートディレクトリに配置されている module.xml をダブルクリックします。
2. 「依存関係」タブを開き、「追加」をクリックします。
3. 以下の内容を入力して、「OK」をクリックします。

ID	jp.co.intra_mart.im_ws_auth_client
バージョン	8.0.2

> **コラム**
>
> 基本的にバージョンはサポートが行われている番号を指定します。
> 使用したい API が他のバージョンに含まれている場合、そのバージョン番号を指定してください。

4. module.xml ファイルを保存した後、「module.xml」タブを開き、不要なタグ（`<tags>`）を除去します。
   最終的に以下のようなソースになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<module conf:schemaLocation="urn:intramart:jackling:toolkit:configurations configurations.xsd"
      xmlns="urn:intramart:jackling:module" xmlns:conf="urn:intramart:jackling:toolkit:configurations"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemeLocation="urn:intramart:jackling:module module.xsd">

   <id>mypackage.sample_client</id>
   <version>1.0.0</version>
   <type>module</type>
   <name>${module.name}</name>
   <vendor>${module.vendor}</vendor>
   <description>${module.description}</description>

   <dependencies>
      <dependency>
         <module-id>jp.co.intra_mart.im_ws_auth_client</module-id>
         <verified-version min="8.0.2">8.0.2</verified-version>
      </dependency>
   </dependencies>
</module>
```

5. 「依存関係の階層」タブを開き、依存関係が解決されさまざまなモジュールが表示されていれば成功です。

スクリプト開発モデルで業務処理を作成する

プロジェクトの準備が完了したら、Web サービスにアクセスするための画面をスクリプト開発モデルで作成します。
ここでは、サンプルとして `sample/web_service/client/member_info_operator.html` と

sample/web_service/client/member_info_operator.js を用意します。

member_info_operator.html ファイルと member_info_operator.js ファイルを、プロジェクトの以下の場所に作成します。

- src/main/jssp/src/sample/web_service/client/member_info_operator.html
- src/main/jssp/src/sample/web_service/client/member_info_operator.js

member_info_operator.html のソースは以下の通りです。

```
<imart type="head">
<script type="text/javascript">
(function($) {
 $(function() {
  /**
   * メンバー情報の追加
   */
  $('#add').click(function() {
   var userCd = $('#userCd').val();
   var password = $('#password').val();
   var id = $('#memberId').val();
   var name = $('#memberName').val();
   var age = $('#memberAge').val();
   var married = $('#memberMarried').val();
   var birthDate = $('#memberBirthDate').val();

   try {
    $.ajax({
     async:false, cache:false, dataType:'text', type:'POST',
     url:'sample/web_service/client/member_info_operator/add', data:{
      'userCd':userCd,
      'password':password,
      'id':id,
      'name':name,
      'age':age,
      'married':married,
      'birthDate':birthDate
     },
     success:function(result) {
      imuiAlert(result == 'true' ? 'OK' : 'NG');
     },
     error:function() {
      imuiAlert('NG');
     }
    });
   } catch (ex) {
    imuiAlert('NG');
   }
  });

  /**
   * メンバー情報の検索
   */
  $('#find').click(function() {
   var userCd = $('#userCd').val();
   var password = $('#password').val();
   var id = $('#memberId').val();

   try {
    $.ajax({
     async:false, cache:false, dataType:'json', type:'POST',
     url:'sample/web_service/client/member_info_operator/find', data:{
      'userCd':userCd,
      'password':password,
      'id':id
     },
     success:function(result) {
      $('#memberId').val(result.id);
      $('#memberName').val(result.name);
      $('#memberAge').val(result.age);
```

```
          $('#memberMarried').val(result.married);
          $('#memberBirthDate').val(result.birthDate);
          imuiAlert('OK');
        },
        error:function() {
          imuiAlert('NG');
        }
       });
     } catch (ex) {
      imuiAlert('NG');
     }
    });

    /**
     * 全てのメンバー情報の検索
     */
    $('#findAll').click(function() {
     var userCd = $('#userCd').val();
     var password = $('#password').val();

     try {
      $.ajax({
       async:false, cache:false, dataType:'json', type:'POST',
       url:'sample/web_service/client/member_info_operator/findAll', data:{
        'userCd':userCd,
        'password':password
       },
       success:function(result) {
        imuiAlert(result.length);
       },
       error:function(a, b, c) {
        imuiAlert('NG');
       }
      });
     } catch (ex) {
      imuiAlert('NG');
     }
    });
  });
})(jQuery);
</script>
</imart>

<div class="imui-form-container-wide">
 <div class="imui-chapter-title">
  <h2>Web サービスにアクセスするユーザ情報</h2>
 </div>
 <table class="imui-form">
  <tbody>
   <tr>
    <th class="wd-20"><label class="imui-required">ユーザコード</label></th>
    <td><imart type="imuiTextbox" id="userCd" value="aoyagi" autofocus /></td>
   </tr>
   <tr>
    <th class="wd-20"><label class="imui-required">パスワード</label></th>
    <td><imart type="imuiTextbox" id="password" value="aoyagi" /></td>
   </tr>
  </tbody>
 </table>
 <div class="imui-chapter-title">
  <h2>登録するメンバー情報</h2>
 </div>
 <table class="imui-form">
  <tbody>
   <tr>
    <th class="wd-20"><label class="imui-required">ID (id)</label></th>
    <td><imart type="imuiTextbox" id="memberId" /></td>
   </tr>
   <tr>
    <th class="wd-20"><label>名前 (name)</label></th>
```

