intra-mart マスカット連携ガイド

Version 6.1

第三版

2008 年 1 月 31 日

＜＜　変更履歴　＞＞

変更年月日	変更内容
2007/7/31	**初版**
2007/8/31	第二版 以下の説明を追加 「**3.1.2.2** 初期表示時のアクション」 「**3.2.2.2** 初期表示時のアクション」
2008/1/31	第三版 **「3.3 デバック」を追加**

<< 目次 >>

1 はじめに 3
1.1 目的 3
1.2 マスカット パッケージ 3
1.3 動作条件 3
2 セットアップ 4
2.1 開発環境のセットアップ 4
3 アプリケーションの開発 5
3.1 im JavaEE Frameworkを利用した開発 5
3.1.1 サーブレットの定義 5
3.1.2 処理内容の決定 6
3.1.3 サービスフレームワークの実装 8
3.1.4 サンプルアプリケーション 12
3.2 サーバサイドJavaScriptを利用した開発 15
3.2.1 サーブレットの定義 15
3.2.2 処理内容の決定 16
3.2.3 サーバサイドJavaScriptの実装 18
3.2.4 サンプルアプリケーション 19
3.3 デバック 21

Copyright 2003-2005 株式会社 NTT データ イントラマート All rights Reserved.

1 はじめに

1.1 目的

マスカットは Ajax ベースのリッチクライアントを開発するためのオープンソース･フレームワークである。ここでは intra-mart WebPlatform/AppFramework とマスカットの連携方法について述べる。

1.2 マスカット パッケージ

intra-mart WebPlatform/AppFramework 6.1 にはマスカット パッケージが組み込まれている。以下が組み込まれているパッケージである。

- maskat-1.4.5

マスカット パッケージについてはマスカットProjectのWebサイトに詳しい情報が記載されている。

http://maskat.sourceforge.jp/

1.3 動作条件

- intra-mart WebPlatform/AppFramework インストール時に指定する「サーバーモジュールの文字コード」、「ウェブブラウザに送信する文字コード」は UTF-8 とする。
- アプリケーションの文字コードはすべて UTF-8 とする。
- クライアントのセッション管理を行うため、ブラウザの Cookie を有効にする。

2　セットアップ

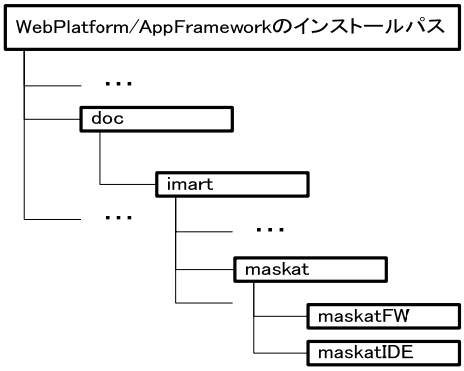
2.1　開発環境のセットアップ

intra-mart WebPlatform/AppFramework 6.1（以下 intra-mart）にはマスカットを利用して開発を行うための IDE（統合開発環境）は含まれていない。そのため以下のマスカット Project の Web サイトからダウンロードする必要がある。

http://maskat.sourceforge.jp/index.php?Download

◆ マスカット IDE(ブラウザ版)

ウェブサイトから maskatIDE-Browser-1.4.5v20070629.zip をダウンロードし、アーカイブを解凍する。解凍された「maskatIDE」ディレクトリを intra-mart WebPlatform/AppFramework 内の以下のパスに配置する。



◆ マスカット IDE(Eclipse 版)

Eclipse 版は Eclipse3.2 以上が必要である。そのため別途 Eclipse3.2 をセットアップしてインストール必要がある。インストール方法についてはマスカット Project の Web サイトに記載されている。

Copyright 2003-2005 株式会社 NTT データ イントラマート All rights Reserved.

