IM-Curl 連携 API

チュートリアルガイド

ver 1.0

2004 年 10 月 8 日

<<　変更履歴　>>

変更年月日	変更内容
2004/10/08	初版

<< 目次 >>

1 はじめに

1.1 本書の目的

本書は、サンプルプログラムを通して、IM-Curl 連携 API による開発のエッセンスを読者に伝えることを目的とします。本書を通読することで、IM-Curl 連携 API の利用方法を理解することができ、intra-mart 上の Web システムと Curl アプレットが相互に通信しあうアプリケーションを作成することができるようになるでしょう。

一方で、本書に含まれない内容もあります。それは以下のとおりです。

- Java 言語の説明
- JSP および Servlet などの ServerSideJava プログラミングに関する説明
- J2EE 技術に関する説明
- Curl に関する説明
- J2EE ベースの intra-mart のプログラミングに関する説明
- UML に関する説明

対象読者の節でも触れますが、これらの知識は本書を読み進めるための前提条件にもなります。

また、本書で用意するサンプルプログラムはあくまでも、IM-Curl 連携 API によるプログラム作成の流れを理解することに主眼をおいていますので、必ずしもベストなコーディング方法とはいえない方法もあえて取っている個所があります。あくまでも、サンプルとしての位置付けでとらえるようにしてください。

1.2 対象読者または前提条件

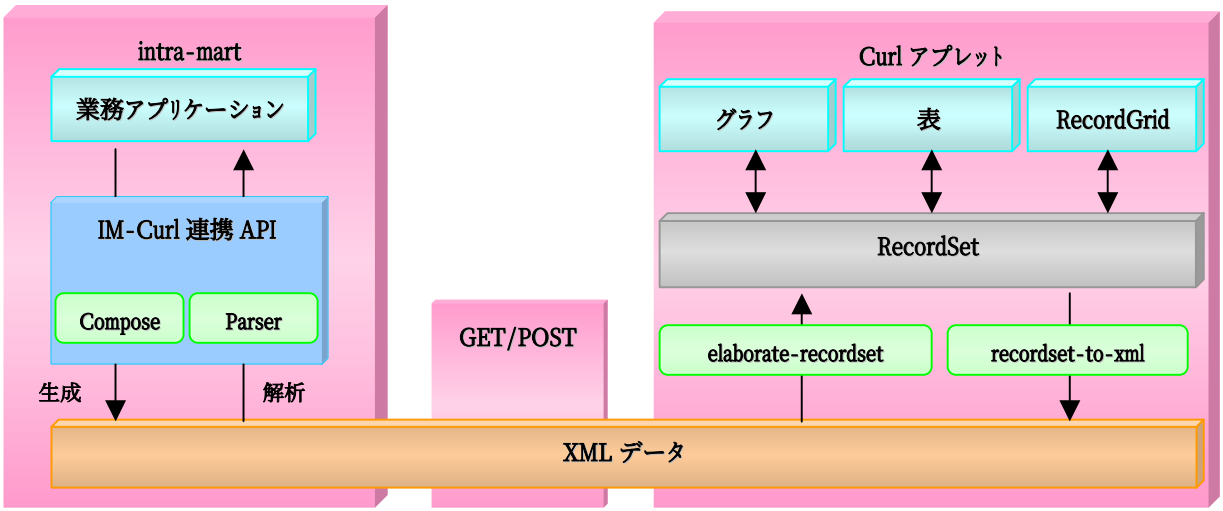
- IM-Curl 連携 API は Java 言語で記述された API です。J2EE 開発モデルでのみ利用可能です。**ページベース開発モデルでは利用できません**。Java 言語の知識がなくても通読できる内容になっていますが、Java 言語の知識があれば、よりスムーズに使用方法を理解することができます。

- 本書は、Curl の解説を目的とするものではありません。したがって、本書を読み進めるために、基本的な Curl に関する理解が必要になります。それほど高度な知識が必要になるわけではありませんが、Curl アプレットの作成方法、動作概要等を理解している読者を対象としています(また、それを強くお勧めします)。

- IM-Curl 連携 API では、CSK(Curl Starter Kit)という Curl を開発する上で非常に有用なコンポーネントをひとつにまとめた、コンポーネント群を使用します。このコンポーネント群は Curl の開発環境である Surge Lab IDE の製品版に付属しております。IM-Curl 連携 API を使用して intra-mart と Curl の連携アプリケーションを構築する際にはこのコンポーネント群が必須になりますので前もって購入しておく必要があります。

1.3 IM-Curl 連携 API の構造

IM-Curl 連携 API とは、intra-mart 上の Web システムと、Curl アプレットが相互に通信するためのインタフェースを規定し、またその通信内容を intra-mart でも Curl アプレットでも簡単に扱えるようにするための JavaAPI 群の総称です。Curl では Web サーバとの通信に HTTP/HTTPS プロトコルを使用し、①GET/POST による通信、②XML データの GET/POST による通信、③SOAP による通信の 3 つの方法を使用することができます。また、Curl での汎用データ保持クラスとして使用される RecordSet クラスを使用すると、CSK にて提供されている recordset-to-xml プロシージャ、elaborate-recordset プロシージャの 2 つのプロシージャを使用することにより Curl 内のオブジェクト表現と文字列(XML)とを簡単に相互に変換することが可能です。IM-Curl 連携 API では上記 2 つのプロシージャで扱うことができる XML データを intra-mart 側で生成･解析する機能を提供します。



1.4 準備

まず、最初にサンプルプログラムを実行するための準備をしましょう。

本書で利用するサンプルプログラムは im-J2EE Framework を用いたアプリケーションなので、intra-mart BaseModule か intra-mart Framework が必要です。バージョンは「Ver4.3.3」を前提としています。開発用の環境に関しましては、それぞれの製品に付属するインストールマニュアルをもとに、開発用の環境を作成してください。

また、Curl アプレットの実行に Surge RTE 及び、CSK が必要になります。バージョンは「Ver3.0.4」を前提としています。開発用の環境に関しましては、それぞれの製品に付属するインストールマニュアルを元に作成してください。Surge RTE のダウンロード方法、開発環境の購入に際しては(株)カール・アジアパシフィック社のサイト(http://www.curlap.com/html/index.html)を参照してください。

次に CSK の配備を行います。Curl の開発環境である Surge Lab IDE の製品版に付属している CSK フォルダ(C:¥Program Files¥Curl Corporation¥Surge¥CSK304¥lib¥CSK)を intra-mart のインストールフォルダ直下の doc/imart フォルダ内の sample/curl フォルダ(例：C:¥imart¥doc¥imart¥sample¥curl)にコピーします。

次にライセンスキーファイルと curl-access.txt ファイルを配備します。ライセンスキーファイルは Surge Lab IDE のインストールフォルダの etc フォルダ(例：C:¥Program Files¥Curl Corporation¥Surge¥4¥etc)の curl-license-3.scurl です。curl-access.txt は intra-mart のインストールフォルダ直下の doc/imart フォルダ内の sample/curl フォルダ(例：C:¥imart¥doc¥imart¥sample¥curl)に格納されています。ファイルの詳細については Curl のヘルプを参照してください。
次に実際にファイルを配備します。上記 2 つのファイルを IM-BaseModule の場合は intra-mart のインストールフォルダ直下の doc フォルダの下(例：C:¥imart¥doc)にコピーしてください。IM-BaseModule でない場合は、お使いのアプリケーションサーバ・WEB サーバのマニュアルに従って WEB ルートディレクトリにコピーしてください。

1.5 ディレクトリ構成

このチュートリアルで使用するサンプルプログラムの主なディレクトリ構成について以下にまとめます。

【intra-mart BaseModule Ver4.3.3】

```
C:\bm433    ・・・ベースモジュール Ver4.3.3 インストールディレクトリ(スタンドアロン)
└ doc          ・・・ライセンスキーファイル、curl-access.txt を配備
   └ imart     ・・・ApplicationServer ルート
      ├ sample
      │  └ curl  ・・・チュートリアル用 Curl アプレットソースファイル、JSP ファイル
      └ WEB-INF
         ├ lib
         │  └ im_curl.jar      ・・・IM-Curl 連携 API 用 JAR ファイル
         └ classes
            ├ ServiceConfig_CurlSample.properties
            │              ・・・チュートリアル用 properties ファイル
            └ jp
               └ co
                  └ intra_mart
                     └ sample
                        └ im_curl
                           ├ ChartDemoServiceController.java
                           ├ InputDemoServiceController.java
                           └ ChartDemoServiceController.java
                                    ・・・チュートリアル用 Java ソースファイル
```