```
      <td><imart type="imuiTextbox" id="memberName" /></td>
    </tr>
    <tr>
      <th class="wd-20"><label>年齢 (age)</label></th>
      <td><imart type="imuiTextbox" id="memberAge" /></td>
    </tr>
    <tr>
      <th class="wd-20"><label>既婚フラグ (married) [true or false]</label></th>
      <td><imart type="imuiTextbox" id="memberMarried" /></td>
    </tr>
    <tr>
      <th class="wd-20"><label>誕生日 (birthDate) [yyyy-mm-dd 形式]</label></th>
      <td><imart type="imuiTextbox" id="memberBirthDate" /></td>
    </tr>
  </tbody>
 </table>
 <div class="imui-operation-parts">
  <imart type="imuiButton" id="add" value="add" class="imui-large-button" />
  <imart type="imuiButton" id="find" value="find" class="imui-large-button" />
  <imart type="imuiButton" id="findAll" value="findAll" class="imui-large-button" />
 </div>
</div>
```

member_info_operator.js のソースは以下の通りです。

```
// ホスト名、ポート番号、コンテキストパスは適宜置き換えてください。
var wsdlURL = 'http://localhost:8080/imart/services/SampleMemberInfoOperatorService?wsdl';

/**
 * 初期化
 */
function init() {
}

/**
 * 認証情報の取得
 */
function getWSUserInfo(userCd, password) {
  return {
    'userID':userCd,
    'password':WSAuthDigestGenerator4WSSE.getDigest(userCd, password),
    'authType':WSAuthDigestGenerator4WSSE.getAuthType(),
    'loginGroupID':'default' // 実際にはプロバイダから提供された接続先ログイングループID/テナントIDを設定します。
  };
}

/**
 * メンバー情報の追加
 */
function add(request) {
  try {
    var userCd = request.userCd;
    var password = request.password;
    var member = {
      'id':request.id,
      'name':request.name,
      'age':request.age,
      'married':request.married == 'true',
      'birthDate':DateTimeFormatter.parseToDate(DateTimeFormatter.STANDARD_DATE_FORMAT_PATTERN,
request.birthDate),
      'children':[]
    };

    var soapClient = new SOAPClient(wsdlURL);
    var wsUserInfo = getWSUserInfo(userCd, password);
    var result = soapClient.add(wsUserInfo, member);
    Debug.console('処理が成功しました。', result);
    outputText(result ? 'true' : 'false');
  } catch (ex) {
```

```
    } catch (ex) {
        Debug.console('エラーが発生しました。', ex);
        throw ex;
    }
}

/**
 * メンバー情報の検索
 */
function find(request) {
    try {
        var userCd = request.userCd;
        var password = request.password;
        var id = request.id;

        var soapClient = new SOAPClient(wsdlURL);
        var wsUserInfo = getWSUserInfo(userCd, password);
        var result = soapClient.find(wsUserInfo, id);
        Debug.console('処理が成功しました。', result);

        var member = {
            'id':result.id,
            'name':result.name,
            'age':result.age,
            'married':result.married ? 'true' : 'false',
            'birthDate':DateTimeFormatter.format(DateTimeFormatter.STANDARD_DATE_FORMAT_PATTERN, result.birthDate)
        };
        outputJSON(member);
    } catch (ex) {
        Debug.console('エラーが発生しました。', ex);
        throw ex;
    }
}

/**
 * 全てのメンバー情報の検索
 */
function findAll(request) {
    try {
        var i;
        var userCd = request.userCd;
        var password = request.password;

        var soapClient = new SOAPClient(wsdlURL);
        var wsUserInfo = getWSUserInfo(userCd, password);
        var result = soapClient.findAll(wsUserInfo);
        Debug.console('処理が成功しました。', result);

        var members = [];
        for (i = 0; i < result.length; i++) {
            members.push({
                'id':result[i].id,
                'name':result[i].name,
                'age':result[i].age,
                'married':result[i].married ? 'true' : 'false',
                'birthDate':DateTimeFormatter.format(DateTimeFormatter.STANDARD_DATE_FORMAT_PATTERN, result[i].birthDate)
            });
        }
        outputJSON(members);
    } catch (ex) {
        Debug.console('エラーが発生しました。', ex);
        throw ex;
    }
}

function outputText(text) {
    var response = Web.getHTTPResponse();
    response.setContentType('text/plain; charset=utf-8');
    response.sendMessageBodyString(text);
}
```

