# intra-mart WebPlatform/AppFramework Ver.7.2

## Struts 連携 プログラミングガイド

2010/04/01 初版

## ＜＜　変更履歴　＞＞

| 変更年月日 | 変更内容 |
|---|---|
| **2010/04/01** | 初版 |
| | |

# ＜＜　目次　＞＞

# 1 はじめに

## 1.1 目的

im-JavaEE Framework は複数のサブフレームワーク(サービスフレームワーク、イベントフレームワーク等)を含むが、それぞれは疎結合であるため他のフレームワークとも大きな変更を加えずに連携できる。ここでは Web 層のフレームワークとして Jakarta プロジェクトで公開されている Struts を選択し、Struts と intra-mart を連携する方法について述べる。尚、intra-mart.7.2には Struts 1.3.8 があらかじめインストールされている。

# 2 アプリケーションの作成

ここでは intra-mart と Struts を連携させたアプリケーションの作成方法を説明する。

## 2.1 StrutsからIm-JavaEE Frameworkのイベントフレームワークへの連携

Struts は J2EE BluePrints と照らし合わせてみると Web 層に特化したフレームワークである。これは im-JavaEE Framework のサービスフレームワークとほぼ一致する。そのため、ここでは Web 層の制御を Struts で行い、バックエンドのビジネスロジックを im-JavaEE Framework のイベントフレームワークで行う方法について述べる。

### 2.1.1 仕組

StrutsとIm-JavaEE FrameworkはStrutsのActionクラスを通じて連携する。im-JavaEE Frameworkのイベントフレームワークを使わずにActionクラス内でビジネスロジックを書くことも可能であるが、ビジネスロジックは外部に出すことを推奨する。実際、Strutsのドキュメント「The Struts User's Guide」の「1.2.1 The Model: System State and Business Logic JavaBeans」[1]にも同様なことが書かれている。

そこでStrutsとim-JavaEE Frameworkを連携する場合、Actionクラスはビジネスロジックの Facade（窓口）として実装する。「図 2-1 im-JavaEE Frameworkのイベントフレームワーク」と「<図 2-2 StrutsとIm-JavaEE Frameworkの連携」を参照。

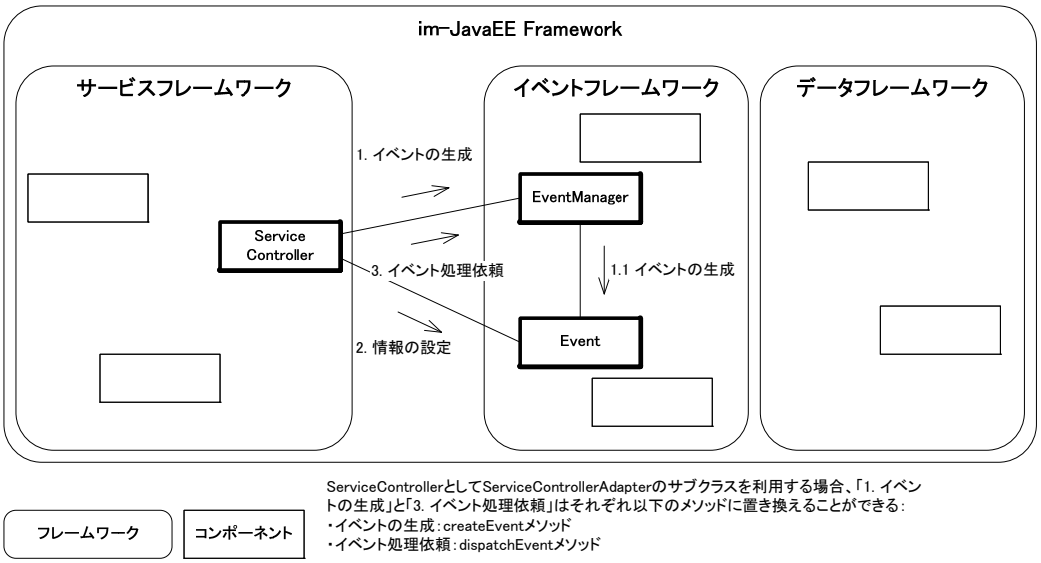
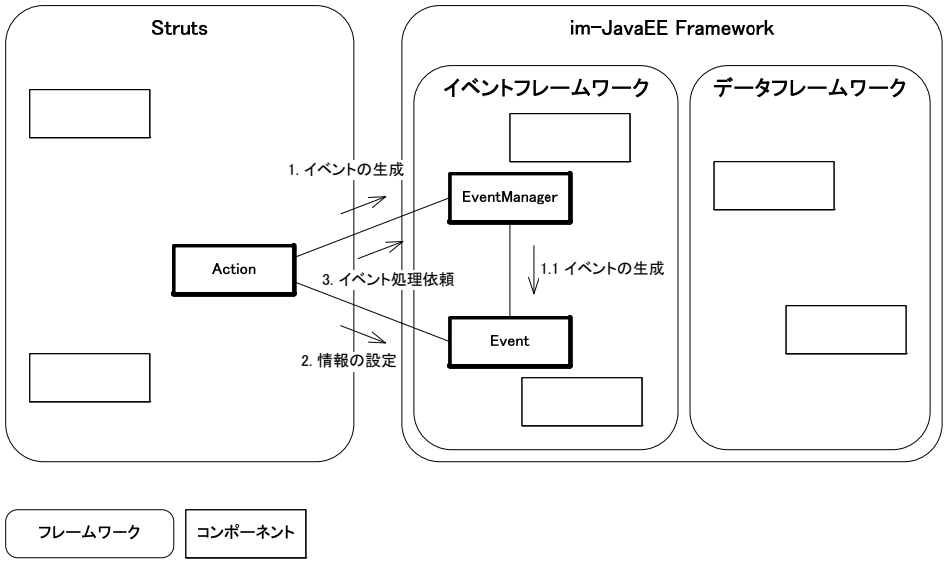