1.6 MIME-TYPE の設定

1.6.1 IM-BaseModule の場合

intra-mart のインストールフォルダ直下の conf フォルダにある http.xml ファイルを修正します。
次の網掛けになっている部分を追加してください。

```
<web-app id="/" directory-servlet="none">
 <classpath id="WEB-INF/classes" source="WEB-INF/classes" compile="true"/>
 <context-param>
  <info>An application information string</info>
 </context-param>
 <mime-mapping>
  <extension>.curl</extension>
  <mime-type>text/vnd.curl</mime-type>
 </mime-mapping>
 <path-mapping url-regexp="^/~([^/]*)" real-path="/home/$1/public_html/"/>
</web-app>
<web-app id="/imart" app-dir="imart" directory-servlet="none">
 <classpath id="WEB-INF/classes" source="WEB-INF/classes" compile="true" encoding="MS932"/>
 <mime-mapping>
  <extension>.curl</extension>
  <mime-type>text/vnd.curl</mime-type>
 </mime-mapping>
</web-app>
```

1.6.2 IM-BaseModule 以外の場合

お使いのアプリケーションサーバ・WEB サーバのマニュアルに従って MIME-TYPE に以下の情報を追加してください。

拡張子	curl
MIME-TYPE	text/vnd.curl

2 IM-Curl 連携 API 使用方法

2.1 IM-Curl 連携 API の概要

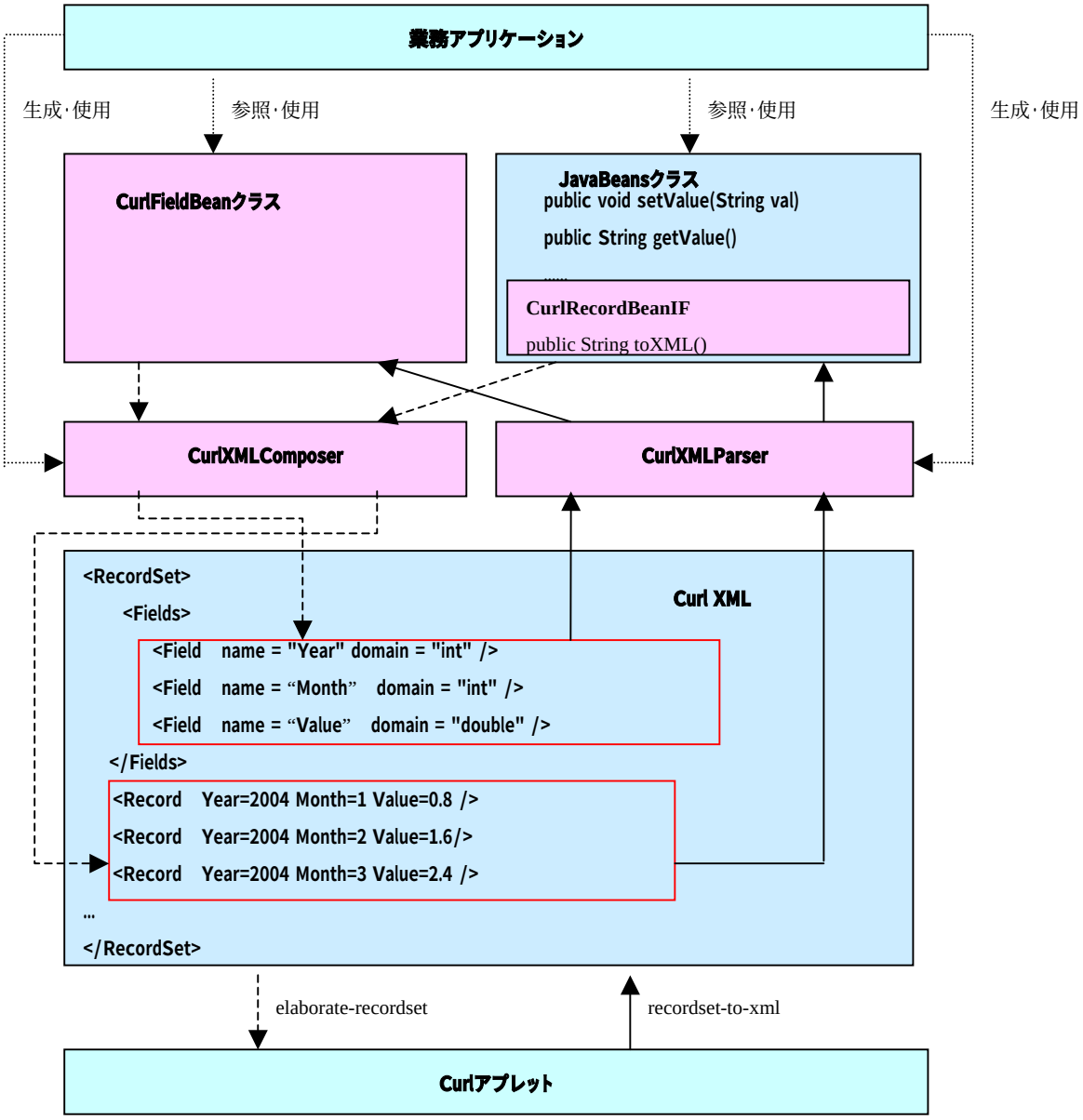
この IM-Curl 連携 API で生成･解析する XML では、Field タグで各項目の定義を、Record タグでは Field タグで定義した属性を持つ行を定義しています。
IM-Curl 連携 API では次の 4 つのクラスを提供します。

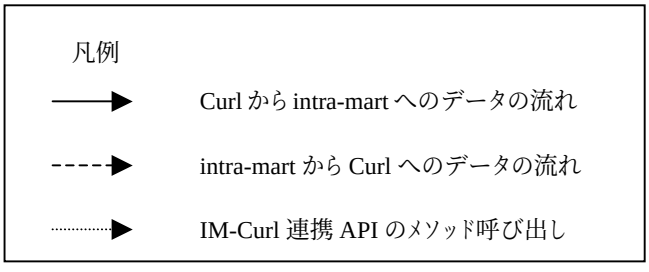
- CurlFieldBean Field　Field タグの情報を持つクラス
- CurlRecordBeanIF　Record タグの情報を持つクラスへのインタフェース
- CurlXMLParser　Curl から送信されてきた XML データを解析･格納するクラス
- CurlXMLComposer　Curl へ送信する XML データを生成するクラス

IM-Curl 連携 API を使用して Curl アプレットと intra-mart を連携させたアプリケーションを作成する際に作成しなければならないものは次のものです。次ページの図では水色で表示しています。

- Curl アプレット
- Curl アプレットで使用する RecordSet の情報を格納する JavaBeans クラス(このクラスでは CurlRecordBeanIF を実装する必要があります)
- intra-mart 上で動作する業務アプリケーション

以上のものを組み合わせて 1 つのアプリケーションを構築します。


業務アプリケーション
生成･使用
参照･使用
参照･使用
生成･使用
CurlFieldBeanクラス
JavaBeansクラス
public void setValue(String val)
public String getValue()
.....
CurlRecordBeanIF
public String toXML()
CurlXMLComposer
CurlXMLParser
<RecordSet>
<Fields>
<Field name = "Year" domain = "int" />
<Field name = “Month” domain = "int" />
<Field name = “Value” domain = "double" />
</Fields>
<Record Year=2004 Month=1 Value=0.8 />
<Record Year=2004 Month=2 Value=1.6/>
<Record Year=2004 Month=3 Value=2.4 />
...
</RecordSet>
Curl XML
elaborate-recordset
recordset-to-xml
Curlアプレット


凡例
Curl から intra-mart へのデータの流れ
intra-mart から Curl へのデータの流れ
IM-Curl 連携 API のメソッド呼び出し