3 アプリケーションの開発

この章では intra-mart において実際にアプリケーションを開発する方法を説明する。マスカットパッケージは独立したクライアントサイドのフレームワークであるため、サーバサイドの実装に依存しない。そのため複数の開発言語に対応している。
intra-mart においてマスカットを利用したアプリケーションを開発する場合、以下の二種類の開発言語が利用可能である。

- im JavaEE Framework
- サーバサイド JavaScript

3.1 im JavaEE Framework を利用した開発

intra-mart には標準で JavaEE に対応したフレームワークである、im JavaEE Framework が含まれている。このフレームワークを利用した開発方法を説明する。im JavaEE Framework に関する詳しい説明は「im JavaEE Framework 仕様書」に記載されている。

3.1.1 サーブレットの定義

intra-martにはマスカットからリクエストされた電文を解析し、アプリケーションが生成した電文をレスポンスに設定するために「MKServiceServlet」が定義されている。「リスト 3-1 MKServiceServletの設定」がweb.xmlに設定されているMKServiceServletである。

```
  <servlet>
    <servlet-name>MKServiceServlet</servlet-name>
    <servlet-class>
      jp.co.intra_mart.extension.maskat.servlet.MKServiceServlet
    </servlet-class>
  </servlet>
・・・
  <servlet-mapping>
    <servlet-name>MKServiceServlet</servlet-name>
    <url-pattern>/MKServiceServlet</url-pattern>
  </servlet-mapping>
```

リスト 3-1 MKServiceServlet の設定

im JavaEE Framework を利用してサーバサイドの実装を行う場合、マスカットは MKServiceServlet に電文を送信する必要がある。

3.1.2 処理内容の決定

マスカットは処理内容を決定するために以下の ID をレスポンスヘッダに付加し、サーバに電文を送信する。

- ◆ レイアウト ID
- ◆ コンポーネント ID
- ◆ イベント ID

ここでは MKServiceServlet が送信された ID によってどのような処理を行うかを説明する。

3.1.2.1 コンポーネントによるアクション

MKServiceServlet は受信した ID から以下の条件でサービスフレームワークを実行する。

- アプリケーション ID
  - レイアウト ID と同じ識別子とする。
- サービス ID
  - コンポーネント ID とイベント ID を「-（ハイフン）」で繋いだものを識別子とする。

例として、レイアウト ID が「**myLayout**」、コンポーネント ID が「**myComponent**」、イベント ID が「**onclick**」の場合 MKServiceServlet は以下の条件でサービスフレームワークを実行する。

- アプリケーション ID
  「**myLayout**」
- サービス ID
  「**myComponent-onclick**」

実際にサービスコンフィグファイルに記述する場合「リスト 3-2 サービスコンフィグファイルの例」のようになる。この例でのファイル名は「**service-config-myLayout.xml**」となる。

```
<?xml version="1.0" encoding="UTF-8"?>
<service-config>
    <service>
        <service-id>myComponent-onclick</service-id>
        <controller-class>・・・</controller-class>
         ・・・
    </service>
</service-config>
```

リスト 3-2 サービスコンフィグファイルの例

Copyright 2003-2005 株式会社 NTT データ イントラマート All rights Reserved.

3.1.2.2 初期表示時のアクション

マスカットは初期表示時にサーバへリクエストを送信することができる。「リスト 3-3 初期表示時のイベント定義XML」は初期表示時にサーバへリクエストを送信する場合のイベント定義XMLの例である。

```
<eventDef>
  <header name="maskat_layoutID" value="myLayout"/>
  <event id="onload" type="remote" async="false" remoteUrl="../../../MKServiceServlet">
  ・・・
  </event>
</eventDef>
```

リスト 3-3 初期表示時のイベント定義 XML

この場合、レイアウトIDが「**myLayout**」、イベントIDが「**onload**」となるがコンポーネントIDは送信されない。MKServiceServletはコンポーネントIDが送信されない場合、初期表示時とみなし、コンポーネントIDを「**onload**」とする。そのため「リスト 3-4 初期表示時のサービスコンフィグファイル」のようにサービスコンフィグファイルを定義する必要がある。

```
<?xml version="1.0" encoding="UTF-8"?>
<service-config>
    <service>
        <service-id>onload-onload</service-id>
        <controller-class>・・・</controller-class>
        ・・・
    </service>
</service-config>
```