```
function outputJSON(object) {
    var response = Web.getHTTPResponse();
    response.setContentType('application/json; charset=utf-8');
    response.sendMessageBodyString(ImJson.toJSONString(object));
}
```

コラム

SOAPClient API を利用して Web サービスを呼び出す前に、認証・認可用のユーザ情報を設定します。
このサンプルでは、認証タイプ「WSSE」を利用しています。
認証タイプ「WSSE」の詳細は、「Web サービス 認証・認可 仕様書の認証・認可 」を参照してください。

サンプルでは WSAuthDigestGenerator4WSSE オブジェクトを利用してパスワード・ダイジェストを作成しています。
認証タイプ「WSSE」は、パスワードのダイジェスト化方法に、WS-Security の UsernameToken 形式を採用しています。
WSAuthDigestGenerator4WSSE オブジェクトは、そのパスワード・ダイジェストの生成に特化したユーティリティです。
「ユーザコード」と「パスワード」を元にパスワード・ダイジェストを生成します。

コラム

wsUserInfoに設定する情報については、Webサービスプロバイダ側の設定と合わせる必要があります。
サンプルでは「getWSUserInfo」メソッド内で、wsUserInfoの情報を定義しています。

コラム

SOAPClient API を使うためのサンプルコードは、SOAPClient オブジェクトの「getSampleCode()」から取得することができます。
以下のようなコードを実行することで、「add()」関数を呼び出すためのサンプルコードがコンソール上に出力されます。

```
// ホスト名、ポート番号、コンテキストパスは適宜置き換えてください。
var wsdlURL = 'http://localhost:8080/imart/services/SampleMemberInfoOperatorService?wsdl';
var soapClient = new SOAPClient(wsdlURL);
var sampleCode = soapClient.getSampleCode("add");
Debug.console(sampleCode);
```

SOAPClient オブジェクトの詳細は、「API リストの SOAPClient オブジェクト 」を参照してください。

コラム

Web サービスの結果が SOAPFault エラーをスローした場合、SOAPClient オブジェクトの関数を実行した際に
SOAPFault オブジェクトが例外としてスローされます。
SOAPFault オブジェクトは、XML 形式の SOAPFault を JavaScript のオブジェクト形式に変換したものです。

SOAPFault オブジェクトを捕捉するためには、Web サービスのオペレーション実行部分を try ～ catch 文で囲んでください。
catch 内で SOAPFault オブジェクトを利用したエラー処理を行うことができます。
SOAPFault オブジェクトの詳細は、「API リストの SOAPFault オブジェクト 」を参照してください。

なお、Webサービス・プロバイダ 側では、Web サービス呼び出し時に設定した認証情報（WSUserInfo）を元に認証・認可が行われます。
該当するユーザが存在しない、パスワードが間違っているなどの理由でユーザ情報が不正な場合、または、Web サービスを実行する権限がない場合には、SOAPFault オブジェクトが例外としてスローされます。
SOAPFault オブジェクトの「faultCode」プロパティには、発生した問題に対応するコードが含まれています。
コードについての詳細は、「Web サービス 認証・認可 仕様書の認証・認可の SOAP フォルトコード」を参照してください。

ルーティングテーブルを登録する

画面の作成が完了したら、画面にアクセスするためのルーティングテーブルを登録します。

sample_member_info.xml ファイルを、プロジェクトの以下の場所に作成します。

- src/main/conf/routing-jssp-config/sample_member_info.xml

sample_member_info.xml のソースは以下の通りです。

```
<?xml version="1.0" encoding="UTF-8"?>
<routing-jssp-config xmlns="http://www.intra-mart.jp/router/routing-jssp-config"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.intra-mart.jp/router/routing-jssp-config routing-jssp-config.xsd">

   <authz-default mapper="welcome-all" />

   <file-mapping path="/sample/web_service/client/member_info_operator"
      page="sample/web_service/client/member_info_operator">
   </file-mapping>

   <file-mapping path="/sample/web_service/client/member_info_operator/add"
      page="sample/web_service/client/member_info_operator" action="add">
   </file-mapping>

   <file-mapping path="/sample/web_service/client/member_info_operator/find"
      page="sample/web_service/client/member_info_operator" action="find">
   </file-mapping>

   <file-mapping path="/sample/web_service/client/member_info_operator/findAll"
      page="sample/web_service/client/member_info_operator" action="findAll">
   </file-mapping>