2.2 intra-mart からの Curl アプレット起動方法

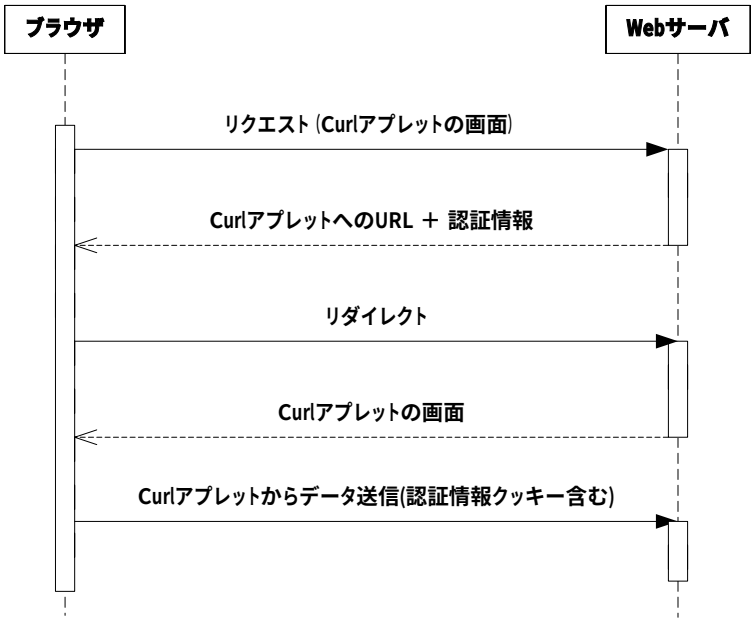
intra-mart では、ログイン後の認証情報をクッキーとしてクライアント側のブラウザに保存しています。intra-mart 上の Curl アプレットをダウンロード・起動すると、見かけ上ブラウザの内側で動作しているように見えますが、プロセスとしてはまったく別のものとなります。

intra-mart にログインすると、ブラウザでは認証情報をクッキーとして受信しますが、Curl アプレットは全く別のプロセスとなるため Curl アプレットに認証情報を明示的に渡さないと、Curl アプレットから intra-mart へデータを送信する際に intra-mart では認証されていないものからの要求であると判断され、通信が拒否されてしまいます。

本書では、Curl アプレットの呼び出し URL に認証情報(セッション ID)を追加して Curl アプレットを起動させ、Curl アプレットでは自らが呼び出された URL から認証情報を取得し、それを intra-mart へのデータ送信時にクッキーとして一緒に送信する、という方法をとっています。

具体的には、ブラウザに一時的な画面を返してリダイレクトを行っています。一時的な画面には Curl アプレットの URL に、セッション情報を追加した URL が含まれており表示後直ちに Web サーバにリクエストを送るようになっています。

Curl アプレットでの認証情報の取得、クッキーの生成、intra-mart へのデータ送信という処理は次節以降のサンプルプログラムのコーディングで使用する access-im.scurl ファイルに記述してあります。詳細はこのファイルをご覧ください。



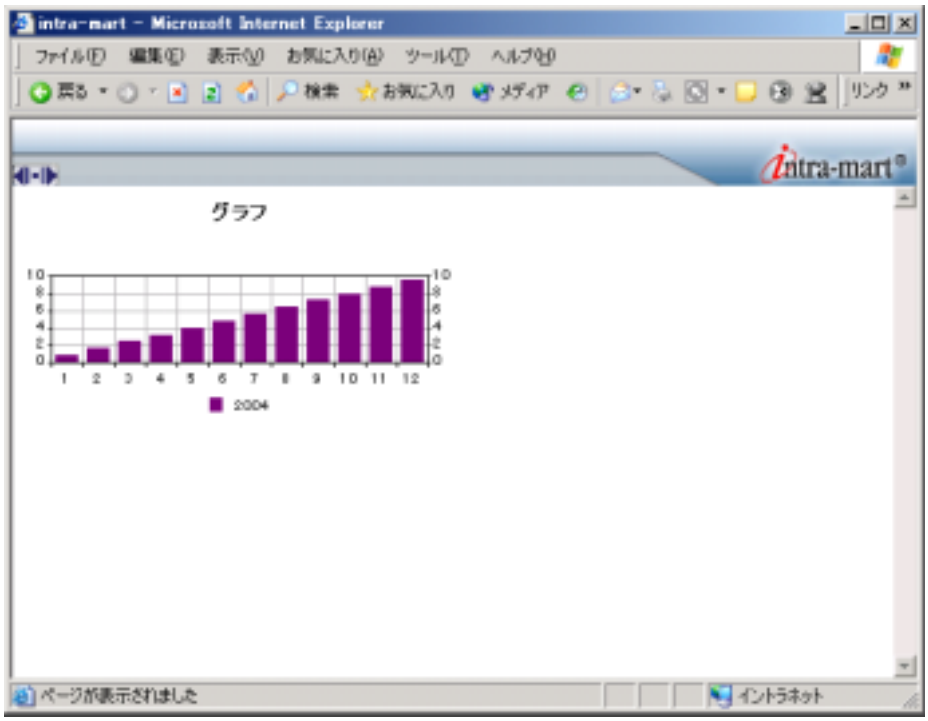
2.3 XML データを intra-mart から Curl へ送信

2.3.1 サンプルプログラムについて

このサンプルプログラムで実装する機能は以下の通りです。

【機能概要】

intra-mart 側のアプリケーションで生成した数値データ(2004 年 1 月から 12 月までの 12 個のデータという想定)を Curl アプレットに送信し、Curl アプレット側では受信したデータをグラフとして表示する。