リスト 3-4 初期表示時のサービスコンフィグファイル

3.1.3 サービスフレームワークの実装

マスカットから送信された電文を解析し、処理を実行した後マスカットに返却する電文を作成するためのサービスフレームワーク実装する。

3.1.3.1 コントローラオブジェクトの作成

マスカットからの電文を解析し、必要な情報を格納するコトローラオブジェクトを作成する。

マスカットから「リスト 3-5 受信電文の例」のような電文を受信した場合、コントローラオブジェクトは「リスト 3-6 コントローラオブジェクトの例」のようになる。

```
<?xml version="1.0" encoding="UTF-8"?>
<sampleParam>
  <arg1>value1</arg1>
  <arg2>value2</arg2>
</sampleParam>
```

リスト 3-5 受信電文の例1

```
Package sample.service.controller;

public class SampleControllerObject extends MKControllerObject {

  public static final String PATH_arg1 = "/sampleParam/arg1";
  public static final String PATH_arg2 = "/sampleParam/arg2";

  private String arg1 = null;
  private String arg2 = null;

  public String getArg1() {
    return arg1;
  }

  public void setArg1(String arg1) {
    this.arg1 = arg1;
  }

  public String getArg2() {
    return arg2;
  }

  public void setArg2(String arg2) {
    this.arg2 = arg2;
  }
}
```

リスト 3-6 コントローラオブジェクトの例 1

「PATH_フィールド名」となる名前の静的フィールドを定義し、代入されるノードパスを設定する。設定されたノードパスの値はコントローラコンバータが電文を解析し、コントローラオブジェクトに値を設定する。

指定したノードパスに複数のノードが存在する場合、フィールドの型を配列にする必要がある。「リスト 3-7 受信電文の例 2」および「リスト 3-8 コントローラオブジェクトの例 2」はその例である。

Copyright 2003-2005 株式会社 NTT データ イントラマート All rights Reserved.

```
<?xml version="1.0" encoding="UTF-8"?>
<sampleParam>
  <arg>value1</arg>
  <arg>value2</arg>
  <arg>value3</arg>
</sampleParam>
```

リスト 3-7 受信電文の例 2

```
Package sample.service.controller;

public class SampleControllerObject extends MKControllerObject {

  public static final String PATH_arg = "/sampleParam/arg";

  private String[] arg = null;

  public String[] getArg() {
    return arg;
  }

  public void setArg(String[] arg) {
    this.arg = arg;
  }
}
```

リスト 3-8 コントローラオブジェクトの例 2

サービスコンフィグファイルは「リスト 3-9 コントローラオブジェクトの設定」のように設定する。

```
<?xml version="1.0" encoding="UTF-8"?>
<service-config>
  <service>
    ・・・
    <controller-converter>
      <converter-class>
        jp.co.intra_mart.extension.maskat.service.controller.NodePathControllerConverter
      </converter-class>
      <init-param>
        <param-name>object</param-name>
        <param-value>sample.service.controller.SampleControllerObject</param-value>
      </init-param>
    </controller-converter>
    ・・・
  </service>
</service-config>
```

リスト 3-9 コントローラオブジェクトの設定

マスカットからの電文をコントローラオブジェクトに割り当てるためには以下の条件を満たす必要がある。

- コントローラコンバータは NodePathControllerConverter を使用する。
- コントローラオブジェクトは MKControllerObject を継承している。
- コントローラオブジェクトにはデフォルトコンストラクタが存在する。
- 設定対象のフィールドは String 型、または String 型の配列である。
- 設定対象のフィールドには setter、getter が存在する。