</routing-jssp-config>
```

Webサービス へアクセスする画面をデプロイする

資材をデプロイする

以上で Web サービスを実行するための資材が完成しました。
次に作成したモジュールをユーザモジュールとして取り込み、war を作成して Resin にデプロイします。

以下の手順に従って、e Builder でユーザモジュールを作成します。

1. プロジェクトを右クリックして、「エクスポート」を選択します。
2. 「e Builder」ー「imm file」を選択して、「次へ」をクリックします。
3. 出力先フォルダに任意の場所を選択して、「終了」をクリックします。
4. しばらくすると、出力先フォルダに sample_client-1.0.0.imm ファイルが作成されます。

次に、以下の手順に従って、e Builder で imm ファイルをユーザモジュールとして取り込みます。

1. e Builder で環境構築時に利用したプロジェクトの「juggling.im」を開きます。
2. 「ユーザモジュール」タブを開き、右上の「モジュールを追加します」アイコンをクリックします。
3. e Builder で作成した sample_client-1.0.0.imm ファイルを選択して開きます。

> **コラム**
>
> 依存関係が不足している場合、上側にエラーメッセージが表示されます。
> この場合、エラーメッセージをクリックして不足しているモジュールを追加してください。

4. juggling.im を保存して、環境構築時と同じ手順で war を作成し、Resin にデプロイします。
5. Resin を再起動します。

次に、以下の手順に従って、テナント環境セットアップを実施します。

1. システム管理画面を開き、システム管理者でログインします。
   http://<HOST>:<PORT>/<CONTEXT_PATH>/system/login
2. 「テナント環境セットアップ」をクリックします。

> **コラム**
>
> 「テナント環境は最新です。セットアップが必要なモジュールはありません。」が表示されている場合は、以降の操作は不要です。

3. 続けて「テナント環境セットアップ」をクリックします。
4. 確認メッセージで「決定」をクリックします。

Webサービス にアクセスする

Resin の再起動とテナント環境セットアップが完了したら、作成した画面にアクセスして Web サービスを実行します。

- 以下の手順に従って、メンバー情報を登録します。

  1. 以下の URL にアクセスします。
     http://<HOST>:<PORT>/<CONTEXT_PATH>/sample/web_service/client/member_info_operator
  2. 以下のような画面が表示されることを確認します。

  3. 以下の内容を入力して、「add」をクリックします。

id	test
name	テストユーザ
age	30
married	true
birthDate	1982-06-12

  4. 「OK」と表示されれば成功です。

- 以下の手順に従って、メンバー情報を検索します。

  1. 以下の内容を入力して、「find」をクリックします。

id	test
name	(空欄)
age	(空欄)
married	(空欄)

birthDate	（空欄）

2. 「OK」と表示され、以下の内容が表示されれば成功です。

id	test
name	テストユーザ
age	30
married	true
birthDate	1982-06-12

- 以下の手順に従って、メンバー情報の件数を表示します。

1. 「findAll」をクリックします。
2. 「1」と表示されれば成功です。

コラム

「add」で追加した分のメンバー情報の件数が表示されます。

項目

トラブルシューティング

Java スタブ・クラスのコンパイルに失敗する場合

SOAPClient オブジェクトを利用する際に以下の例外が発生する場合は、自動生成された Java スタブ・クラスのコンパイルに失敗しています。

```
[INFO] j.c.i.s.j.i.SOAPClientObject - Compile Stub(Java): /foo/bar/XXXXXStub.java
[ERROR] j.c.i.s.j.i.SOAPClientObject - null
java.lang.IllegalArgumentException
  at java.lang.ProcessImpl.<init>(ProcessImpl.java)
  at java.lang.ProcessImpl.start(ProcessImpl.java)
  at java.lang.ProcessBuilder.start(ProcessBuilder.java)
  at java.lang.Runtime.exec(Runtime.java)
  at jp.co.intra_mart.system.javascript.imapi.SOAPClientObject.compileStub(SOAPClientObject.java)
```

この現象が発生する場合は、以下の手順に従って、環境変数「AXIS2_HOME」を設定してください。

1. e Builder で環境構築時に利用したプロジェクトの「juggling.im」を開き、以下のモジュールが含まれていることを確認します。
   - 「ベースモジュール」タブ内の「ライブラリ」ー「サードパーティ製ライブラリ」ー「Apache Axis2」
2. war を展開してできたコンテキストパスと同名のフォルダを開き、以下のファイルが存在することを確認します。
   - `%CONTEXT_PATH%/WEB-INF/lib/axis2-xxx-x.x.x.jar` （x は任意）
3. intra-mart Accel Platform を実行する環境の環境変数「AXIS2_HOME」に、war を展開してできた `%CONTEXT_PATH%/WEB-INF` をパスとして設定します。

https で提供されている WSDL を利用する場合

以下のような、WSDL が https で提供されている場合の SOAPClient を利用するためには、接続先のサーバ証明書の取得・登録が必要です。

- `https://<HOST>/<CONTEXT_PATH>/services/SampleWebService?wsdl`

以下の手順に従って、サーバ証明書の取得・登録を行います。

1. 接続先のサーバ証明書を取得します。
   サーバ証明書の取得方法はいくつかありますが、ここでは Windows 環境の Internet Explorer 9 を利用して証明書を取得する方法を示します。
   1. Internet Explorer 9 を開き、WSDL の URL を入力してアクセスします。

2. Alt キーを押下してメニューバーを開き、「ツール」ー「インターネット オプション」を選択します。
3. 「コンテンツ」タブを開き、「証明書」をクリックします。
4. 取得したい証明書を選択して「エクスポート」をクリックします。
5. ウィザードを進めてサーバ証明書ファイルを保存します。

2. JDK に含まれる keytool を利用して、サーバ証明書をキーストアに追加します。
   例： サーバ証明書ファイルが C:\temp\server.crt に保存されており、別名「sample_alias」でキーストアエントリに追加する場合

```
keytool -import -alias sample_alias -file C:\temp\server.crt
```

> **コラム**
>
> 上記コマンドを実行すると、ユーザのホームディレクトリの「.keystore」ファイルに、キーストアが作成されます。
> keytool の詳細は、以下「JDK ドキュメントの keytool - 鍵と証明書の管理ツール」を参照してください。
>
> http://docs.oracle.com/javase/jp/7/technotes/tools/windows/keytool.html （日本語）
> http://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html （英語）

3. アプリケーションサーバの JavaVM のシステムプロパティに「javax.net.ssl.trustStore」を追加します。
   例： Resin のインストール先が C:\resin で、ユーザ名が user_name の場合
   C:\resin\conf\resin.properties

