StrutsCatalogVariableScreenFields

It is a quite common that you come across a screen whose common requirement is to layout a part or whole of the screen dynamically based on the runtime decisions made by the user. A typical example is of screen used for submission of report process. Screen for such a functionality is made of two blocks, in the first block user inputs report process name and presses a “Search” button on the basis of report name specified a list of arguments appears in the lower block, against which the user is supposed to provide values.

In general against every field in JSP,

```
ActionForm
```

bean has to have a pair of accessor and mutator methods (getter and setter methods). In this scenario since the number of fields varies there could be a problem if some fields does not have a corresponding accessor and mutator method. Best practices There are three possible solutions to this problem

1. Name all the fields in specific pattern like field1, field2, field3 etc and provide their accessor and mutator methods in the FormBean. In this approach there is a maximum limit on number of fields that can appear on screen.
2. The purpose of

   ```
   FormBean
   ```

   is to reduce the overhead of doing getParameter () in the Servlet. We always have the option of not using a

   ```
   FormBean
   ```

   and doing the getParameter in Action class.
3. Third option is to utilize the setProperty ()/getProperty () methods provided by Struts especially for this type of problem.

In case we have a small and fixed number of fields on screen, it is not a bad idea to go for the first approach described above, but in general it has been found that the last approach is the best one.

To use this, define two methods in your

```
ActionForm
```

bean class. These methods may in turn be written to store the values into a

```
HashMap
```

. Populate all the required fields in the form Action Class or any Helper class.

```
public class testVarForm extends ActionForm
{
  private HashMap hMap = new HashMap();

  public testVarForm()   {  }

  public void setProperty(String key, Object value)  {
    this.hMap.put(key, value);
  }

  public Object getProperty(String key)   {
    return this.hMap.get(key);
  }

  public HashMap getHashMap()   {
    return this.hMap;
  }
  public void setHashMap(HashMap newHMap)
  {
    this.hMap =newHMap;
  }
}
```

Then use these field and respective values in your JSP using the following tag.

```
<html:text property="property(1)"/>
<html:text property="property(2)"/>

OR

<bean:write name="testVarForm" property="property(1)"/>
```

--Puneet Agarwal

Comments:

There's a fourth, and, IMHO, more elegant option available based on the JavaBeans standard: Use indexed properties.

In your HTML, you can use:

```
    <html:text property="X[0]" />;
    <html:text property="X[1]" />;
```

If you use dyna forms, you specify the type as an implementation class of a List (e.g. java.util.ArrayList).

If you are using hand coded forms, just use accessor and mutator methods like:

```
    public Object getX(int index) { return x.get(index); }
    public void setX(int index, Object value) { x.set(index, value); }
```

One Gotcha: These are naive implementations that don't check for index out of bounds; production code must be more robust.

Delve a bit more into the API and you'll find that Struts builds on this concept to allow you to use mapped properties as well.

– Java Architect

Also see: StrutsCatalogInstrumentableForms – Michael McGrady