図 2-1 im-JavaEE Framework のイベントフレームワーク>

[1] http://jakarta.apache.org/struts/userGuide/introduction.html#modelConcepts




<図 2-2 Struts と im-JavaEE Framework の連携>

### 2.1.2 呼び出し方法

Struts の Action から im-JavaEE Framework のイベントフレームワークを扱う方法として以下のものが考えられる。

- イベントフレームワークを直接利用

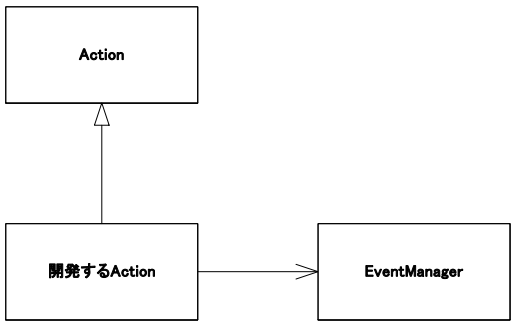
- 共通メソッドが含まれるクラスを継承して利用
- 共通メソッドが含まれるクラスに委譲して利用

これらの中では「共通メソッドが含まれるクラスに委譲して利用」が最も推奨される。この方法は拡張しやすく、変更に伴う影響が少ないものと思われる。

#### 2.1.2.1 イベントフレームワークを直接利用

im-JavaEE Frameworkのイベントフレームワークを Actionクラスから直接利用する場合、「<図 2-3 直接利用」のような構造となる。この場合のコードは「<リスト 2-1 Actionから直接利用」に示すようなものになる。この場合、イベントの生成から処理結果の取得までを開発者が自分でコーディングする必要がある。

この方法は im-JavaEE Framework のイベントフレームワークに接続するすべての Action クラスに対して必要である。そのため、メンテナンスなどの観点から推奨されない。




<図 2-3 直接利用>

<リスト 2-1 Action から直接利用>

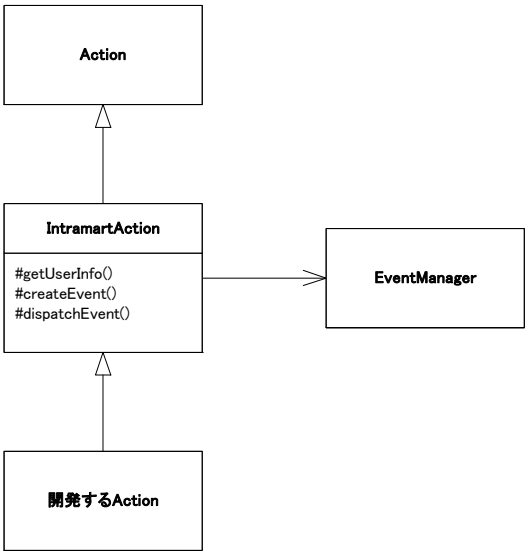
```
import java.io.IOException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

import jp.co.intra_mart.framework.base.util.UserInfo;
import jp.co.intra_mart.framework.base.event.Event;
import jp.co.intra_mart.framework.base.event.EventManager;
import jp.co.intra_mart.framework.base.event.EventResult;
import jp.co.intra_mart.framework.extension.common.util.ServiceUtils;

public class TestAction extends Action {
    public ActionForward perform(ActionMapping mapping,
                                 ActionForm form,
                                 HttpServletRequest request,
                                 HttpServletResponse response)
    throws IOException, ServletException {

        // ログイン情報の取得
        UserInfo userInfo = ServiceUtils.getUserInfo(request, response);

        // イベントの取得
        EventManager em = EventManager.getEventManager();
        TestEvent event =
            (TestEvent)em.createEvent(application_id, key, userInfo);

        // イベントの設定
        event.setAAA(aaa);
        event.setBBB(bbb);

        // イベントの実行
        TestEventResult result = em.dispatch(event);

        // イベント実行結果による処理・遷移
        ・・・
    }
}
```