```
jvm_args : -Djavax.net.ssl.trustStore="C:\Users\user_name\.keystore"
```

> **コラム**
>
> すでに jvm_args が存在する場合は、末尾に半角空白で 1 文字空けて追記してください。

また、WSDL の URL が https で始まっていたとしても、WSDL に記述されているエンドポイントが https でない場合は、SOAPClient API を利用する際に、明示的にエンドポイントを指定してください。

```
// ホスト名、コンテキストパスは適宜置き換えてください。
var wsdlURL     = 'https://localhost/imart/services/SampleWebService?wsdl';
var serviceName = null;
var portName    = null;
var endPoint    = 'https://localhost/imart/services/SampleWebService'; // ← 明示的に指定します。

var soapClient = new SOAPClient(wsdlURL, serviceName, portName, endPoint);
```

「指定した要求に失敗しました」が発生する場合

本現象が発生した場合に考えられる原因は、以下の通りです。

- 指定した Web サービスを実行する権限がない可能性があります。

解決方法の詳細は、「Web サービス 認証・認可 仕様書の認証・認可の SOAP フォルトコード」の「wsse:RequestFailed」を参照してください。

「指定した RequestSecurityToken を理解できません」が発生する場合

本現象が発生した場合に考えられる原因は、以下の通りです。

- 認証タイプに対応する認証モジュールが存在しない可能性があります。

解決方法の詳細は、「Web サービス 認証・認可 仕様書の認証・認可の SOAP フォルトコード」の「wsse:BadRequest」を参照してください。

「要求が無効か、形式が間違っています」が発生する場合

本現象が発生した場合に考えられる原因は、以下の通りです。

- SOAP ボディにユーザ情報が存在していない可能性があります。
- ユーザ情報が格納されている要素名が「wsUserInfo」になっていない可能性があります。
- Web サービスとして公開する Java クラス（JavaScript ラッパークラス）のコンパイル方法が誤っている可能性があります。

WSDL の URL をブラウザで開き、Web サービスの関数定義内の引数名を確認してください。

■正

```
<xs:element name="add">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="wsUserInfo" type="ax22:WSUserInfo" nillable="true" minOccurs="0"/>
      <xs:element name="member" type="ax24:Member" nillable="true" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

■誤

```
<xs:element name="add">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="param0" type="ax22:WSUserInfo" nillable="true" minOccurs="0"/>
      <xs:element name="param1" type="ax24:Member" nillable="true" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

上記のように、引数名が「param0」「param1」のようになっている場合は、Java クラスのコンパイル方法が誤っています。

解決方法の詳細は、「Web サービス 認証・認可 仕様書の認証・認可の SOAP フォルトコード」の「wsse:InvalidRequest」を参照してください。

Storage 上の WSDL ファイルを利用する場合

Storage 上に保存されている WSDL ファイルを利用する場合は、その WSDL ファイルを指し示している PublicStorage オブジェクトを、SOAPClient のコンストラクタの第 1 引数に指定してください。
SOAPClient オブジェクトの詳細は、「API リストの SOAPClient オブジェクト」を参照してください。

なお、WSDL ファイルの解析時に必要なファイルの拡張子や、必要なファイルが格納されているディレクトリ名を設定することができます。
SOAPClient の設定についての詳細は、「設定ファイルリファレンスの SOAPClient オブジェクトの設定」を参照してください。

複数の Web サービスが定義されている WSDL を利用する場合

WSDL 内に複数の Web サービスが定義されている場合は、Web サービス名を明示的に指定して SOAPClient オブジェクトを利用する必要があります。
具体的には、SOAPClient オブジェクトのコンストラクタ第 2 引数に実行したい Web サービス名を指定します。

JavaScript 形式から Java 形式へのオブジェクト変換に失敗する場合

本現象が発生した場合に考えられる原因は、以下の通りです。

- JavaScript 形式 → Java 形式へのオブジェクト変換規則に違反している可能性があります。

例えば、下記のような例外が発生した場合は、「JavaScript の配列」を「配列として定義されていない JavaBean のプロパティ」に変換しようとした際に発生します。

