東京ITスクール
TOKYO IT SCHOOL

ActionForm

目次

1. ActionForm

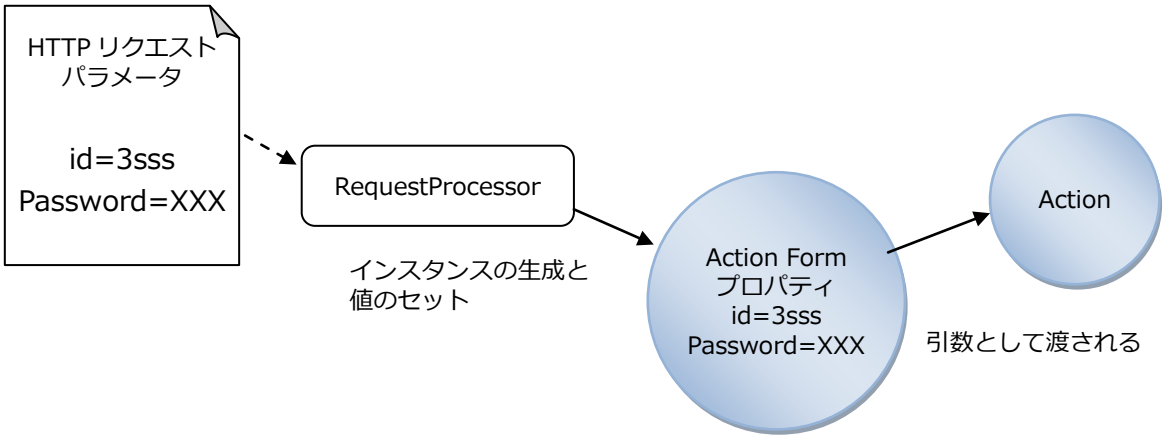
1　ActionForm の動作

Struts では ActionForm を利用してリクエストパラメータの値の取得を手軽に簡単に実装することができます。

Struts では設定ファイルで ActionForm を使用するように定義されていると、自動的にリクエストパラメータの値を読み込んで、ActionForm のプロパティに設定します。

この動作は RequestProcessor によって、Action の execute()メソッドが実行される前に行われます。つまり Action の execute()メソッドでは、HttpServletRequest ではなく、ActionForm を利用してリクエストパラメータの値を取得することができるのです。これには HttpServletRequest からの取得に比べて、値の取り扱いがわかりやすいという利点があります。

【ActionForm の動作】



ActionForm の作成は非常に単純で、フォームから受け取りたいリクエストパラメータをプロパティとして宣言することと、プロパティの値を取得、設定するための getter/setter メソッドを実装するだけです。具体的には、「set (パラメータ名)」「get (パラメータ名)」でパラメータ名の先頭を大文字にしたメソッド名を作成します。たとえば name というパラメータを受け取りたい場合は、setName()という名前の setter メソッドと、getName()という名前の getter メソッドを実装します。

2　基本的な ActionForm

それでは、「Struts の概要と動作」で使用したサンプルを用いて基本的な ActionForm の作成方法を確認していきましょう。

【index.html】

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>ログイン画面</title>
    </head>
    <body>
        <h2>ログイン画面</h2>
        <form action="../sample01/login.do" method="post" >
            ID: <input type="text" name="id"><br>
            PASSWORD: <input type="password" name="password"><br>
            <input type ="submit" value= "ログイン">
        </form>
    </body>
</html>
```

まずは、入力用のフォームを HTML で作成しています。次に ActionForm を確認しましょう。

【LoginForm.java】

```
package sample01;

import org.apache.struts.action.ActionForm;

public class LoginForm extends ActionForm {

    // リクエストパラメータ名と同じプロパティ
    private String id = null;
    private String password = null;

    // getter/setter メソッド
    public String getId() {
        return id;
    }

    public String getPassword() {
        return password;
```

```
    }

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

    public void setPassword(String password) {
        this.password = password;
    }
}
```

ActionForm は org.apache.struts.action.ActionForm を継承して作成します。リクエストパラメータとして送られてくるのは id と password ですので、対応するプロパティと getter/setter メソッドを作成しています。

次に ActionForm を使用するための設定を、Struts 設定ファイルに行います。内容は次のようになります。

【struts-config.xml】

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<!DOCTYPE struts-config PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
    "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">

<struts-config>
    <form-beans>
        <form-bean name="sample01_loginForm" type="sample01.LoginForm"/>
    </form-beans>

    <action-mappings>
        <action path="/sample01/login"
                type="sample01.LoginAction"
                name="sample01_loginForm"
                scope="request">
            <forward name="success" path="/WEB-INF/jsp/sample01/success.jsp"/>
        </action>
    </action-mappings>
</struts-config>
```