3.1.3.2 サービスコントローラの作成

コントローラオブジェクトを受け取り、実際の処理を行うサービスコントローラを実装する。

「リスト 3-10 サービスコントローラの例」はサービスコントローラの作成例である。

```
package sample.service;

import jp.co.intra_mart.extension.maskat.service.DefaultMKServiceResult;
import jp.co.intra_mart.framework.base.service.ServiceControllerAdapter;
import jp.co.intra_mart.framework.base.service.ServiceResult;
import jp.co.intra_mart.framework.system.exception.ApplicationException;
import jp.co.intra_mart.framework.system.exception.SystemException;

public class SampleServiceController extends ServiceControllerAdapter {

  public ServiceResult service() throws SystemException, ApplicationException {

    // 受信電文を取得
    SampleControllerObjectobj = (SampleControllerObject) getControllerObject();

    // 送信電文の作成
    String xmlString = "<sample>・・・</sample>";

    return new DefaultMKServiceResult(xmlString);
  }
}
```

リスト 3-10 サービスコントローラの例

マスカットへ電文を送信ためには service メソッドの戻り値が MKServiceResult インタフェースの実装クラスである必要がある。この例では MKServiceResult の実装クラスである DefaultMKServiceResult を戻り値としている。MKServiceResult の実装クラスを独自に作成する場合は MKServiceResult#getTelegram()メソッドを実装し、マスカットへ送信する電文を返却しなければならない。

intra-mart には標準で以下の MKServiceResult インタフェースの実装クラスが含まれている。

- DefaultMKServiceResult
  org.w3c.dom.Document または XML 文字列から電文を生成する。
- MKErrorsResult
  マスカットへエラーを通知する電文を生成する。

サービスコンフィグファイルは「リスト 3-11 サービスコントローラの設定」のように設定する。

```
<?xml version="1.0" encoding="UTF-8"?>
<service-config>
  <service>
    ・・・
    <controller-class>sample.service.SampleServiceController</controller-class>
    ・・・
  </service>
</service-config>
```

リスト 3-11 サービスコントローラの設定

Copyright 2003-2005 株式会社 NTT データ イントラマート All rights Reserved.

3.1.3.3 トランジションの設定

サービスコントローラで生成された電文をマスカットに送信するトランジションを設定する。

intra-mart には標準で MKTransition が含まれている。

サービスコンフィグファイルは「リスト 3-12 トランジションの設定」のように設定する。

```
<?xml version="1.0" encoding="UTF-8"?>
<service-config>
  <service>
    ・・・
    <transition-class>
      jp.co.intra_mart.extension.maskat.service.MKTransition
    </transition-class>
    ・・・
  </service>
</service-config>
```

リスト 3-12 トランジションの設定

MKTransition はサービスコントローラから返された MKServiceResult の getTelegram()からドキュメントを取得し、レスポンスとして電文をマスカットに返却する。MKTransition を使用する場合は以下の条件を満たす必要がある。

- ServiceController#service()メソッドの戻り値は MKServiceResult インタフェースの実装クラスである。
- MKServiceResult#getTelegram()はマスカットに送信するためのドキュメントを返却する。

3.1.4 サンプルアプリケーション

ここでは足し算を行うサンプルアプリケーションを作成する。

作成するファイルは以下の物となる。

- doc/imart/maskat/contents/demo_imJavaEE/add.html
- doc/imart/maskat/contents/demo_imJavaEE/add.xml
- doc/imart/maskat/contents/demo_imJavaEE/add_e.xml
- doc/imart/WEB-INF/classes/service-config-add.xml
- doc/imart/WEB-INF/classes/sample/AddControllerObject.java
- doc/imart/WEB-INF/classes/sample/AddServiceController.java