```
IllegalConversionException: Cannot convert 'JavaScript NativeArray' into 'Java class ＜クラス名＞'
```

また、下記のような例外が発生した場合は、「JavaScript の “aaa” という文字列」を「String 以外の型（Number など）で宣言されている JavaBean のプロパティ」に変換しようとした際に発生します。

`NumberFormatException: For input string: "＜文字列＞"`

サンプルコード

バイナリファイルを送受信するサンプル

Web サービスとして公開する JavaScript ラッパークラスの関数の引数、および、返却値の型に「byte[]」を指定することで、バイナリファイルを送受信することができます。
引き渡されたバイト配列は自動的に Base64 にエンコードされ SOAP メッセージとして送受信されます。

バイナリファイルの送受信を行うサンプルのファイルは以下の通りです。
（ファイルの中身は後述します。）

- Webサービス・プロバイダ
  - `src/main/jssp/src/sample/web_service/provider/public_storage_access.js`
  - `src/main/java/sample/web_service/provider/PublicStorageAccessService.java`
  - `src/main/webapp/WEB-INF/services/sample_public_storage/META-INF/services.xml`
- Webサービス・クライアント
  - `src/main/jssp/src/sample/web_service/client/public_storage_access.html`
  - `src/main/jssp/src/sample/web_service/client/public_storage_access.js`
  - `src/main/conf/routing-jssp-config/sample_public_storage.xml`

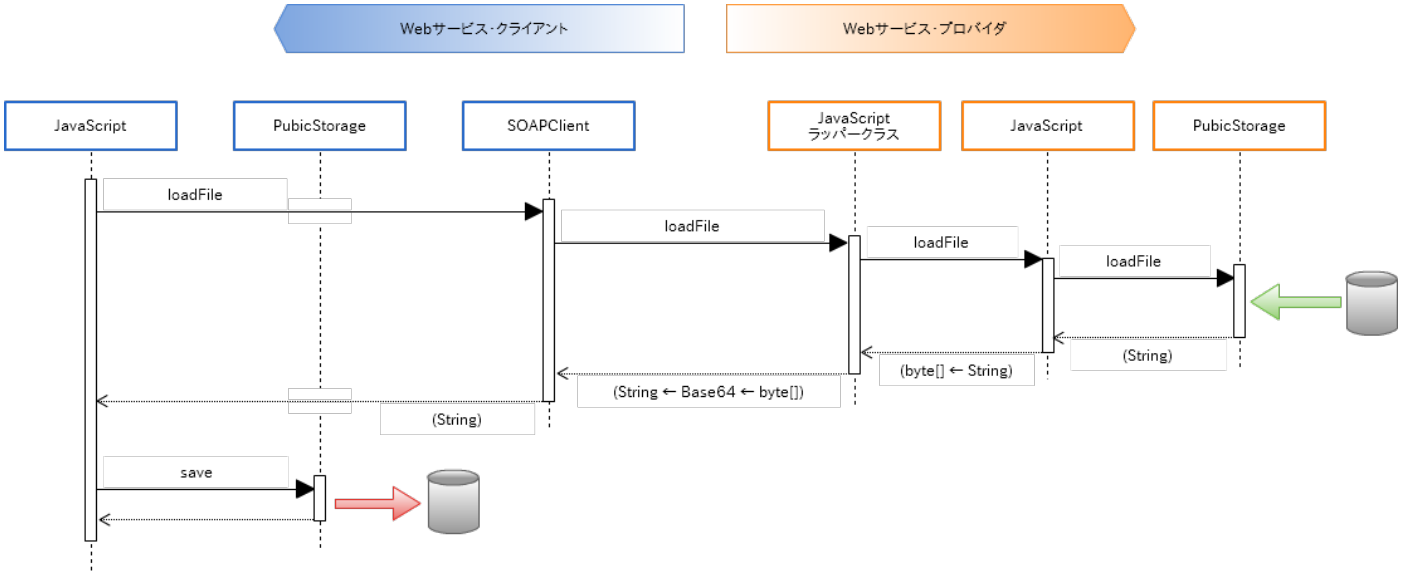
サンプルをプロジェクトに配置後、ユーザモジュールを作成し、warを作成し、デプロイを行います。
デプロイについての詳細は Webサービス・プロバイダ の「*資材をデプロイする*」、および、Webサービス・クライアント の「*資材をデプロイする*」を参照してください。
デプロイ後、以下の手順に従って、サンプル画面を表示します。

1. 以下の URL にアクセスします。
   `http://<HOST>:<PORT>/<CONTEXT_PATH>/sample/web_service/client/public_storage_access`
2. 以下のような画面が表示されることを確認します。

このサンプルプログラムは、Webサービス・クライアント 側と Webサービス・プロバイダ 側の PublicStorage 内にあるファイルの内容を送受信します。

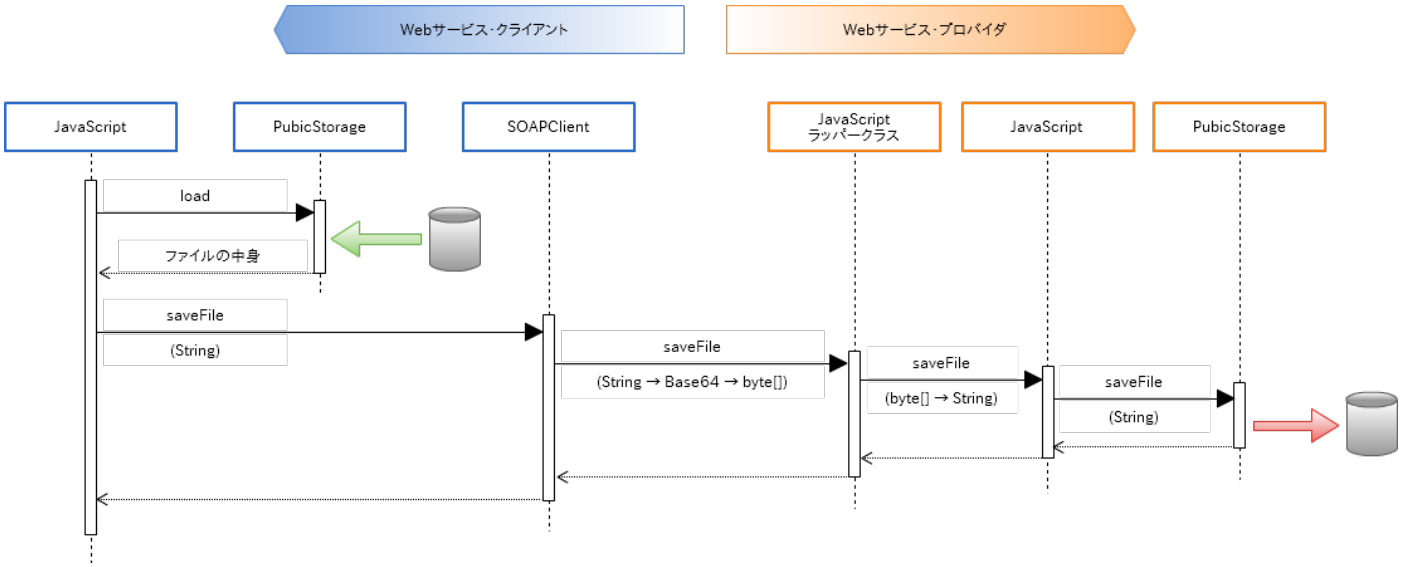
- バイナリファイルを受信する

  ファイルパスを入力して「A -> B コピー」をクリックすると、Web サービスを経由して Webサービス・プロバイダ 側の PublicStorage からファイルの中身を取得し、Webサービス・クライアント 側の PublicStorage に保存します。
  「A -> B コピー」をクリックした場合の処理の流れは、以下の通りです。



- バイナリファイルを送信する

  ファイルパスを入力して「B -> A コピー」をクリックすると、Webサービス・クライアント 側の PublicStorage からファイルを読み取り、Web サービスを経由して Webサービス・プロバイダ 側の PublicStorage にファイルを保存します。
  「B -> A コピー」をクリックした場合の処理の流れは、以下の通りです。



> **コラム**
>
> スクリプト開発モデルの場合、バイナリデータを送受信する際は String 型を使用します。

> **注意**
>
> Web サービスとして公開する関数の引数に JavaBean が指定されている場合、その JavaBean 内のバイト配列（byte[]）型のプロパティは、正常にデータが送受信されません。
> これは Apache Axis2 の現行仕様による制限です。
> バイナリファイルを送受信する場合は、JavaBean のプロパティではなく、Web サービスとして公開する関数の引数としてバイト配列（byte[]）を指定してください。

サンプルファイルの中身は、以下の通りです。

- src/main/jssp/src/sample/web_service/provider/public_storage_access.js