### 2.1.2.2 共通メソッドが含まれるクラスを継承して利用

これはイベントフレームワークを利用するメソッドを実装したActionクラスのサブクラスを作成する方法である。この方法では、「<図 2-4 継承を利用」のような構造となる。この場合のコードは「<リスト 2-2 イベントフレームワークを使用するメソッドがあるActionクラス」に示すようなものになる。

開発者は以下のクラスを継承して新しい Action クラスを作成する。
jp.co.intra_mart.framework.extension.struts.action.IntramartAction

この方法を採用した場合、Action クラスを実装する開発者はサービスフレームワークの jp.co.intra_mart.framework.base.service.ServiceControllerAdapter とほぼ同様の感覚でコーディングすることが可能である。欠点としては、開発者が他の Action クラスを extends して開発をしたい場合、多重継承の問題が出てくるという点が挙げられる。




<図 2-4 継承を利用>

<リスト 2-2 イベントフレームワークを使用するメソッドがある Action クラス>

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import jp.co.intra_mart.framework.extension.struts.action.IntramartAction;
import jp.co.intra_mart.framework.system.exception.SystemException;

public class SampleAction extends IntramartAction {
    public ActionForward perform(ActionMapping mapping,
                                 ActionForm form,
                                 HttpServletRequest request,
                                 HttpServletResponse response)
    throws IOException, ServletException
    {
        // イベントの取得
        SampleEvent event = null;
        try {
            event =
                (DeleteStaffEvent)
                    createEvent("sampleApp", "sampleKey", request, response);
        } catch (SystemException e) {
            throw new ServletException(e.getMessage(), e);
        }

        // イベントの設定
        SampleForm sampleForm = (SampleForm)form;
        event.setSampleData(sampleForm.getSampleData());

        // イベントの実行
        try {
            dispatchEvent(event);
        } catch (Exception e) {
            throw new ServletException(e.getMessage(), e);
        }

        // 次の画面の準備
        return mapping.findForward("next");
    }
}
```

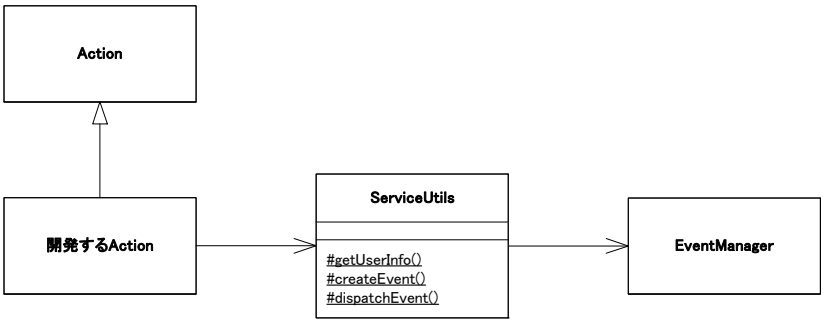
### 2.1.2.3 共通メソッドが含まれるクラスに委譲して利用

これはイベントフレームワークを利用するメソッドを実装したユーティリティクラスをAcitonクラスから利用する方法である。この方法では、「＜図 2-5 委譲を利用」のような構造となる。この場合のコードは「<リスト 2-3 ユーティリティクラスの利用」に示すようなものになる。

開発者は以下のクラスを利用するように Action クラスを作成する。

jp.co.intra_mart.framework.extension.common.util.ServiceUtils

この場合、「2.1.2.2 共通メソッドが含まれるクラスを継承して利用」のような多重継承の問題は出ないが、ユーティリティを使うという点で開発者はServiceControllerAdapterとは若干コーディングスタイルを変更する必要がある。




＜図 2-5 委譲を利用＞

<リスト 2-3 ユーティリティクラスの利用>

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import jp.co.intra_mart.framework.extension.common.util.ServiceUtils;
import jp.co.intra_mart.framework.system.exception.SystemException;

public class SampleAction extends Action {
    public ActionForward perform(ActionMapping mapping,
                                 ActionForm form,
                                 HttpServletRequest request,
                                 HttpServletResponse response)
    throws IOException, ServletException
    {
        // イベントの取得
        SampleEvent event = null;
        try {
            event =
                (SampleEvent)
                    ServiceUtils.
                        createEvent("sampleApp", "sampleKey", request, response);
        } catch (SystemException e) {
            throw new ServletException(e.getMessage(), e);
        }

        // イベントの設定
        SampleForm sampleForm = (SampleForm)form;
        event.setSampleData(sampleForm.getSampleData());

        // イベントの実行
        try {
            ServiceUtils.dispatchEvent(event);
        } catch (Exception e) {
            throw new ServletException(e.getMessage(), e);
        }

        // 次の画面の準備
        return mapping.findForward("next");
    }
}
```