まず form-beans 要素を定義し、その中に form-bean 要素を追加します。form-bean 要素では ActionForm の名前と型をそれぞれ name 属性、type 属性で指定します。サンプルの ActionForm は sample01.LoginForm なので、type 属性にその値を、name 属性に名前として sample01_loginForm を指定しています。

続いて、この ActionForm を使用する Action の設定を行います。action 要素の name 属性に、form-bean 要素の name 属性で指定した ActionForm の名前を指定します。

これで設定は完了です。フォームがサブミットされると、リクエストパラメータの値が ActionForm の各プロパティにセットされます。その ActionForm が Action の execute()メソッドに渡されますので、引数にある ActionForm を適当な型にキャストして使用します。以下に、この ActionForm を扱うために Action のコードを示します。

【LoginAction.java】

```
package sample01;

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;

public class LoginAction extends Action {
    public ActionForward execute(ActionMapping mapping,
                                 ActionForm form,
                                 HttpServletRequest request,
                                 HttpServletResponse response)
            throws Exception {

        // ActionForm を LoginForm にキャスト
        LoginForm loginForm = (LoginForm) form;

        //フォームに入力された ID を取得する
        String id =loginForm.getId();

        //フォームに入力されたパスワードを取得する
        String password =loginForm.getPassword();
        String message = " ID : " + id + "  パスワード : " + password;

        // message を request スコープに登録する
        request.setAttribute("message", message);

        // ActionForward を返却
        return mapping.findForward("success");
    }
}
```

最後にフォーワード先の JSP ページを確認します。

【success.jsp】

```
<%@ page contentType="text/html; charset=UTF-8" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>ログイン結果</title>
    </head>
    <body>
        <h2>ログイン結果</h2>
        <%= (String)request.getAttribute("message") %>
    </body>
</html>
```

それでは、動作を確認してみましょう。Tomcat を起動し、下記 URL をブラウザで表示します。

http://localhost:8080/StrutsSample/sample01/index.html

入力ページで ID、パスワードを入力し、「送信」ボタンを押下すると、ActionForm を介してリクエストパラメータの値が取り出され、フォーワード先の JSP ページで表示されていることが確認できます。

3　日本語への対応

実は前項で確認したアプリケーションは、日本語に対応していません。ID に日本語を入力して送信すると、文字化けが発生します。

Struts による Web アプリケーションの開発では、この問題を解決するのに 3 通りの方法があります。

(1) setCharacterEncoding()メソッドを実行するフィルタを設定する
(2) org.apache.struts.action.RequestProcessor の processPreprocess()メソッドをオーバーライドし、その中で setCharacterEncoding()メソッドを実行する
(3) setCharacterEncoding()メソッドを使用せずに、String クラスを用いた文字コードの変換を行う

今回は、最も一般的な対応となる (1)の方法を紹介します。それでは、「/src/filters/」直下に「EncodingFilter.java」を作成しましょう。

【EncodingFilter.java】

```
package filters;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class EncodingFilter implements Filter {
    public void init(FilterConfig arg0) throws ServletException {
    }

    public void doFilter(ServletRequest request,
                         ServletResponse response,
                         FilterChain chain)
            throws IOException, ServletException {
        request.setCharacterEncoding("UTF-8");
        chain.doFilter(request, response);
    }

    public void destroy() {
    }
}
```

続いて、web.xml にフィルタの設定を行います。

【web.xml：フィルタの設定】

```
<web-app>
    <filter>
        <filter-name>Encoding</filter-name>
        <filter-class>filters.EncodingFilter</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>Encoding</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
・・・(略)・・・
```

すべての URL においてフイルタを動作させるためには、filter-mapping 要素にネストされた url-pattern 要素のボディ部に「/*」を指定します。

以上で対応は完了です。ID に日本語を入力しても、文字化けは発生しないことを確認してみましょう。

4　プロパティの型

先ほどの例では ActionForm のプロパティが String 型でしたが、もちろん boolean 型や int 型、double 型などの数値型を使用することもできます。

(1)　プロパティが boolean 型の場合

プロパティが boolean 型の場合は、リクエストパラメータの値が true、yes、on（大文字/小文字は区別しません）のときに true が設定され、それ以外は false が設定されます。チェックボックスや、場合によってはラジオボタンの値を受け取るプロパティに使用するのがよいでしょう。また、boolean 型のプロパティの getter メソッドは、JavaBeans の規約に準じて getXxx()ではなく、isXxx()とするのが望ましいでしょう。

プロパティが boolean 型の例

```
private boolean mailMag = false;

public boolean isMailMag() {
    return mailMag;
}

public void setMailMag(boolean mailMag) {
    this.mailMag rried = mailMag;
}
```

(2) プロパティが数値型の場合

プロパティが数値型の場合は、リクエストパラメータの値が”1"などの数値として解析できるものはその値が設定され、"a"などの数値として解析できないときは 0 が設定されます。

