Codebehind Plugin

Deprecated Plugin

Since 2.1 this plugin has been deprecated in favor of the Convention Plugin. See this page for details on how to port your application to the Convention plugin.

The Codebehind Plugin reduces mundane configuration by adding "Page Controller" conventions.

There are two common situations where the plugin applies convention over configuration:

1. **Default mappings** - (or "pages with no mappings") These are cases where the page is mostly static and doesn't require an Action class to execute logic. Common examples are index pages and those that heavily use JSP tags or JSF components.
2. **Default results** - The purpose of most Actions is to execute code to prepare the data for a specific page. The name of this page is often the same as the Action itself.

To improve the first case, the plugin will detect the presence of a page with no corresponding Struts mapping and automatically substitute a mapping that uses the default Action class for the package, which is usually ActionSupport, a NO-OP Action.

For the problem of default results, the plugin will make it unnecessary to define those results by detecting the presence of a page for that Action and creating the appropriate configuration on-the-fly.

In these two ways, the plugin encourages a page-based development style, handling the linking of Struts actions with pages and pages with Results in a common way.

✓ To see the plugin in action, review the "Person Manager" example in the Showcase application.

Features

- Provides default mappings for pages that don't have Actions
- Provides default results by auto-discovering pages

Usage

To use this plugin, simply copy its jar into your application. The plugin can be used to find default mappings and results.

Default Mappings

To better facilitate a code-behind development approach, the plugin will detect the case where the request has no defined Struts action mapping, yet there exists a corresponding page. It will then create a dummy action mapping referencing the default Action class (usually ActionSupport), allowing the page to be displayed normally. Additionally, the default interceptor stack for the configured package will be applied, bringing the workflow benefits of interceptor stacks to simple pages.

When no explicitly configured Action can be found for a request, the plugin searches the web application for a likely page. Specifically, the following pattern is used to locate a page:

`/NAMESPACE/ACTION.(jsp|vm|ftl)`

For example, if the request is for `http://www.company.com/myapp/member/login.action`, the plugin will look for the following pages, in this order:

1. `/member/login.jsp`
2. `/member/login.vm`
3. `/member/login.ftl`

If any of those pages are found, the plugin will construct an ActionConfig object on the fly, using the ActionSupport class for the Action and a single Result that points to the discovered page. The ActionConfig will be put in the configured package, meaning that it will inherit the default Interceptor stack for that package. The default package is `codebehind-default`, however, it can be configured in any configuration file via the `struts.codebehind.defaultPackage` constant.

Default Results

In many applications, a majority of Results could have the same root name as the action mapping. To reduce this unnecessary configuration, the Struts plugin will try to guess the appropriate Result, if none is explicitly configured. This technique works for any result code, including `success`. When combined with the Zero Configuration style, the amount of configuration in an application dwindles to next to nothing.

When no explicitly configured Result is found for an Action's result code, the plugin, again, searches the web application for a matching page. Specifically, the following patterns, in the following order, are used to locate a page:

1. `/NAMESPACE/ACTION-RESULT_CODE.(jsp|vm|ftl)`
2. `/NAMESPACE/ACTION.(jsp|vm|ftl)`

These two patterns are searched for each of the three default page extensions: jsp, vm, and ftl. For example, if the request is for `http://www.company.com/myapp/member/login.action`, so that the action name is `login` and the namespace is `member`, and the Action class returned a code of `success`, the plugin will look for the following pages, in this order:

1. `/member/login-success.jsp`
2. `/member/login.jsp`
3. `/member/login-success.vm`
4. `/member/login.vm`
5. `/member/login-success.ftl`
6. `/member/login.ftl`

If any of those pages are found, the appropriate Result will be constructed and processed.

Settings

The following settings can be customized. See the developer guide.

Setting	Description	Default	Possible Values
`struts.codebehind.defaultPackage`	The default package to use for created Action mappings	`codebehind-default`	Any existing package name
`struts.configuration.classpath.disableActionScanning`	Whether to disable scanning the classpath for Action classes or not	`false`	`true` or `false`

Installation

This plugin can be installed by copying the plugin jar into your application's `/WEB-INF/lib` directory. No other files need to be copied or created.