intra-mart メニューに登録するパスは「doc/imart/maskat/contents/demo_imJavaEE/add.html」である。

doc/imart/maskat/contents/demo_imJavaEE/add.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>足し算プログラム</title>
<link rel='stylesheet' type='text/css'
  href='../../maskatFW/rialtoEngine/style/rialto.css'/>
<link rel='stylesheet' type='text/css'
  href='../../maskatFW/rialtoEngine/style/behavior.css'>
<link rel='stylesheet' type='text/css'
  href='../../maskatFW/rialtoEngine/style/defaultSkin.css'/>
<script type="text/javascript"
  src="../../maskatFW/core/maskatConfig.js"></script>
<script type="text/javascript">
rialtoConfig.pathRialtoE = "../../maskatFW/rialtoEngine/";
</script>
<script type='text/javascript'
  src='../../maskatFW/rialtoEngine/javascript/rialto.js'></script>
<script type="text/javascript"
  src="../../maskatFW/core/maskat.js"></script>
<script type="text/javascript">
function pageLoad(){
 maskat.loadLayoutFile('add.xml', 'document.getElementById("divConteiner")');
 maskat.loadEventFile('add_e.xml');
}
</script>
</head>
<body onload="pageLoad();" onselectstart="return true">
<form>
<div id="divConteiner"
  style="position:absolute;
         left:0px; top:0px;
         width:500px;
         height:200px;
         border:1px solid black;"></div>
</form>
</body>
</html>
```


Copyright 2003-2005 株式会社 NTT データ イントラマート All rights Reserved.

doc/imart/maskat/contents/demo_imJavaEE/add.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE layoutDef SYSTEM "layoutDef.dtd">
<layoutDef>
   <layout name="myLayout" refParentHTML="document.getElementById('divConteiner')">
      <label name="title" top="10" left="10" text="足し算プログラム"></label>
      <text name="remote_arg1" top="70" left="10" width="100"></text>
      <label name="remote_plus" top="70" left="120" text="+"></label>
      <text name="remote_arg2" top="70" left="140" width="100"></text>
      <button name="remote_equal" top="70" left="260" title="="></button>
      <text name="remote_ans" top="70" left="370" width="100"></text>
   </layout>
</layoutDef>
```

doc/imart/maskat/contents/demo_imJavaEE/add_e.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventDef SYSTEM "eventDef.dtd">
<eventDef>
  <header name="maskat_layoutID" value="add"/>
  <component id="remote_equal">
    <event id="onclick" type="remote" async="false" remoteUrl="../../../MKServiceServlet">
      <param rootNode="addParam">
        <source obj="remote_arg1" node="arg1" desc="パラメータ 1"/>
        <source obj="remote_arg2" node="arg2" desc="パラメータ 2"/>
      </param>
      <result rootNode="sample">
        <target out="remote_ans" in="result"/>
      </result>
    </event>
  </component>
</eventDef>
```

doc/imart/WEB-INF/classes/service-config-add.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<service-config>
  <service>
    <service-id>remote_equal-onclick</service-id>
    <controller-class>sample.AddServiceController</controller-class>
    <controller-converter>
      <converter-class>
        jp.co.intra_mart.extension.maskat.service.controller.NodePathControllerConverter
      </converter-class>
      <init-param>
        <param-name>object</param-name>
        <param-value>sample.AddControllerObject</param-value>
      </init-param>
    </controller-converter>
    <transition-class>
      jp.co.intra_mart.extension.maskat.service.MKTransition
    </transition-class>
  </service>
</service-config>
```

doc/imart/WEB-INF/classes/sample/AddControllerObject.java

```
package sample;

public class AddControllerObject extends MKControllerObject {

    public static final String PATH_arg1 = "/addParam/arg1";
    public static final String PATH_arg2 = "/addParam/arg2";

    private String arg1 = null;
    private String arg2 = null;

    public String getArg1() {
        return arg1;
    }
    public void setArg1(String arg1) {
        this.arg1 = arg1;
    }
    public String getArg2() {
        return arg2;
    }
    public void setArg2(String arg2) {
        this.arg2 = arg2;
    }
}
```

doc/imart/WEB-INF/classes/sample/AddServiceController.java

```
package sample;

public class AddServiceController extends ServiceControllerAdapter {

    public ServiceResult service() throws SystemException, ApplicationException {

        String result = null;
        try {
            // 受信電文を取得
            AddControllerObject obj = (AddControllerObject) getControllerObject();

            // 送信電文を生成
            int arg1 = Integer.parseInt(obj.getArg1());
            int arg2 = Integer.parseInt(obj.getArg2());
            int ans = arg1 + arg2;
            result = "<sample><result>" + ans + "</result></sample>";
        } catch (NumberFormatException e) {
            MKError error = new MKError();
            error.setMessage("数値を指定してください");
            return new MKErrorsResult(error);
        } catch (Exception e) {
            throw new SystemException(e);
        }
        return new DefaultMKServiceResult(result);
    }
}
```


Copyright 2003-2005 株式会社 NTT データ イントラマート All rights Reserved.