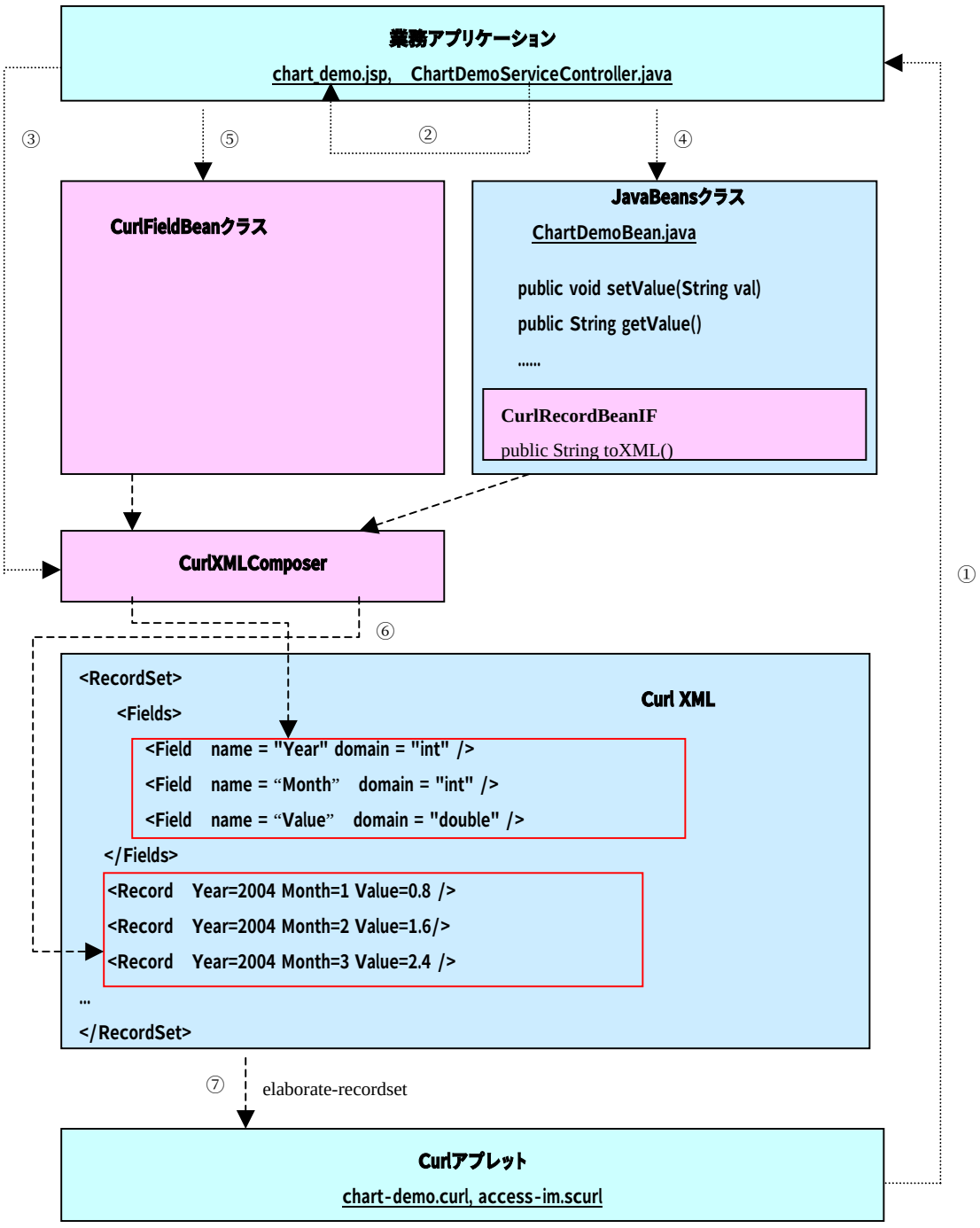


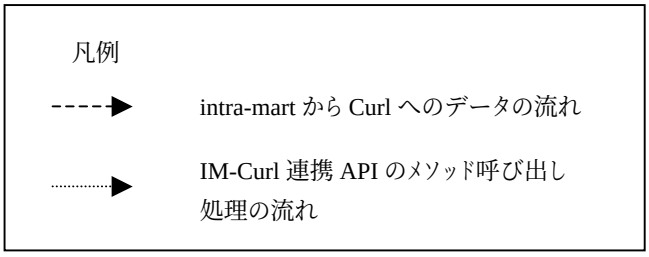
【使用するファイル】

- chart-demo.curl
  Curl アプレット本体です。この中で、グラフオブジェクトの生成、表示領域の大きさなどを定義しています。
- access-im.scurl
  Curl アプレットと intra-mart との通信部分を定義しています。
- chart_demo.jsp
  Curl アプレットに送信する XML データの文字列を生成します。このサンプルプログラムでは JSP 内でデータの生成も行っています。
- ChartDemoBean.java
  Curl アプレット内で使用する RecordSet の情報を保持する JavaBeans です。このサンプルプログラムでは year, month, value の 3 つのフィールドを持っています。
- ChartDemoServiceController.java
  Curl アプレットからリクエストを受けた際に実行されるサービスコントローラです。このサンプルプログラムではなにも処理を行いません。
- ServiceConfig_CurlSample.properties
  Curl からリクエストを受けた際に intra-mart 側でどのプログラムを起動するのか、どういう遷移をするのかを定義したファイルです。このサンプルプログラムでは Curl アプレットから ChartDemoServiceController を起動し、次に chart_demo.jsp へ遷移し結果を受信するという流れになっています。
- chart_demo_caller.jsp
  Curl アプレット呼び出し用 JSP です。

【プログラムの流れの概要】

① Curl アプレットが intra-mart 上の ChartDemoServiceController を呼び出す。
② ServiceConfig_CurlSample.properties に記述してある chart_demo.jsp に処理が移る。
③ chart_demo.jsp では、CurlXMLConposer をインスタンス化する。
④ ChartDemoBean をインスタンス化し、データをセットする。セットしたインスタンスを CurlXMLComposer に追加する。これを 12 回(月のデータを 1 年分)繰り返す。
⑤ CurlFieldBean クラスをインスタンス化し、データをセットする。セットしたインスタンスを CurlXMLComposer に追加する。これを 3 回(Year,Month,Value の 3 つ分)繰り返す。
⑥ CurlXMLComposer クラスの toXML メソッドを呼び出し結果を Curl アプレットに返す。
⑦ Curlアプレットは受信した XML データを elaborate-recordset プロシージャを使用して RecordSet クラスに変換しグラフに表示する。


業務アプリケーション
chart_demo.jsp, ChartDemoServiceController.java
③
⑤
②
④
CurlFieldBeanクラス
JavaBeansクラス
ChartDemoBean.java
public void setValue(String val)
public String getValue()
.....
CurlRecordBeanIF
public String toXML()
CurlXMLComposer
①
⑥
<RecordSet>
<Fields>
<Field name = "Year" domain = "int" />
<Field name = “Month” domain = "int" />
<Field name = “Value” domain = "double" />
</Fields>
<Record Year=2004 Month=1 Value=0.8 />
<Record Year=2004 Month=2 Value=1.6/>
<Record Year=2004 Month=3 Value=2.4 />
...
</RecordSet>
Curl XML
⑦
elaborate-recordset
Curlアプレット
chart-demo.curl, access-im.scurl


凡例
intra-mart から Curl へのデータの流れ
IM-Curl 連携 API のメソッド呼び出し
処理の流れ

2.3.2 サンプルプログラムのコーディング

<%intra-mart インストールディレクトリ%>/doc/imart/sample/curl/chart-demo.curl

```
{curl 3.0 applet}
{curl-file-attributes character-encoding = "shift-jis"}

{applet manifest = "manifest.mcurl"}
{import * from CURL.DATA-ACCESS.BASE}

{import * from COM.CURL.CSK.CHART}
{import * from COM.CURL.CSK.DATASET-CHART}
{import * from COM.CURL.CSK.DATASET-XML}

{include "access-im.scurl"}

{value
    let access-im:Access-IM = {Access-IM}
    let data:RecordSet = {elaborate-recordset {access-im.read-from-im "CurlSample", "chart_demo"}}

    let chart-data:CtData =
        {chart-data-from-recordset
            data,
            "Year",
            "Month",
            "Value"     ・・・・・・ 説明 1
        }

    {VBox
        || Width and height are required since the {example} where
        || this is going doesn't play nicely with PlainDocument
        width = 8cm,
        height = 4cm,
        halign = "center",
        {bold グラフ},
        {CtBarChart
            chart-data,
            bar-width-fraction = 0.75,
            top-margin = 1cm,
            bottom-margin = 1cm,
            left-margin = 0.5cm,
            right-margin = 0.5cm
        }
    }
}
```

<%intra-mart インストールディレクトリ%>/doc/imart/sample/curl/chart_demo.jsp

```
<%@ page contentType="text/xml; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import = "jp.co.intra_mart.foundation.im_curl.*" %>
<%@ page import = "jp.co.intra_mart.sample.im_curl.*" %>
<%@ page import = "java.util.*" %>

<%
   CurlXMLComposer xmlCreator = new CurlXMLComposer();       ・・・ 説明 2
    for (int i = 1; i < 13; i++) {
      ChartDemoBean bean = new ChartDemoBean();                ・・・ 説明 3
      bean.setYear(2004);
      bean.setMonth(i);
      bean.setValue(i * 0.8);
      xmlCreator.addRecord(bean);
    }
    CurlFieldBean field_year = new CurlFieldBean();             ・・・ 説明 4
    field_year.setName("Year");
    field_year.setDomain("int");
    xmlCreator.addField(field_year);

    CurlFieldBean field_month = new CurlFieldBean();            ・・・ 説明 5
    field_month.setName("Month");
    field_month.setDomain("int");
    xmlCreator.addField(field_month);

    CurlFieldBean field_value = new CurlFieldBean();            ・・・ 説明 6
    field_value.setName("Value");
    field_value.setDomain("double");
    xmlCreator.addField(field_value);

%>
<%= xmlCreator.toXML()%>                                        ・・・ 説明 7
```

<%intra-mart インストールディレクトリ%>/doc/imart/WEB-INF/classes/jp/co/intra_mart/sample/im_curl/test/ChartDemoBean.java

```
package jp.co.intra_mart.sample.im_curl;

import jp.co.intra_mart.foundation.im_curl.CurlRecordBeanIF;


public class ChartDemoBean implements CurlRecordBeanIF {
  private int year;
  private int month;
  private double value;

  public String toXMLNode() {
    StringBuffer buffer = new StringBuffer();

    buffer.append("<Record");                                    ・・・ 説明 8
    buffer.append(" Year=\"").append(year).append("\"");
    buffer.append(" Month=\"").append(month).append("\"");
    buffer.append(" Value=\"").append(value).append("\"");
    buffer.append(" />");

    return buffer.toString();
  }

  public int getMonth() {
    return month;
  }

  public double getValue() {
    return value;
  }

  public void setMonth(int i) {
    month = i;
  }

  public void setValue(double d) {
    value = d;
  }

  public int getYear() {
    return year;
  }

  public void setYear(int i) {
    year = i;
  }

}
```

<%intra-mart インストールディレクトリ%>/doc/imart/WEB-INF/classes/jp/co/intra_mart/sample/im_curl/ChartDemoServiceController.java

```
package jp.co.intra_mart.sample.im_curl;

import java.io.*;
import jp.co.intra_mart.framework.base.service.ServiceControllerAdapter;
import jp.co.intra_mart.framework.base.service.ServiceResult;
import jp.co.intra_mart.framework.base.service.RequestException;
import jp.co.intra_mart.framework.system.exception.ApplicationException;
import jp.co.intra_mart.framework.system.exception.SystemException;

import java.util.*;
import javax.servlet.http.HttpSession;

import jp.co.intra_mart.foundation.im_curl.*;

public class ChartDemoServiceController extends ServiceControllerAdapter {
  public ServiceResult service() throws SystemException, ApplicationException {
    try{

      // 必要であればここでイベントフレームワークを呼び出し、
      // 結果を JSP 等の遷移先プログラムへ引き渡します。

    }catch(Exception e){
      e.printStackTrace();
    }

    return null;
  }
}
```

2.3.3 説明

【説明1】 Curl アプレットでのデータ構造定義

Curl アプレットでのデータ構造を定義します。この例では、column-field に Year、row-field に Month、value-field に Value を割り当てています。ここで指定した値と同じ属性の Record タグを生成することになります。

