Struts Dialogs: SelectAction

1. Overview

`SelectAction` is an abstract Struts **Action** that dispatches a browser event to a handler method. This class allows to process a form submission using either regular submit button or an image button. It works with regular links as well.

- Processes submit events (POST) or link events (GET).
- Handles pushbuttons, image buttons and regular links in universal fashion.
- Does not use `parameter` attribute of action mapping.
- Allows to set an arbitrary caption for a pushbutton.
- Button caption can be easily changed at runtime.

2. Technical Details

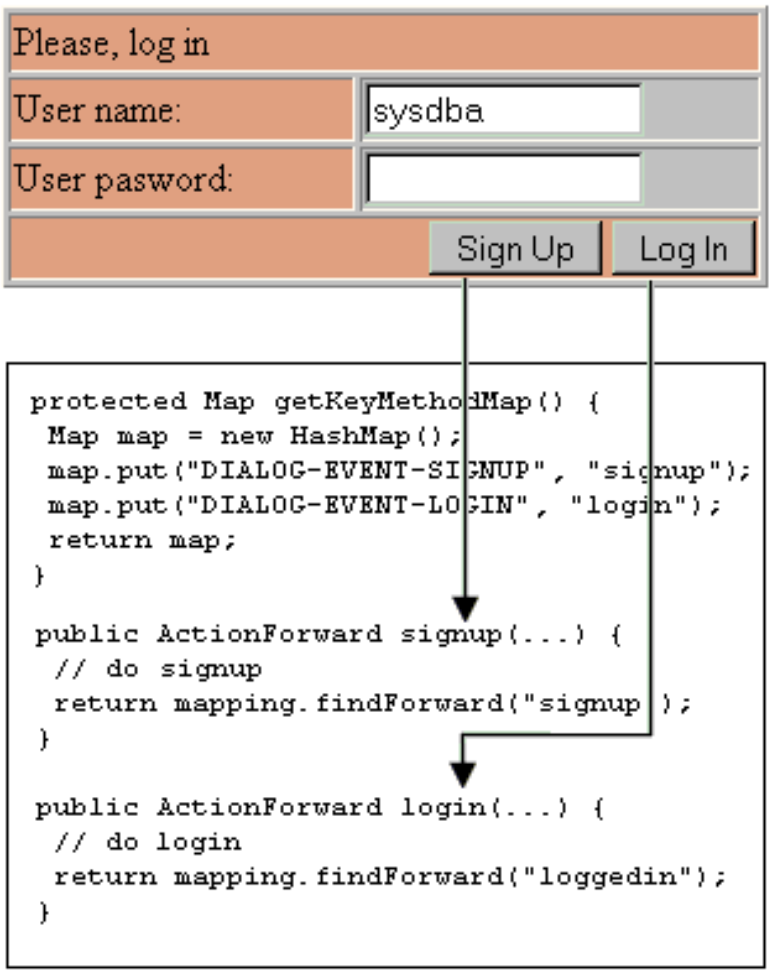
Unlike `DispatchAction` and `LookupDispatchAction` classes, which correlate `value` attribute of `submit` form element with name of a handler method, this class uses `name` attribute. This allows to display a user-friendly caption on a submit button and to change button caption without redeployment.

The subclass must define a map, which correlates event names with method names, and handler methods themselves. Events names must be different from names of action form properties. Each method must have the same signature as `execute` method.

By definition, an event is a request parameter, which starts from a known prefix. The prefix is specified in `getInitKey` method and can be redefined in a subclass. Default prefix is "DIALOG-EVENT".

If you decide to redifine the event prefix, make sure that it does not contain periods. Also, do not use periods for specific event names. Struts may throw an exception taking the name for nested property and trying to set its value.

```
...
<input type="submit" name="DIALOG-EVENT-SIGNUP" value="Sign Up"/>
<input type="submit" name="DIALOG-EVENT-LOGIN" value="Log In"/>
...
```



SelectAction

3. Usage

To use `SelectAction`, subclass it, implement `getKeyMethodMap` method, and map specific events to method handlers. Event name must start with event prefix:

```
protected Map getKeyMethodMap() {
    Map map = new HashMap();
    map.put(getInitKey()+"-ADD", "add");
    map.put(getInitKey()+"-DELETE", "delete");
    map.put(getInitKey()+"-CREATE", "create");
    map.put(getInitKey()+"-LOGIN", "login");
    return map;
}
```

Remember that standard `<html:cancel/>` button is implicitly mapped to `cancelled` method. You need to implement this method to receive cancel events.

Implement handler methods themselves, for example:

```
public ActionForward add(ActionMapping mapping,
                         ActionForm form,
                         HttpServletRequest request,
                         HttpServletResponse response)
     throws IOException, ServletException {
     // do add
     return mapping.findForward("success");
}
```

Use event names in `name` attribute of submit buttons, or as query paramter for regular links:

```
<form action="/selecttest.do" method="post">
    <input type="submit" name="DIALOG-EVENT-ADD" value="Add item"/>
    <input type="submit" name="DIALOG-EVENT-DELETE" value="Delete item"/>
</form>
<input type="image" name="DIALOG-EVENT-LOGIN" src="login.gif" value="Log In">
<a href="/selecttest.do?DIALOG-EVENT-CREATE">Create item</a>
```

4. Notes

- Subclass does not have to redefine `execute` method.
- If duplicate values exist for the keys returned by `getKeyMethodMap`, the first one will be returned. If no corresponding key is found then an exception will be thrown.
- Cancel button must be handled by implementing standard `cancelled` method, if you want to use `isCanceled` method too.
- According to HTML specification, at most one submit element is sent from browser to the server when form is submitted. If form is submitted without explicitly clicking a button, the result depends on a browser. Either no buttons are submitted, or the first button defined on the form is submitted. You need to override `unspecified` method to

handle default submits.

5. Sample Code

- SelectAction Example