3.2 サーバサイド JavaScript を利用した開発

intra-mart に搭載されている Mozilla Rhino を利用してサーバサイド JavaScript を実行することが可能です。
ここではその具体的な手順を示します。

3.2.1 サーブレットの定義

intra-martにはマスカットからリクエストされた電文を解析し、アプリケーションが生成した電文をレスポンスに設定するために「MKJSServlet」が定義されている。「リスト 3-13 MKJSServletの定義」がweb.xmlに設定されているMKJSServletである。

```
  <servlet>
    <servlet-name>MKJSServlet</servlet-name>
    <servlet-class>
      jp.co.intra_mart.extension.maskat.servlet. MKJSServlet
    </servlet-class>
    <init-param>
      <param-name>srcDir</param-name>
      <param-value>/maskat</param-value>
    </init-param>
  </servlet>
・・・
  <servlet-mapping>
    <servlet-name>MKJSServlet</servlet-name>
    <url-pattern>/MKJSServlet</url-pattern>
  </servlet-mapping>
```

リスト 3-13 MKJSServlet の定義

サーバサイド JavaScript を利用してサーバサイドの実装を行う場合、マスカットは MKJSServlet に電文を送信する必要がある。
初期化パラメータ「srcDir」は js ファイルを配置するルートディレクトリである。デフォルトは「maskat」となっており、この場合以下のディレクトリがルートディレクトリとなる。

- pages/platform/src/maskat
- pages/product/src/maskat
- pages/src/maskat

3.2.2 処理内容の決定

マスカットは処理内容を決定するために以下の ID をレスポンスヘッダに付加し、サーバに電文を送信する。

- ◆ レイアウト ID
- ◆ コンポーネント ID
- ◆ イベント ID

ここでは MKJSServlet が送信された ID によってどのような処理を行うかを説明する。

3.2.2.1 コンポーネントによるアクション

MKJSServlet は受信した ID から以下の条件でサーバサイド JavaScript を実行する。

ここでは MKJSServlet の初期化パラメータ「srcDir」がデフォルトの「maskat」であることを前提に説明する。

- js ファイルパス
  - maskat/レイアウト ID/コンポーネント ID.js
- 実行関数名
  - イベント ID と等しい関数名

例として、レイアウト ID が「**myLayout**」、コンポーネント ID が「**myComponent**」、イベント ID が「**onclick**」の場合 MKServiceServlet は以下の条件でサーバサイド JavaScript を実行する。

- js ファイルパス
  「**maskat/myLayout/myComponent.js**」
- 実行関数名
  「**onclick**」

実際にjs ファイル作製する場合「」のようになる。

上記の通りこの例でのファイル名は「**maskat/myLayout/myComponent.js**」となる。

```
function onclick(xmlString) {

    var obj = new XML(xmlString);
 ・・・
    return "<sample>....</sample>";
}
```


Copyright 2003-2005 株式会社 NTT データ イントラマート All rights Reserved.

