Integration with existing applications

Integration with existing applications

Contents

You may have an existing JSP (or Struts, Spring MVC, etc.) application that you want to migrate to Tapestry. It's quite common to do this in stages, moving some functionality into Tapestry and leaving other parts, initially, in the other system. You may need to prevent Tapestry from handling certain requests.

How do I make a form on a JSP submit into Tapestry?

Tapestry's Form component does a lot of work while an HTML form is rendering to store all the information needed to handle the form submission in a later request; this is all very specific to Tapestry and the particular construction of your pages and forms; it can't be reproduced from a JSP.

Fortunately, that isn't necessary: you can have a standard HTML Form submit to a Tapestry page, you just don't get to use all of Tapestry's built in conversion and validation logic.

All you need to know is how Tapestry converts page class names to page names (that appear in the URL). It's basically a matter of stripping off the *root-package*`.pages` prefix from the fully qualified class name. So, for example, if you are building a login screen as a JSP, you might want to have a Tapestry page to receive the user name and password. Let's assume the Tapestry page class is `com.example.myapp.pages.LoginForm`; the page name will be `loginform (although, since` Tapestry is case insensitive, LoginForm would work just as well), and the URL will be `/loginform`.

LoginForm.tml

```
<form method="post" action="/loginform">

  <input type="text" value="userName"/>
  <br/>
  <input type="password" value="password"/>
  <br/>
  <input type="submit" value="Login"/>

</form>
```

On the Tapestry side, we can expect that the LoginForm page will be activated; this means that its activate event handler will be invoked. We can leverage this, and Tapestry's RequestParameter annotation:

LoginForm.java

```
public class LoginForm
{
  void onActivate(@RequestParameter("userName") String userName, @RequestParameter("password") String password)
  {
     // Validate and store credentials, etc.
  }
}
```

The RequestParameter annotation extracts the named query parameter from the request, coerces its type from String to the parameter type (here, also String) and passes it into the method.

How do I share information between a JSP application and the Tapestry application?

From the servlet container's point of view, there's no difference between a servlet, a JSP, and an entire Tapestry application. They all share the same ServletContext, and (once created), the same HttpSession.

On the Tapestry side, it is very easy to read and write session attributes:

ShowSearchResults.java

```
public class ShowSearchResults
{
  @SessionAttribute
  private SearchResults searchResults;
}
```

Reading the instance variable `searchResults` is instrumented to instead read the corresponding HttpSession attribute named "searchResults". You can also specify the `value` attribute of the SessionAttribute annotation to override the default attribute name.

Writing to the field causes the corresponding HttpSession attribute to be modified.

The session is automatically created as needed.

How do I put the Tapestry application inside a folder, to avoid conflicts?

Support for this was added in 5.3; see the notes on the configuration page.