【説明2】 CurlXMLComposer のインスタンス生成

CurlXMLComposer クラスのインスタンスを生成します。引数は不要です。

【説明3】 ChartDemoBean のインスタンス生成

ChartDemoBean クラスのインスタンスを生成し、値をセットします。

【説明4～6】 CurlFieldBean のインスタンス作成

CurlFieldBean クラスのインスタンスを生成し、値をセットします。ここで setName メソッドに渡している引数が説明 1 で指定した文字列と同じものとなっています。

【説明 7】 XML データの生成

説明 3～6 で、生成したデータを XML の文字列へと変換します。

【説明 8】 JavaBeans クラスの XML ノード生成メソッド

JavaBeans クラスで保持している情報を実際に XML ノードへ変換するメソッドです。IM-Curl 連携 API では CurlRecordBeanIF を実装するクラスとして JavaBeans を生成します。

2.4 XML データを Curl から intra-mart へ送信

2.4.1 CurlXMLParser はどのように JavaBeans クラスを使用するか？

IM-Curl 連携 API では、Curl の RecordSet クラスの情報をもつクラスとして JavaBeans を使用します。ただし、インタフェースのみを提供し実際のクラスは提供しません。これは、どのようなデータを処理するのかはアプリケーション固有のものであるため前もって提供することは不可能だからです。

IM-Curl 連携 API では、Java のリフレクション機能を利用し、CurlFieldBean クラスの情報と、指定された JavaBeans の情報からこのデータはどのメソッドを使用すればよいかを判断し、指定された JavaBeans クラスに格納していきます。

CurlXMLParser クラスは次のような順序で処理を行います。

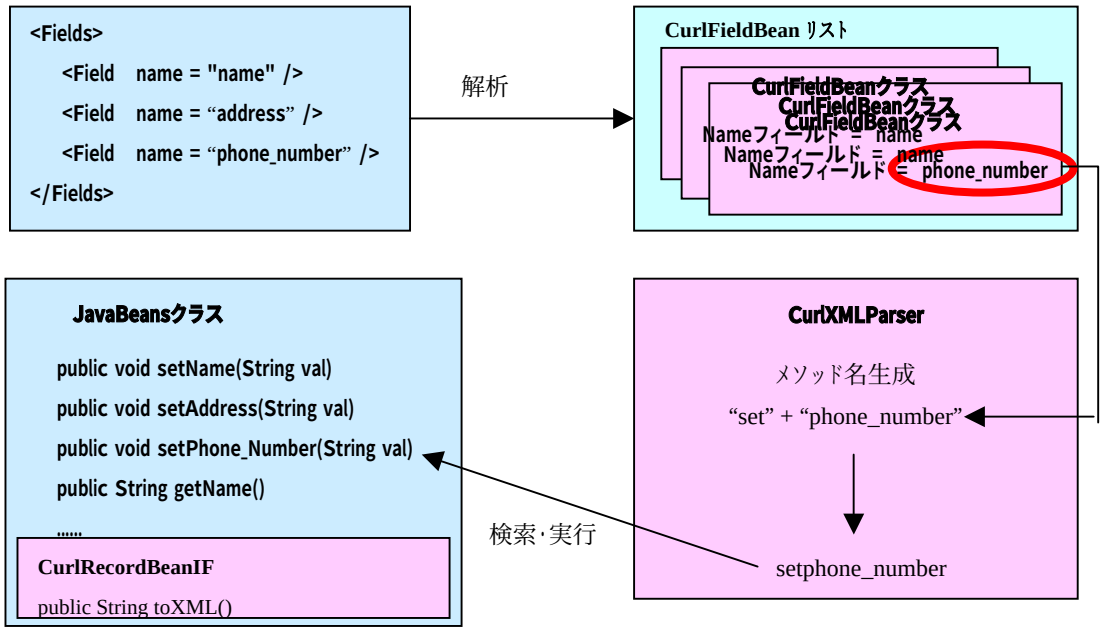
1. Curl アプレットから送信された XML データを解析し、CurlFieldBean のリストを生成する
2. 指定された JavaBeans クラスのインスタンスを生成する
3. 生成したインスタンスのメソッド一覧を取得する
4. 1で生成したリスト内の CurlFieldBean を順に取得し、取得した CurlFieldBean の name フィールド(=Field タグの name 属性)を取得する
5. ”set”という文字列に取得した name 属性を追加し、これを JavaBeans クラスへデータをセットする際のメソッド名とする。
6. 5で決定したメソッド名が3で取得した指定された JavaBeans のメソッド一覧に存在するかをチェックし、存在する場合はそのメソッドを呼び出す

注意点

Curl では変数名にハイフンや、クエスチョンマークを使用することができます。しかし、Java ではこれらの文字を変数名に使用することはできません。従って Curl 側で Field タグの属性に左記のような Java で使用できない文字を使うと正しく解析できません。Curl で使用する Field 名には Java で使用できる名前を付けるようにしてください。

メソッド名検索時に、メソッド名を全て小文字に変換したもので検索を行います。このため、同じメソッド名で大文字小文字が違うメソッド名(例：setName⇔Setname)を使用すると正しく動作しない可能性がありますので、異なるメソッド名を使用するようにしてください。

このサンプルプログラムでは、次の図のような流れになります。

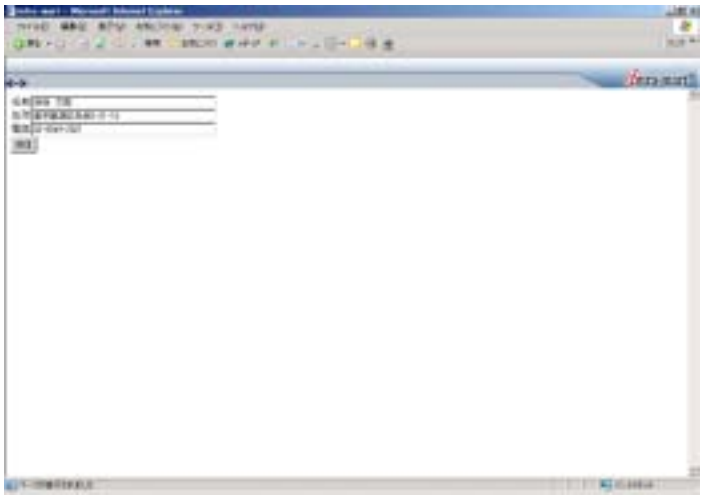


2.4.2 サンプルプログラムについて

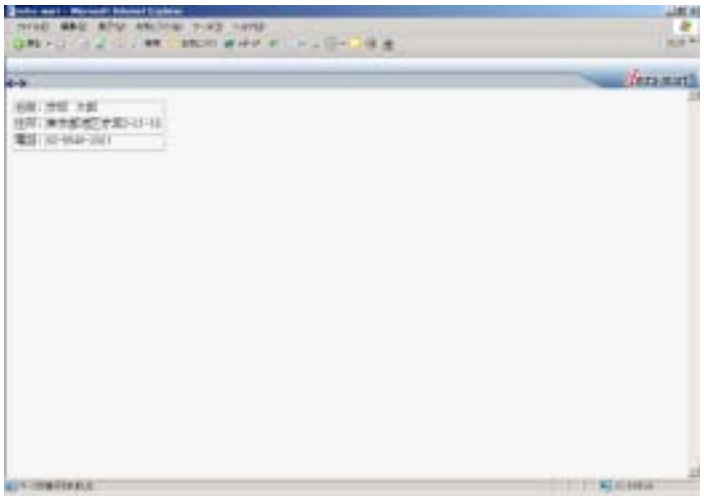
このサンプルプログラムで実装する機能は以下の通りです。

【機能概要】

Curl アプレットで生成した RecordSet クラスから XML データを生成し、intra-mart 側に送信する。受信した XML データを解析し、JavaBeans クラスに格納し、JSP 内で格納された情報を表示する。



