Plugins

Struts 2 plugins contain classes and configuration that extend, replace, or add to existing Struts framework functionality. A plugin can be installed by adding its JAR file to the application's class path, in addition to the JAR files to fulfill whatever dependencies the plugin itself may have. To configure the plugin, the JAR should contain a `struts-plugin.xml` file, which follows the same format as an ordinary `struts.xml` file.

Since a plugin can contain the `struts-plugin.xml` file, it has the ability to:

- Define new packages with results, interceptors, and/or actions
- Override framework constants
- Introduce new extension point implementation classes

Many popular but optional features of the framework are distributed as plugins. An application can retain all the plugins provided with the distribution, or just include the ones it uses. Plugins can be used to organize application code or to distribute code to third-parties.

Packages defined in a plugin can have parent packages that are defined in another plugin. Plugins may define configuration elements with classes not contained in the plugin. Any classes not included in the plugin's JAR must be on the application's classpath at runtime. As from Struts 2.3.5

The framework loads its default configuration first, then any plugin configuration files found in others JARs on the classpath, and finally the "bootstrap" `struts.xml`.

1. `struts-default.xml` (bundled in the Core JAR)
2. `struts-plugin.xml` (as many as can be found in other JARs)
3. `struts.xml` (provided by your application)

Since the `struts.xml` file is always loaded last, it can make use of any resources provided by the plugins bundled with the distribution, or any other plugins available to an application.

Static resources

To include static resources in your plugins add them under "/static" in your jar. And include them in your page using "/struts" as the path, like in the following example:

htmlAssuming /static/main.css is inside a plugin jar, to add it to the page: <@s.url value="/struts/main.css" var="css" /> <link rel="stylesheet" type="text /css" href="%{#css}" />

Read also StaticContentLoader JavaDoc.

Extension Points

Extension points allow a plugin to override a key class in the Struts framework with an alternate implementation. For example, a plugin could provide a new class to create Action classes or map requests to Actions.

The following extension points are available in Struts 2:

{snippet:id=extensionPoints|javadoc=true|url=org.apache.struts2.config.DefaultBeanSelectionProvider}

Plugin Examples

Let's look at two similar but different plugins bundled with the core distribution.

Sitemesh plugin

SiteMesh is a popular alternative to Tiles. SiteMesh provides a common look-and-feel to an application's pages by automatically wrapping a plain page with common elements like headers and menubars.

The `sitemesh-plugin.jar` contains several classes, a standard JAR manifest, and a plugin configuration file.

+ META-INF/ + manifest.mf + org + apache + struts2 + sitemesh + FreeMarkerPageFilter.class + TemplatePageFilter.class + VelocityPageFilter.class + struts-plugin.xml

While the SiteMesh Plugin doesn't provide any new results, interceptors, or actions, or even extend any Struts integration points, it does need to know what settings have been enabled in the Struts framework. Therefore, its `struts-plugin.xml` looks like this:

{snippet:id=all|lang=xml|url=struts2/plugins/sitemesh/src/main/resources/struts-plugin.xml}

The two bean elements, with the "static" flag enabled, tell Struts to inject the current settings and framework objects into static property setters on startup. This allows, for example, the FreeMarkerPageFilter class to get an instance of the Struts FreemarkerManager and the current encoding setting.

Tiles plugin

Tiles is a popular alternative to SiteMesh. Tiles provides a common look-and-feel to an application's pages by breaking the page down into common fragments or "tiles".

The `tiles-plugin.jar` contains several classes, a standard JAR manifest, and a configuration file.

+ META-INF/ + manifest.mf + org + apache + struts2 + tiles + StrutsTilesListener.class + StrutsTileUtilImpl.class + views + tiles + TilesResult.class + struts-plugin.xml

Since the Tiles Plugin does need to register configuration elements, a result class, it provides a `struts-plugin.xml` file.

Plugin Registry

For a list of bundled plugins, see the Plugin Reference Documentation. For more about bundled and third-party plugins, visit the Apache Struts Plugin Registry.

Back to Home