(3) プロパティが配列の場合

プロパティには配列も使用できます。たとえば、以下の HTML フォームから値が送られてくるケースを考えてみます。

プロパティが配列の場合の例（html）

```
<select name="musics" multiple〉
    <option>rock</option>
    <option>pops</option>
    <option>techno</option>
</select>
```

musics というパラメータ名で複数の値が送られてくる可能性がありますので、ActionForm のプロパティおよび getter/setter メソッドは以下のようになります。

プロパティが配列の場合の例（ActionForm）

```
private String[] musics = null;

public String[] getMusics() {
    return musics;
}

public void setMusic(String[] musics) {
    this.musics = musics;
}
```

要素が 1 つ選択された場合、ActionForm には要素数が 1 の配列がセットされ、2 つ選択された場合は要素数が 2 の配列がセットされます。

配列のインデックスを指定して値を設定することも可能です。たとえば、ActionForm で以下のようにすることができます。getter/setter メソッドの引数に、インデックスを表す int 型の引数を取るようにします。

この場合の注意点ですが、配列は要素数を指定して初期化しておかなければなりません。またその要素数は、インデックスよりも大きくしておく必要があります。もしそのように配列を初期化しなかった場合には、setter メソッドで、例外（java.lang.NullPointerException や java.lang.ArrayIndexOutOfBoundsException)が発生します。

要素番号を指定の例（ActionForm）

```
private String[] musics = {"", "",""};

public String getMusics(int index) {
    return musics[index];
}
public void setMusics(int index, String music) {
    this.musics[index] = music;
}
```

これに対応する HTML フォームは、以下のようになります。「プロパティ名[インデックス]」という形のパラメータ名となります。

要素番号を指定の例（html）

```
<input type="text" name="address[0]">
<input type="text" name="address[1]">
<input type="text" name="address[2]">
```

2. DynaActionForm

ActionForm はリクエストパラメータの値を JavaBean にセットする作業を自動化することで開発効率を向上させます。しかし、大規模なアプリケーションでは、プロパティおよび getter/setter メソッドのコーディングだけでも十分な作業量となります。またソースファイルの数も多くなり、ディレクトリ内の見通しも悪くなります。

そこで Struts には、コーディング不要の ActionForm として DynaActionForm という仕組みが用意されています。DynaActionForm は Struts 設定ファイルに設定を行うだけで、すぐに ActionForm として利用できるようになる便利なクラスです。

1 DynaActionForm の使い方

DynaActionForm の使用は簡単です。Struts 設定ファイルに以下のように記述します。

【struts-config.xml:DynaActionForm の設定】

```
<form-bean name="sample02_loginDynaForm"
            type="org.apache.struts.action.DynaActionForm">
    <form-property name="id" type="java.lang.Integer" initial="0"/>
    <form-property name="password" type="java.lang.String" />
</form-bean>
```

上記は Integer 型と String 型のプロパティを、1 つずつ持つ ActionForm の設定例です。form-bean 要素の type 属性に org.apache.struts.action.DynaActionForm を指定し、form-property 要素を form-bean 要素の内部に定義します。form-property 要素の name 属性には ActionForm のプロパティ名を、type 属性にはプロパティの型を指定します。また、必要に応じて initial 属性を定義し、プロパティの初期値を指定することもできます。

DynaActionForm のプロパティとして使用できる型の一覧を以下に示します。

DynaActionForm で使用できる型

使用可能なプロパティ	解釈できない値を設定しようとした場合
java.math.BigDecimal	(例外発生)
java.math.BigInteger	(例外発生)
boolean と java.lang. Boolean	false
byte と java.lang.Byte	0
char と java.lang.Character	(1 文字目のみ設定)
java.lang.Class	(例外発生)
double と java.lang.Double	0
float と java.lang.Float	0
int と java.lang.Integer	0
long と java.lang.Long	0
short と java.lang.Short	0
java.lang.String	(すべての値が解釈される)
java.sql.Date	(例外発生)
java.sql.Time	(例外発生)
java.sql.Timestamp	(例外発生)

なお、initial 属性を省略した場合、プロパティが基本データ型の場合は例外が発生し、オブジェクトの場合は String を除いて null が設定されます。String は空行が設定されます。例外が発生する可能性がありますので、特別な理由がない限り、initial 属性は設定する方が安全です。

2　DynaActionForm の実装例

DynaActionForm は、基本的に ActionForm と同じように扱うことができますが、getter/setter メソッドを実装していないため、Action での値の設定、取得には get()メソッド、set()メソッドを使用します。

それではサンプルで動きを確認しましょう。作成、および変更するファイルの一覧を以下に示します。

ファイル名	解説	ロケーション
struts-config.xml	Struts 設定ファイル	/WEB-INF/
LoginAction.java	入力された ID やパスワードを処理する Action	/src/sample02/
success.jsp	ログイン結果を示す JSP	/WEB-INF/jsp/sample02/
index.html	ID やパスワードの入力フォーム	/sample02/

