Result Types

Most use cases can be divided into two phases. First, we need to change or query the application's state, and then we need to present an updated view of the application. The Action class manages the application's state, and the Result Type manages the view.

Predefined Result Types

The framework provides several implementations of the `com.opensymphony.xwork2.Result` interface, ready to use in your own applications.

Chain Result	Used for Action Chaining
Dispatcher Result	Used for web resource integration, including JSP integration
FreeMarker Result	Used for FreeMarker integration
HttpHeader Result	Used to control special HTTP behaviors
Redirect Result	Used to redirect to another URL (web resource)
Redirect Action Result	Used to redirect to another action mapping
Stream Result	Used to stream an InputStream back to the browser (usually for file downloads)
Velocity Result	Used for Velocity integration
XSL Result	Used for XML/XSLT integration
PlainText Result	Used to display the raw content of a particular page (i.e jsp, HTML)
Tiles 2 Result	Used to provide Tiles 2 integration
Tiles 3 Result	Used to provide Tiles 3 integration
Postback Result	Used to postback request parameters as a form to the specified destination
JSON Result	Used to serialize actions into JSON

Optional

JasperReports Plugin	Used for JasperReports Tutorial integration	Optional, third-party plugin

Additional Result Types can be created and plugged into an application by implementing the `com.opensymphony.xwork2.Result` interface. Custom Result Types might include generating an email or JMS message, generating images, and so forth.

Default Parameters

To minimize configuration, Results can be configured with a single value, which will be converted into a parameter, and each Result can specify which parameter this value should be set as. For example, here is a result defined in XML that uses a default parameter:

```
<result type="freemarker">foo.fm</result>
```

That is the equivalent to this:

```
<result type="freemarker">
  <param name="location">foo.vm</param>
</result>
```

Since probably 95% of your actions won't need results that contain multiple parameters, this little shortcut saves you a significant amount of configuration. It also follows that if you have specified the default parameter, you don't need to set the same parameter as a specifically-named parameter.

Registering Result Types

All Result Types are plugged in via the Result Configuration.

Extending

You can always extend defined result types and implement whatever logic you need. To simplify process of that, you can define your custom `ResultFactory` and use it with connection with custom interface which your Result implements. Check Define dedicated factory to see how to do it.

Struts 2 provides one such extension for you: `ParamNameAwareResult` interface when used with `StrutsResultBuilder` can limit parameters assigned to the result. So you can simple extend existing result with such a functionality as below:

```
public class MyResult extends ServletDispatcherResult implements ParamNameAwareResult {

    public boolean acceptableParamName(String name, String value) {
        return "accept".equals(name);
    }

}
```

and then register it and use instead of default `dispatcher` result.

Next: DispatcherListener