3.2.2.2 初期表示時のアクション

マスカットは初期表示時にサーバへリクエストを送信することができる。「リスト 3-14 初期表示時のイベント定義XML」は初期表示時にサーバへリクエストを送信する場合のイベント定義XMLの例である。

```
<eventDef>
  <header name="maskat_layoutID" value="myLayout"/>
  <event id="onload" type="remote" async="false" remoteUrl="../../../MKJSServlet">
  ・・・
  </event>
</eventDef>
```

リスト 3-14 初期表示時のイベント定義 XML

この場合、レイアウト ID が「**myLayout**」、イベント ID が「**onload**」となるがコンポーネント ID は送信されない。MKJSServlet はコンポーネント ID が送信されない場合、初期表示時とみなし、コンポーネント ID を「**onload**」とする。そのため MKServiceServlet は以下の条件でサーバサイド JavaScript を実行する。

- js ファイルパス
  「**maskat/myLayout/onload.js**」
- 実行関数名
  「**onload**」

3.2.3　サーバサイド JavaScript の実装

マスカットから送信された電文を解析し、処理結果を送信するサーバサイド JavaScript を実装する。

3.2.3.1　関数の実装

以下は js ファイルの例である。

```
function onclick(xmlString) {

    var obj = new XML(xmlString);
    var ans = parseInt(obj.arg1) + parseInt(obj.arg2);
    return "<sample><result>" + ans + "</result></sample>";
}
```

関数のパラメータにはマスカットから送信された電文が String 型で渡される。この電文を XML パーサを利用して、解析する必要がある。
この例では、E4X(ECMAScript for XML)を利用して受信電文を解析している。
関数の戻り値にはマスカットへ送信する XML 文字列を与える必要がある。

3.2.3.2　エラーの送信

マスカットは通常以下の形式の電文を送信することでエラー処理を実行する。

```
<?xml version="1.0" encoding="UTF-8"?>
<errors>
  <error>
    <errorCode>・・・</errorCode>
    <messageCode>・・・</messageCode>
    <message>・・・</message>
    <info>・・・</info>
    <systemErrorMessage>・・・</systemErrorMessage>
  </error>
</errors>
```

サーバサイド JavaScript からエラー電文を送信する場合「MKError」、「MKErrors」を使用してエラー電文を送信することが可能である。以下はその例である。

```
function onclick(xmlString) {

  var obj = new XML(xmlString);
  if (obj.foo != bar) {
    var error = new MKError();
    error.setErrorCode("・・・");

    var errors = new MKErrors();
    errors.addError(error);
    throw errors;
  }
   ・・・
}
```

MKErrors のインスタンスを throw することで自動的にエラー電文を作成し、マスカットに送信される。

Copyright 2003-2005 株式会社 NTT データ イントラマート All rights Reserved.

3.2.4 サンプルアプリケーション

ここでは足し算を行うサンプルアプリケーションを作成する。

作成するファイルは以下の物となる。

- doc/imart/maskat/contents/demo_imJS/add.html
- doc/imart/maskat/contents/demo_imJS/add.xml
- doc/imart/maskat/contents/demo_imJS/add_e.xml
- pages/src/maskat/add/remote_equal.js

intra-mart メニューに登録するパスは「doc/imart/maskat/contents/demo_imJS/add.html」である。

doc/imart/maskat/contents/demo_imJS/add.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>足し算プログラム</title>
<link rel='stylesheet' type='text/css'
  href='../../maskatFW/rialtoEngine/style/rialto.css'/>
<link rel='stylesheet' type='text/css'
  href='../../maskatFW/rialtoEngine/style/behavior.css'>
<link rel='stylesheet' type='text/css'
  href='../../maskatFW/rialtoEngine/style/defaultSkin.css'/>
<script type="text/javascript"
  src="../../maskatFW/core/maskatConfig.js"></script>
<script type="text/javascript">
rialtoConfig.pathRialtoE = "../../maskatFW/rialtoEngine/";
</script>
<script type='text/javascript'
  src='../../maskatFW/rialtoEngine/javascript/rialto.js'></script>
<script type="text/javascript"
  src="../../maskatFW/core/maskat.js"></script>
<script type="text/javascript">
function pageLoad(){
 maskat.loadLayoutFile('add.xml', 'document.getElementById("divConteiner")');
 maskat.loadEventFile('add_e.xml');
}
</script>
</head>
<body onload="pageLoad();" onselectstart="return true">
<form>
<div id="divConteiner"
  style="position:absolute;
         left:0px; top:0px;
         width:500px;
         height:200px;
         border:1px solid black;"></div>
</form>
</body>
</html>
```

doc/imart/maskat/contents/demo_imJS/add.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE layoutDef SYSTEM "layoutDef.dtd">
<layoutDef>
   <layout name="myLayout" refParentHTML="document.getElementById('divConteiner')">
      <label name="title" top="10" left="10" text="足し算プログラム"></label>
      <text name="remote_arg1" top="70" left="10" width="100"></text>
      <label name="remote_plus" top="70" left="120" text="+"></label>
      <text name="remote_arg2" top="70" left="140" width="100"></text>
      <button name="remote_equal" top="70" left="260" title="="></button>
      <text name="remote_ans" top="70" left="370" width="100"></text>
   </layout>
</layoutDef>
```