Eclipse 上から見たディレクトリ構成を以下に示します。

【ディレクトリ構成】

```
StrutsSample
  |
  ├ src
  |   |
  |   └ sample02
  |       |
  |       └ LoginAction.java
  |
  └ WebContent
      |
      ├ sample02
      |   |
      |   └index.html・・・・・・・・・・・・・ログインページ
      |
      └ WEB-INF
          |
          ├ jsp
          |   |
          |   └ sample02
          |       |
          |       └ success.jsp ・・・・・・・・ログイン結果を表示する JSP
          |
          ├ struts.config.xml
          └ web.xml
```

【LoginAction.java】

```
package sample02;

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 org.apache.struts.action.DynaActionForm;

public class LoginAction extends Action {
    public ActionForward execute(ActionMapping mapping,
                                 ActionForm form,
                                 HttpServletRequest request,
                                 HttpServletResponse response)
```

```
        throws Exception {

        // DynaActionForm を使用しているので、ActionForm をそれにキャスト
        DynaActionForm dynaForm = (DynaActionForm)form;

        // フォームに入力された ID を取得する
        String id = ((Integer)dynaForm.get("id")).toString();

        // フォームに入力されたパスワードを取得する
        String password = (String)dynaForm.get("password");
        String message = " ID : " + id + "  パスワード : " + password;

        // message を request スコープに登録する
        request.setAttribute("message", message);

        // ActionForward を返却
        return mapping.findForward("success");
    }
}
```

【struts-config.xml】

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<!DOCTYPE struts-config PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
    "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">

<struts-config>
    <form-beans>
        <form-bean name="sample01_loginForm" type="sample01.LoginForm"/>
        <form-bean name="sample02_loginDynaForm"
                    type="org.apache.struts.action.DynaActionForm">
            <form-property name="id" type="java.lang.Integer" initial="0"/>
            <form-property name="password" type="java.lang.String" />
        </form-bean>
    </form-beans>

    <action-mappings>
        <action path="/sample01/login"
                type="sample01.LoginAction"
                name="sample01_loginForm"
                scope="request">
            <forward name="success" path="/WEB-INF/jsp/sample01/success.jsp"/>
        </action>
```

```
        <action path="/sample02/login"
                type="sample02.LoginAction"
                name="sample02_loginDynaForm"
                scope="request">
            <forward name="success" path="/WEB-INF/jsp/sample02/success.jsp"/>
        </action>
    </action-mappings>
</struts-config>
```

【index.html】

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>ログイン画面</title>
    </head>
    <body>
        <h2>ログイン画面</h2>
        <form action="../sample02/login.do" method="post" >
            ID: <input type="text" name="id"><br>
            PASSWORD: <input type="password" name="password"><br>
            <input type ="submit" value= "ログイン">
        </form>
    </body>
</html>
```

【success.jsp】

```
<%@ page contentType="text/html; charset=UTF-8" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>ログイン結果</title>
    </head>
    <body>
        <h2>ログイン結果</h2>
        <%= (String)request.getAttribute("message") %>
    </body>
</html>
```

それでは、動作を確認しましょう。Tomcat を起動し、下記 URL をブラウザで表示します。

http://localhost:8080/StrutsSample/sample02/index.html

DynaActionForm を介してリクエストパラメータの値が取り出され、フォーワード先の JSP ページで表示されていることが確認できます。

3　配列、その他の型を使う場合

DynaActionForm では、配列や java.util.Map を実装したクラス、java.util.List を実装したクラスを使用することができます。配列を使用する場合、Struts 設定ファイルでは以下のように設定します。

【struts-config.xml：配列を型に持つ DynaActionForm の設定】

```
<form-bean name="postalCodeForm"
            type="org.apache.struts.action.DynaActionForm">
    <form-property name="postalCode"
                    type="java.lang.String[]" size="2"/>
</form-bean>
```

form-property 要素の type 属性で配列を指定し、size 属性で配列のサイズを指定します。配列に初期値を指定するには各要素を"'"（シングルクォーテーション）で囲み"，"（カンマ）で区切ります。initial 属性を省略した場合や、initial 属性に解釈不可能な値を設定した場合の挙動は、配列ではなく同じ型のプロパティを単独に指定した場合と同じになります。

```
<form-property name="postalCode" type="java.lang.String[]"
                initial="'277', '0001'"/>
```

Action で DynaActionForm から値を取得するには、以下のようにします。

```
String postalCode1 = (String)dynaForm.get("postalCode", 0);
String postalCode2 = (String)dynaForm.get("postalCode", 1);
```

DynaActionForm の説明は以上になります。