## 2.2 メニュー登録

intra-mart7.2から Struts で作成したアプリケーションを呼び出すためにメニューを登録する。intra-mart7.2にログイングループ管理者でログインし、[ログイングループ管理] – [メニュー管理] - [メニュー設定]からメニューを登録する。登録する URL は「app/index.do」のような通常の Struts を利用して作成されたアプリケーションの URL である。

# 3 付録 A im-JavaEE Framework と Struts

im-JavaEE Frameworkのサービスフレームワークおよびは完全には一致しないが、おおよそ「表 A-1 Struts とim-JavaEE Framework」のように分類することができる。

表 A-1 Struts と im-JavaEE Framework

| 比較項目 | サービス フレームワーク（im-JavaEE Framework） | Struts 1.3 |
|---|---|---|
| 設定 | service-config～.xml | struts-config.xml |
| コントローラ | ServiceServlet | ActionServlet |
| 入力情報の変換 | （なし） | RequestProcessor の processActionForm メソッド |
| 入力情報 | （なし） | ActionForm |
| 入力チェック | ServiceController の check メソッド | ActionForm の validate メソッド |
| Web 層の処理 | ServiceController の service メソッド | Action の perform メソッド |
| キーによる遷移先の決定 | Transition の getNextPage | ActionMapping の findForward メソッド |
| 出力情報の変換 | HelperBean | （なし） |
| 出力情報 | （なし） | ActionForm |
| ファイルアップロード | ServiceControllerAdapter の getEntity メソッド | ActionForm の getMultipart RequestHandler メソッド |

# 4 付録 B 変更内容

## 4.1 4.3 から 5.0 への変更点

### 4.1.1 検証済み Struts

intra-mart 5.0 では Struts 1.2.7 で動作検証を行っている。

### 4.1.2 Struts の組込み方法の変更

intra-mart 4.3 では標準で Struts 1.1 が組込まれていたが、intra-mart 5.0 では利用者が Struts をダウンロードし組込む方式に変更。

### 4.1.3 Struts 連携方法の変更

intra-mart 4.3 以前では intra-mart が保持するログイン情報（ログインユーザ、ログイングループ）を取得するために特殊な仕組みが必要であったため、拡張モジュール（StrutsConnectServlet）を利用して解決していた。
intra-mart 5.0 ではログイン情報が容易にセッションから取り出せるため拡張モジュールを利用することなく intra-mart のメニューから直接 Struts で作成されたアプリケーションに遷移することが可能である。

### 4.1.4 Struts 連携モジュール

- StrutsConnectFilter の追加
- セッション管理を intra-mart 5.0 に含まれている新規の SessionFilter に変更
- ログインユーザ、ログイングループ取得メソッドが非推奨となり、新規にログインユーザ情報を取得するメソッドを追加。

## 4.2 5.0 から 5.1 への変更点

### 4.2.1 検証済み Struts

intra-mart 5.1 では Struts 1.2.8 で動作検証を行っている。

## 4.3 5.1 から 6.0 への変更点

### 4.3.1 Struts を同梱

intra-mart 6.0 では Struts 1.2.9 を同梱している。

## 4.4 6.0 から 6.1 への変更点

### 4.4.1 Struts のバージョンを変更

intra-mart 6.1 では Struts 1.3.8 を同梱している。

intra-mart WebPlatform/AppFramework Ver.7.2
Struts 連携 プログラミングガイド

2010/04/01 初版


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