```
/**
 * ファイルの読み込み
 */
function loadFile(path) {
    var storage = new PublicStorage(path);
    return storage.load();
}

/**
 * ファイルの書き込み
 */
function saveFile(path, data) {
    var storage = new PublicStorage(path);
    return storage.save(data);
}
```

- src/main/java/sample/web_service/provider/PublicStorageAccessService.java

```
package sample.web_service.provider;

import jp.co.intra_mart.foundation.web_service.auth.WSUserInfo;
import jp.co.intra_mart.jssp.util.JavaScriptUtility;

import org.apache.axis2.AxisFault;

public class PublicStorageAccessService {
    public byte[] loadFile(final WSUserInfo wsUserInfo, final String path) throws AxisFault {
        try {
            final String pagePath = "sample/web_service/provider/public_storage_access";
            final String functionName = "loadFile";
            final byte[] bytes = (byte[]) JavaScriptUtility.executeFunction(pagePath, functionName, byte[].class, path);
            return bytes;
        } catch (final Exception ex) {
            throw AxisFault.makeFault(ex);
        }
    }

    public boolean saveFile(final WSUserInfo wsUserInfo, final String path, final byte[] data) throws AxisFault {
        try {
            final String pagePath = "sample/web_service/provider/public_storage_access";
            final String functionName = "saveFile";
            JavaScriptUtility.executeVoidFunction(pagePath, functionName, path, data);
            return true;
        } catch (final Exception ex) {
            throw AxisFault.makeFault(ex);
        }
    }
}
```

- src/main/webapp/WEB-INF/services/sample_public_storage/META-INF/services.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceGroup>
   <service name="SamplePublicStorageAccessService">
      <parameter name="ServiceClass">sample.web_service.provider.PublicStorageAccessService</parameter>
      <module ref="im_ws_auth"/>
      <messageReceivers>
         <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-only"
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver" />
         <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
      </messageReceivers>

      <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>

      <operation name="loadFile">
         <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>
      </operation>

      <operation name="saveFile">
         <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>
      </operation>

   </service>
</serviceGroup>
```