Curl アプレットで入力



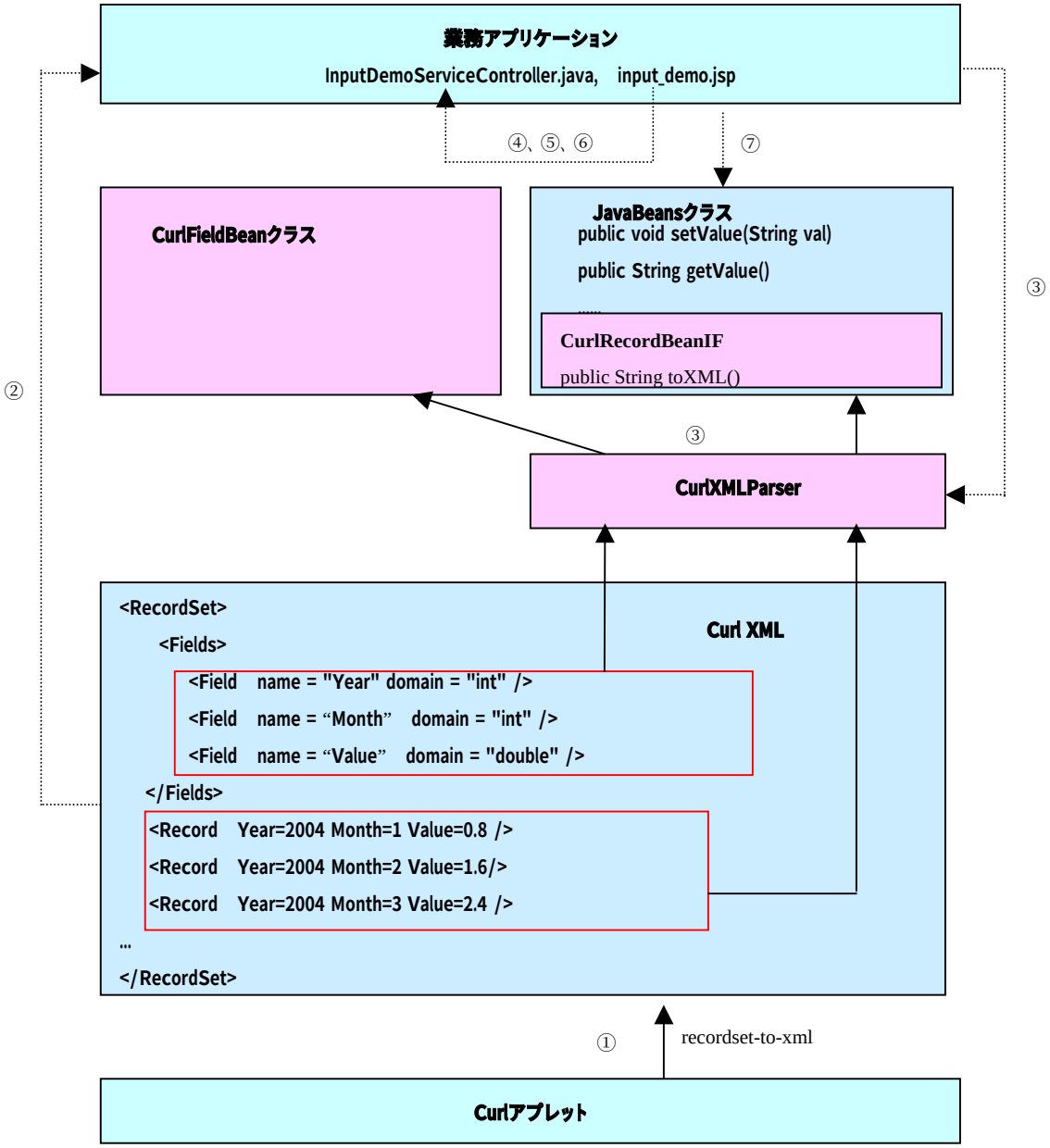
入力された内容を JSP で表示

【使用するファイル】

- input-demo.curl
  Curl アプレット本体です。この中で、テキスト入力オブジェクトの生成、レイアウト、送信ボタンクリック時の処理などを定義しています。
- access-im.scurl
  Curl アプレットと intra-mart との通信部分を定義しています。
- input_demo.jsp
  Curl アプレットから受信した XML データを解析した情報を表示します。
- InputDemoBean.java
  Curl アプレット内で使用する RecordSet の情報を保持する JavaBeans です。このサンプルプログラムでは name, address, phone の 3 つのフィールドを持っています。
- InputDemoServiceController.java
  Curl アプレットからリクエストを受けた際に実行されるサービスコントローラです。このサンプルプログラムでは”xml”という名前で送信された XML データを取得し、解析後 HttpSession へ格納するという処理を行っています。。
- ServiceConfig_CurlSample.properties
  Curl からリクエストを受けた際に intra-mart 側でどのプログラムを起動するのか、どういう遷移をするのかを定義したファイルです。このサンプルプログラムでは Curl アプレットから InputDemoServiceController を起動し、次に input_demo.jsp へ遷移し結果を表示するという流れになっています。
- input_demo_caller.jsp
  Curl アプレット呼び出し用 JSP です。

【プログラムの流れの概要】

① Curl アプレットが入力された情報から RecordSet クラスのインスタンスを作成し、XML に変換する。
② XML に変換したものを InputDemoServiceController へ送信する。
③ InputDemoServiceController では、CurlXMLParser をインスタンス化する。また、インスタンス化する際に受信した XML データと、InputDemoBean を指定する。インスタンス化した際に XML データは CurlFieldBean クラスのリストと InputDemoBean のリストへ格納される。
④ CurlXMLParser から取得した InputDemoBean のリストを HttpSession へ格納する
⑤ ServiceConfig_CurlSample.properties に記述してある input_demo.jsp に処理が移る。
⑥ input_demo.jsp では、セッションから InputDemoBean のリストを取得する。
⑦ 取得したリストから InputDemoBean を取得し、属性値を取り出し HTML へ埋めこむ。これをリストの長さだけ繰り返す。



凡例

Curl から intra-mart へのデータの流れ

IM-Curl 連携 API のメソッド呼び出し処理の流れ

2.4.3 サンプルプログラムのコーディング

<%intra-mart インストールディレクトリ%>/doc/imart/sample/curl/input-demo.curl

```
{curl 3.0 applet}
{curl-file-attributes character-encoding = "shift-jis"}
{applet manifest = "manifest.mcurl"}

{include "access-im.scurl"}

{let
    field-name:TextField = {TextField},
    field-address:TextField = {TextField},
    field-phone:TextField = {TextField},
    post-button:CommandButton =
        {CommandButton
            label="送信",
            {on Action do
                let record:RecordSet =
                    {RecordSet
                        {RecordFields
                            {RecordField "name"},                ・・・ 説明 1
                            {RecordField "address"},
                            {RecordField "phone"}
                        },
                        {RecordData
                            name = field-name.value,
                            address = field-address.value,
                            phone = field-phone.value
                        }
                    }
                let access-im:Access-IM = {Access-IM}
                {access-im.post-to-im "CurlSample", "regist", record, "xml"}          ・・・ 説明 2
            }
        }
}

{VBox
    width = 8cm,
    {HBox "名前", field-name},
    {HBox "住所", field-address},
    {HBox "電話", field-phone},
    post-button
}
```

<%intra-mart インストールディレクトリ%>/doc/imart/WEB-INF/classes/jp/co/intra_mart/sample/im_curl/InputDemoServiceController.java

```
package jp.co.intra_mart.sample.im_curl;
import java.io.*;
import jp.co.intra_mart.framework.base.service.ServiceControllerAdapter;
import jp.co.intra_mart.framework.base.service.ServiceResult;
import jp.co.intra_mart.framework.base.service.RequestException;
import jp.co.intra_mart.framework.system.exception.ApplicationException;
import jp.co.intra_mart.framework.system.exception.SystemException;

import java.util.*;
import javax.servlet.http.HttpSession;

import jp.co.intra_mart.foundation.im_curl.*;

public class InputDemoServiceController extends ServiceControllerAdapter {
 public ServiceResult service() throws SystemException, ApplicationException {
  try{
   String xml =
        new String(((String) getRequest().getParameter("xml")).getBytes("SJIS"));    ・・・ 説明3
   CurlXMLParser parser =
      new CurlXMLParser(
        new ByteArrayInputStream(xml.getBytes("UTF-8")),
        "jp.co.intra_mart.sample.im_curl.InputDemoBean");                            ・・・ 説明4
   ArrayList recordList = parser.getRecordList();                                    ・・・ 説明5
   HttpSession session = getRequest().getSession(false);
   session.setAttribute("recordList", recordList);                                   ・・・ 説明6

  }catch(Exception e){
   e.printStackTrace();
  }

  return null;
 }
}
```

<%intra-mart インストールディレクトリ%>/doc/imart/WEB-INF/classes/jp/co/intra_mart/sample/im_curl/sample/InputDemoBean.java

```
package jp.co.intra_mart.sample.im_curl;

import jp.co.intra_mart.foundation.im_curl.CurlRecordBeanIF;

public class InputDemoBean implements CurlRecordBeanIF {
  private String name;
  private String address;
  private String phone;

  public String toXMLNode() {
    StringBuffer buf = new StringBuffer();

    buf.append("<Record");
    buf.append(" name=\"").append(name).append("\"");
    buf.append(" address=\"").append(address).append("\"");
    buf.append(" phone=\"").append(phone).append("\"");
    buf.append(" />");

    return buf.toString();
  }

  public String getAddress() {
    return address;
  }

  public String getName() {
    return name;
  }

  public String getPhone() {
    return phone;
  }

  public void setAddress(String string) {
    address = string;
  }

  public void setName(String string) {
    name = string;
  }

  public void setPhone(String string) {
    phone = string;
  }
}
```