doc/imart/maskat/contents/demo_imJS/add_e.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventDef SYSTEM "eventDef.dtd">
<eventDef>
  <header name="maskat_layoutID" value="add"/>
  <component id="remote_equal">
    <event id="onclick" type="remote" async="false" remoteUrl="../../../MKJSServlet">
      <param rootNode="addParam">
        <source obj="remote_arg1" node="arg1" desc="パラメータ 1"/>
        <source obj="remote_arg2" node="arg2" desc="パラメータ 2"/>
      </param>
      <result rootNode="sample">
        <target out="remote_ans" in="result"/>
      </result>
    </event>
  </component>
</eventDef>
```

pages/src/maskat/add/remote_equal.js

```
function onclick(xmlString) {

    var obj = new XML(xmlString);

    if (isNaN(obj.arg1) || isNaN(obj.arg2)) {
        var error = new MKError();
        error.setMessage("数値を指定してください");

        var errors = new MKErrors();
        errors.addError(error);
        throw errors;
    }

    var ans = parseInt(obj.arg1) + parseInt(obj.arg2);
    return "<sample><result>" + ans + "</result></sample>";
}
```


Copyright 2003-2005 株式会社 NTT データ イントラマート All rights Reserved.

3.3 デバック

この章ではマスカットを利用して開発する上でのデバック方法を説明する。

マスカットにはイベントやエラーなどの情報を表示するための log4javascript が含まれている。
以下が log4javascript を使用する場合の記述例である。log4javascript.js は maskat.js より先に読み込む必要があるため、maskat.js より上に記述しなければならない。

doc/imart/maskat/contents/demo_imJS/add.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>足し算プログラム</title>
<link rel='stylesheet' type='text/css'
  href='../../maskatFW/rialtoEngine/style/rialto.css'/>
<link rel='stylesheet' type='text/css'
  href='../../maskatFW/rialtoEngine/style/behavior.css'>
<link rel='stylesheet' type='text/css'
  href='../../maskatFW/rialtoEngine/style/defaultSkin.css'/>
<script type="text/javascript"
  src="../../maskatFW/core/maskatConfig.js"></script>
<script type="text/javascript">
rialtoConfig.pathRialtoE = "../../maskatFW/rialtoEngine/";
</script>
<script type='text/javascript'
  src='../../maskatFW/rialtoEngine/javascript/rialto.js'></script>
<script type="text/javascript"
  src="../../maskatFW/log4javascript/log4javascript.js"></script>
<script type="text/javascript"
  src="../../maskatFW/core/maskat.js"></script>
<script type="text/javascript">
function pageLoad(){
 maskat.loadLayoutFile('add.xml', 'document.getElementById("divConteiner")');
 maskat.loadEventFile('add_e.xml');
}
</script>
</head>
<body onload="pageLoad();" onselectstart="return true">
<form>
<div id="divConteiner"
  style="position:absolute;
         left:0px; top:0px;
         width:500px;
         height:200px;
         border:1px solid black;"></div>
</form>
</body>
</html>
```

intra-mart マスカット連携ガイド

第三版　：January 31, 2008

Copyright 2000-2007 (株)NTT データイントラマート
All rights Reserved.

TEL: 03-5549-2821
FAX: 03-5549-2816
URL: http://www.intra-mart.jp/