- src/main/jssp/src/sample/web_service/client/public_storage_access.html

```
<imart type="head">
<script type="text/javascript">
(function($) {
 $(function() {
  /**
   * ファイルの読み込み
   */
  $('#loadFile').click(function() {
   var userCd = $('#userCd').val();
   var password = $('#password').val();
   var pathA = $('#pathA').val();
   var pathB = $('#pathB').val();

   try {
    jQuery.ajax({
     async:false, cache:false, dataType:'text', type:'POST',
     url:'sample/web_service/client/public_storage_access/loadFile', data:{
      'userCd':userCd,
      'password':password,
      'pathA':pathA,
      'pathB':pathB
     },
     success:function(result) {
      imuiAlert(result == 'true' ? 'OK' : 'NG');
     },
     error:function() {
      imuiAlert('NG');
     }
    });
   } catch (ex) {
    imuiAlert('NG');
   }
  });

  /**
   * ファイルの書き込み
   */
  $('#saveFile').click(function() {
   var userCd = $('#userCd').val();
   var password = $('#password').val();
   var pathA = $('#pathA').val();
```

```
    var pathB = $('#pathB').val();

    try {
     jQuery.ajax({
      async:false, cache:false, dataType:'text', type:'POST',
      url:'sample/web_service/client/public_storage_access/saveFile', data:{
       'userCd':userCd,
       'password':password,
       'pathA':pathA,
       'pathB':pathB
      },
      success:function(result) {
       imuiAlert(result == 'true' ? 'OK' : 'NG');
      },
      error:function() {
       imuiAlert('NG');
      }
     });
    } catch (ex) {
     imuiAlert('NG');
    }
   });
 });
})(jQuery);
</script>
</imart>

<div class="imui-form-container-wide">
 <div class="imui-chapter-title">
  <h2>Web サービスにアクセスするユーザ情報</h2>
 </div>
 <table class="imui-form">
  <tbody>
   <tr>
    <th class="wd-20"><label class="imui-required">ユーザコード</label></th>
    <td><imart type="imuiTextbox" id="userCd" value="aoyagi" autofocus /></td>
   </tr>
   <tr>
    <th class="wd-20"><label class="imui-required">パスワード</label></th>
    <td><imart type="imuiTextbox" id="password" value="aoyagi" /></td>
   </tr>
  </tbody>
 </table>
 <div class="imui-chapter-title">
  <h2>PublicStorage のファイル</h2>
 </div>
 <table class="imui-form">
  <tbody>
   <tr>
    <th class="wd-20"><label class="imui-required">ファイルパスA (プロバイダ側)</label></th>
    <td><imart type="imuiTextbox" id="pathA" style="width: 400px;" /></td>
   </tr>
   <tr>
    <th class="wd-20"><label class="imui-required">ファイルパスB (クライアント側)</label></th>
    <td><imart type="imuiTextbox" id="pathB" style="width: 400px;" /></td>
   </tr>
  </tbody>
 </table>
 <div class="imui-operation-parts">
  <imart type="imuiButton" id="loadFile" value="A -> B にコピー" class="imui-large-button" />
  <imart type="imuiButton" id="saveFile" value="B -> A にコピー" class="imui-large-button" />
 </div>
</div>
```

- src/main/jssp/src/sample/web_service/client/public_storage_access.js

```
// ホスト名、ポート番号、コンテキストパスは適宜置き換えてください。
var wsdlURL = 'http://localhost:8080/imart/services/SamplePublicStorageAccessService?wsdl';
```

```
/**
 * 初期化
 */
function init() {
}

/**
 * 認証情報の取得
 */
function getWSUserInfo(userCd, password) {
    return {
        'userID':userCd,
        'password':WSAuthDigestGenerator4WSSE.getDigest(userCd, password),
        'authType':WSAuthDigestGenerator4WSSE.getAuthType(),
        'loginGroupID':'default'
    };
}

/**
 * ファイルの読み込み
 */
function loadFile(request) {
    try {
        var userCd = request.userCd;
        var password = request.password;
        var pathA = request.pathA;
        var pathB = request.pathB;

        var soapClient = new SOAPClient(wsdlURL);
        var wsUserInfo = getWSUserInfo(userCd, password);
        var result = soapClient.loadFile(wsUserInfo, pathA);
        Debug.console('読み取りに成功しました。', result);

        var storage = new PublicStorage(pathB);
        storage.save(result);
        Debug.console('書き込みに成功しました。');

        outputText('true');
    } catch (ex) {
        Debug.console('エラーが発生しました。', ex);
        throw ex;
    }
}

/**
 * ファイルの書き込み
 */
function saveFile(request) {
    try {
        var userCd = request.userCd;
        var password = request.password;
        var pathA = request.pathA;
        var pathB = request.pathB;

        var storage = new PublicStorage(pathB);
        var data = storage.load();
        Debug.console('読み取りに成功しました。', data);

        var soapClient = new SOAPClient(wsdlURL);
        var wsUserInfo = getWSUserInfo(userCd, password);
        var result = soapClient.saveFile(wsUserInfo, pathA, data);
        Debug.console('書き込みに成功しました。');
        outputText('true');
    } catch (ex) {
        Debug.console('エラーが発生しました。', ex);
        throw ex;
    }
}

function outputText(text) {
```

```
    var response = Web.getHTTPResponse();
    response.setContentType('text/plain; charset=utf-8');
    response.sendMessageBodyString(text);
}
```

- src/main/conf/routing-jssp-config/sample_public_storage.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<routing-jssp-config xmlns="http://www.intra-mart.jp/router/routing-jssp-config"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.intra-mart.jp/router/routing-jssp-config routing-jssp-config.xsd">

 <authz-default mapper="welcome-all" />

 <file-mapping path="/sample/web_service/client/public_storage_access"
  page="sample/web_service/client/public_storage_access">
 </file-mapping>

 <file-mapping path="/sample/web_service/client/public_storage_access/loadFile"
  page="sample/web_service/client/public_storage_access" action="loadFile">
 </file-mapping>

 <file-mapping path="/sample/web_service/client/public_storage_access/saveFile"
  page="sample/web_service/client/public_storage_access" action="saveFile">
 </file-mapping>

</routing-jssp-config>
```