<%intra-mart インストールディレクトリ%>/doc/imart/sample/curl/input_demo.jsp

```
<%@ page contentType="text/html; charset=Shift_JIS" pageEncoding="Windows-31J" %>
<%@ page import = "jp.co.intra_mart.foundation.im_curl.*" %>
<%@ page import = "jp.co.intra_mart.sample.im_curl.*" %>
<%@ page import = "java.util.*" %>
<%
ArrayList recordList = (ArrayList) session.getAttribute("recordList");                ・・・ 説明 7
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 //EN">
<HTML>
<HEAD>
<TITLE>Hello intra-mart.</TITLE>
</HEAD>

<BODY bgcolor="WhiteSmoke">
<table border="1">
<%
  for (int i=0; i<recordList.size(); i++) {
    InputDemoBean bean = (InputDemoBean) recordList.get(i);                           ・・・ 説明 8
%>
     <tr>
     <td>名前：</td><td><%= bean.getName() %></td>
     </tr>
     <td>住所：</td><td><%= bean.getAddress() %></td>
     </tr>
     <td>電話：</td><td><%= bean.getPhone() %></td>
     </tr>
<%
    }
%>
  </table>
</BODY>
<HTML>
```

2.4.4 説明

【説明1】 Curl アプレットでのデータ構造定義

Curl アプレットでのデータ構造を定義します。この例では、name、address、phone という列を定義しています。ここで指定した値と同じ属性の Record タグが生成されることになります。

【説明2】 intra-mart への送信

access-im.scurl というファイルで定義されている post-to-im というプロシージャに生成した RecordSet を引数として渡して呼び出します。

【説明3】 XML データの受信

HTTP のリクエスト情報から説明2で送信した XML データを受信し、String オブジェクトへ格納します。

【説明4】 CurlXMLParser クラスのインスタンス化

取得した String オブジェクトを InputStream へ変換したものと、InputDemoBean を指定して CurlXMLParser クラスをインスタンス化します。JavaBeans を指定する際は完全指定された名前を使用してください。

【説明5】 InputDemoBean のリストの取得

CurlXMLParser クラスの getRecordList メソッドを呼び出し、InputDemoBean のリストを取得します。

【説明6】 InputDemoBean のリストをセッションへ格納

取得したリストをセッションへ格納します。

【説明7】 セッションからリストを取得

説明 6 で格納した InputDemoBean のリストをセッションから取得します。

【説明8】 リストから InputDemoBean を取得

セッションから取得したリストから InputDemoBean を取得します。

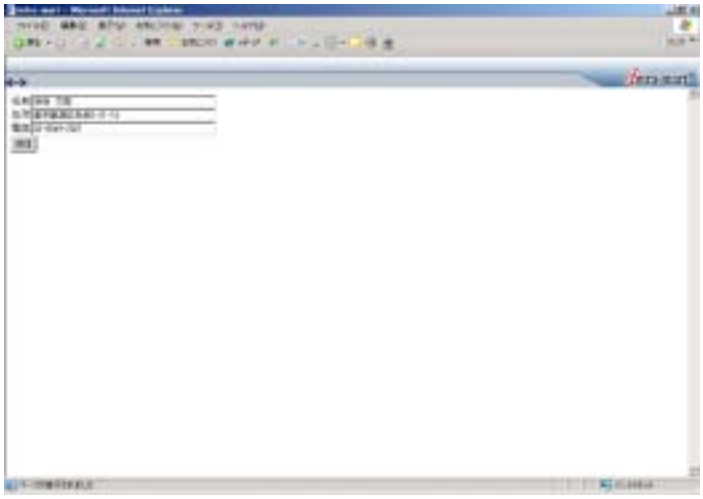
2.5 XML データを Curl から intra-mart、intra-mart から Curl へ送信

2.5.1 サンプルプログラムについて

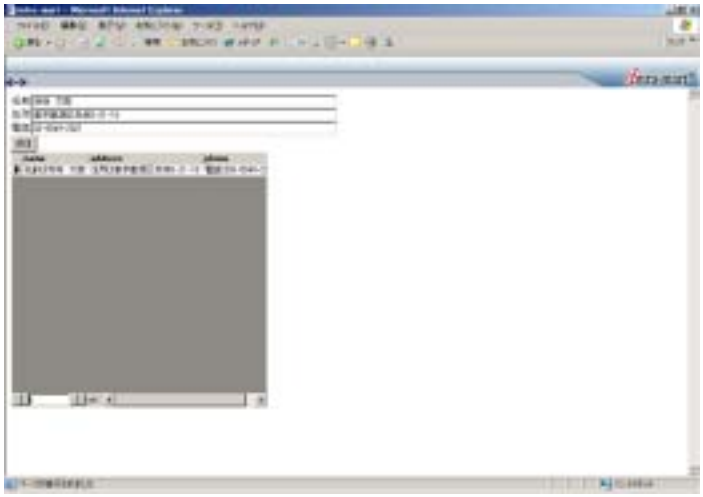
このサンプルプログラムで実装する機能は以下の通りです。2.4で作成したサンプルプログラムを修正したものとなります。

【機能概要】

Curl アプレットで生成した RecordSet クラスから XML データを生成し、intra-mart 側に送信する。受信した XML データを解析し、JavaBeans クラスに修正したデータを格納し Curl アプレットに送信する。



Curl アプレットで入力



入力された内容を intra-mart 側で加工したものを下側の RecordGrid で表示

【使用するファイル】

- return-demo.curl

  Curl アプレット本体です。この中で、テキスト入力オブジェクトの生成、レイアウト、送信ボタンクリック時の処理などを定義しています。

- access-im.scurl

  Curl アプレットと intra-mart との通信部分を定義しています。

- return_demo.jsp

  Curl アプレットから受信した XML データを修正したものを XML データとして Curl アプレットへ送信します。

- InputDemoBean.java

  Curl アプレット内で使用する RecordSet の情報を保持する JavaBeans です。このサンプルプログラムでは name, address, phone の 3 つのフィールドを持っています。2.3 で作成したものをそのまま使っています。

- ReturnDemoServiceController.java

  Curl アプレットからリクエストを受けた際に実行されるサービスコントローラです。このサンプルプログラムでは”xml”という名前で送信された XML データを取得し、解析後修正したデータを HttpSession へ格納するという処理を行っています。。

