SpeclessPagesInWEB-INF

NOTE: This is outdated information that applies only to Tapestry 4. For the current information, see

http://tapestry.apache.org/component-templates.html

How to have specless pages, but keep your templates under WEB-INF

With Tapestry 4.0 and Java 1.5, annotations make it possible to eliminate page specifications. Tapestry will create an implicit specification for any template that is referenced. However, it will only look for these templates in the servlet root, which eliminates the added security of keeping templates under WEB-INF so that they're not directly readable.

One way around this is to use an `ISpecificationResolverDelegate` to look in WEB-INF to generate these implicit specifications. This code will look for specifications relative to the application root, which is where your Tapestry application specification (Foo.application) sits. If that file is in WEB-INF, then this code will search for the templates under WEB-INF, keeping them out of the servlet context root.

The class:

```
public class AppSpecRelativeSpecificationResolverDelegate implements
                ISpecificationResolverDelegate
{
        private String templateExtension;

        public IComponentSpecification findPageSpecification(IRequestCycle cycle,
                        INamespace namespace, String simplePageName)
        {
                String templateName = simplePageName + getTemplateExtension();
                Resource namespaceLocation = namespace.getSpecificationLocation();
                Resource templateResource = namespaceLocation.getRelativeResource( templateName );
                if ( templateResource.getResourceURL() != null )
                {
                        return setupImplicitPage( simplePageName, templateResource, namespaceLocation );
                }

                return null;
        }

        public IComponentSpecification findComponentSpecification(
                        IRequestCycle cycle, INamespace namespace, String type)
        {
                return null;
        }

    private String getTemplateExtension( )
    {
        return templateExtension;
    }

    private IComponentSpecification setupImplicitPage(String simpleName, Resource resource, Resource
namespaceLocation)
    {
        Resource pageResource = namespaceLocation.getRelativeResource(simpleName + ".page");

        IComponentSpecification specification = new ComponentSpecification();
        specification.setPageSpecification(true);
        specification.setSpecificationLocation(pageResource);
        specification.setLocation(new LocationImpl(resource));

        return specification;
    }

    public void setTemplateExtension(String extension)
    {
        if ( extension.charAt(0) != '.' )
                extension = "." + extension;

        templateExtension = extension;
    }
}
```

And here's the necessary .application file config (package names obviously need to change):

```
<extension name="org.apache.tapestry.specification-resolver-delegate"
        class="com.foo.bar.AppSpecRelativeSpecificationResolverDelegate">
            <configure property="templateExtension" value=".html"/>
</extension>
```

OR in hivemodule.xml:

```
	<implementation service-id="tapestry.page.SpecificationResolverDelegate">
		<invoke-factory>
			<construct class="com.foo.bar.AppSpecRelativeSpecificationResolverDelegate">
				<set property="templateExtension" value=".html"/>
			</construct>
		</invoke-factory>
	</implementation>
```