- ServiceConfig_CurlSample.properties

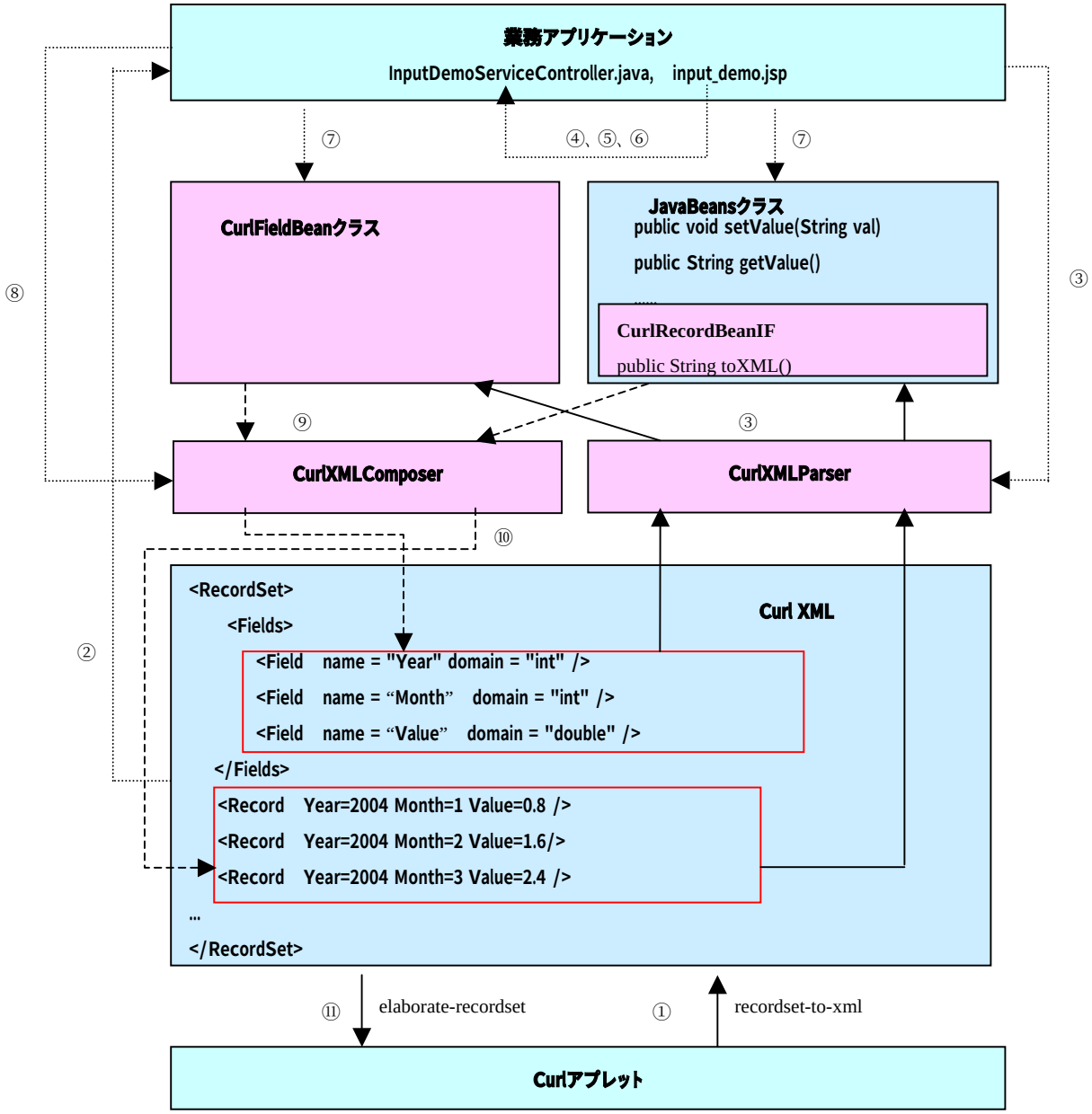
  Curl からリクエストを受けた際に intra-mart 側でどのプログラムを起動するのか、どういう遷移をするのかを定義したファイルです。このサンプルプログラムでは Curl アプレットから ReturnDemoServiceController を起動し、次に return_demo.jsp へ遷移し結果を表示するという流れになっています。

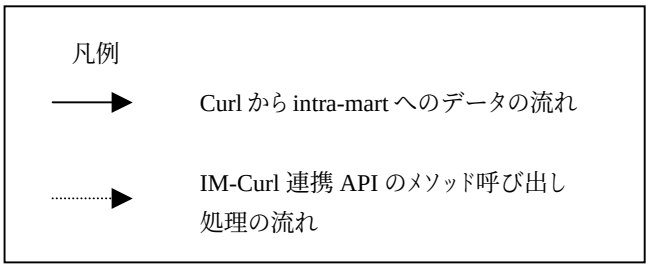
- return_demo_caller.jsp

  Curl アプレット呼び出し用 JSP です。

【プログラムの流れの概要】

① Curl アプレットが入力された情報から RecordSet クラスのインスタンスを作成し、XML に変換する。
② XML に変換したものを ReturnDemoServiceController へ送信する。
③ ReturnDemoServiceController では、CurlXMLParser をインスタンス化する。また、インスタンス化する際に受信した XML データと、InputDemoBean を指定する。インスタンス化した際に XML データは CurlFieldBean クラスのリストと InputDemoBean のリストへ格納される。
④ InputDemoBean のリストの内容を加工する。
⑤ CurlXMLParser から取得した InputDemoBean のリストと CurlFieldBean のリストを HttpSession へ格納する
⑥ ServiceConfig_CurlSample.properties に記述してある return_demo.jsp に処理が移る。
⑦ return_demo.jsp では、セッションから InputDemoBean のリストと CurlFieldBean のリストを取得する。
⑧ CurlXMLComposer のインスタンスを生成する。
⑨ CurlXMLComposer のインスタンスに取得したリストを追加する。
⑩ CurlXMLComposer クラスの xoXML メソッドを呼び出し結果を Curl アプレットへ返す。
⑪ Curl アプレットは受信した XML データを elaborate-xml プロシージャを使用して RecordSet クラスに変換し、RecordGrid に表示する。





2.5.2 サンプルプログラムのコーディング

<%intra-mart インストールディレクトリ%>/doc/imart/sample/curl/return-demo.curl

```
{curl 3.0 applet}
{curl-file-attributes character-encoding = "shift-jis"}
{applet manifest = "manifest.mcurl"}

{include "access-im.scurl"}

{let
	rs:#RecordSet,
	frame:Frame = {Frame},
	field-name:TextField = {TextField},
	field-address:TextField = {TextField},
	field-phone:TextField = {TextField},
	post-button:CommandButton =
		{CommandButton
			label="送信",
			{on Action do
				let record:RecordSet =
					{RecordSet
						{RecordFields
							{RecordField "name"},
							{RecordField "address"},
							{RecordField "phone"}
						},
						{RecordData
							name = field-name.value,
							address = field-address.value,
							phone = field-phone.value
						}
					}
				let access-im:Access-IM = {Access-IM}
				set rs = {elaborate-recordset
							{access-im.read-from-im "CurlSample", "return", record=record}} ・・・ 説明 1
				{frame.add replace?=true,
					{RecordGrid record-source = rs}
				}
			}
		}
}

{VBox
	width = 480px,
	{HBox "名前", field-name},
	{HBox "住所", field-address},
	{HBox "電話", field-phone},
	post-button,
	frame
}
```

<%intra-mart インストールディレクトリ%>/doc/imart/WEB-INF/classes/jp/co/intra_mart/sample/im_curl/ReturnDemoServiceController.java

```
package jp.co.intra_mart.sample.im_curl;
import java.io.*;
import jp.co.intra_mart.framework.base.service.ServiceControllerAdapter;
import jp.co.intra_mart.framework.base.service.ServiceResult;
import jp.co.intra_mart.framework.base.service.RequestException;
import jp.co.intra_mart.framework.system.exception.ApplicationException;
import jp.co.intra_mart.framework.system.exception.SystemException;

import java.util.*;
import javax.servlet.http.HttpSession;

import jp.co.intra_mart.foundation.im_curl.*;

public class ReturnDemoServiceController extends ServiceControllerAdapter {
  public ServiceResult service() throws SystemException, ApplicationException {
    try{
      String xml = new String(((String) getRequest().getParameter("xml")).getBytes("SJIS"));
      CurlXMLParser parser =
        new CurlXMLParser(
          new ByteArrayInputStream(xml.getBytes("UTF-8")), "jp.co.intra_mart.sample.im_curl.InputDemoBean");
      ArrayList recordList = parser.getRecordList();

      for (int i=0; i<recordList.size(); i++) {
        InputDemoBean bean = (InputDemoBean) recordList.get(i);
        bean.setName("名前は" + bean.getName());
        bean.setAddress("住所は" + bean.getAddress());
        bean.setPhone("電話は" + bean.getPhone());                    ・・・ 説明 2
      }

      HttpSession session = getRequest().getSession(false);
      session.setAttribute("fieldList", parser.getFieldsList());
      session.setAttribute("recordList", recordList);
    }catch(Exception e){
      e.printStackTrace();
    }

    return null;
  }
}
```

<%intra-mart インストールディレクトリ%>/doc/imart/sample/curl/return_demo.jsp

```
<%@ page contentType="text/xml; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import = "jp.co.intra_mart.foundation.im_curl.*" %>
<%@ page import = "jp.co.intra_mart.sample.im_curl.*" %>
<%@ page import = "java.util.*" %>

<%
   ArrayList fieldList = (ArrayList) session.getAttribute("fieldList");
   ArrayList recordList = (ArrayList) session.getAttribute("recordList");

   CurlXMLComposer xmlCreator = new CurlXMLComposer();
   xmlCreator.setFieldList(fieldList);
   xmlCreator.setRecordList(recordList);                                    ・・・ 説明 3
%>
<%= xmlCreator.toXML()%>
```

2.5.3 説明

【説明1】 Curl アプレットから intra-mart への XML データの送受信

read-from-im メソッドに RecordSet を指定することで XML データの送信と受信を行います。受信した XML データは直接 elaborate-recordset プロシージャに引数として渡し、RecordSet へと変換しています。

【説明2】 intra-mart での加工

Curl アプレットで入力されたデータを加工しています。

【説明3】 CurlXMLComposer へのリストのセット

ReturnDemoServiceController で加工したレコード一覧リストと、フィールド一覧リストを CurlXMLComposer クラスのインスタンスにセットしています。

IM-Curl 連携 API
チュートリアルガイド

初版　：Octorber 8, 2004

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