PagesAndComponentsInWEB-INF

NOTE: This is outdated information that applies only to Tapestry 4. For current information see http://tapestry.apache.org/component-templates.html.

How to place pages and components under a specific folder in WEB-INF

The following is an implementation of the ISpecificationResolverDelegate which searches page and component specifications in a given folder under WEB-INF. With this configuration you can reference pages and components without defining them in the application specification. It also enables you to store pages and components in subfolders, which can be useful if the number of files in your WEB-INF grows.

Please notice: The following does **not** work for Home.[page|html] which has to reside directly under WEB-INF!

Also note: This did not work correctly for components until the fix for TAPESTRY-894 was released in a 4.1.1-SNAPSHOT on Dec 10, 2006.

The class:

```
public class MySpecificationResolverDelegate implements ISpecificationResolverDelegate {

        private String pagePath;
        private String componentPath;

        private boolean isInitialized;
        private boolean isCacheDisabled; // not used in sample
        private Resource pagesBaseLocation;
        private Resource componentsBaseLocation;

        /**
         * @param componentPath The componentPath to set.
         */
        public void setComponentPath(String componentPath) {
                this.componentPath = componentPath;
        }

        /**
         * @param pagePath The pagePath to set.
         */
        public void setPagePath(String pagePath) {
                this.pagePath = pagePath;
        }

        /**
         * @see org.apache.tapestry.resolver.ISpecificationResolverDelegate#findComponentSpecification(org.
apache.tapestry.IRequestCycle, org.apache.tapestry.INamespace, java.lang.String)
         */
        public IComponentSpecification findComponentSpecification(IRequestCycle cycle, INamespace namespace,
String type) {
                if (!isInitialized)
                        init(namespace);

                if (namespace.isApplicationNamespace()) {
                        Resource componentResource = componentsBaseLocation.getRelativeResource(type + ".jwc");
                if (componentResource != null)
                        try {
                                return    cycle.getInfrastructure().getSpecificationSource().
getComponentSpecification(componentResource);
                        } catch (ApplicationRuntimeException e) {
                                return null;
                        }
                }
                return null;
        }

        /**
         * @see org.apache.tapestry.resolver.ISpecificationResolverDelegate#findPageSpecification(org.apache.
tapestry.IRequestCycle, org.apache.tapestry.INamespace, java.lang.String)
         */
        public IComponentSpecification findPageSpecification(IRequestCycle cycle, INamespace namespace, String
simplePageName) {
```

```
                if (!isInitialized)
                        init(namespace);

                if (namespace.isApplicationNamespace()) {
                        Resource pageResource = pagesBaseLocation.getRelativeResource(simplePageName + ".page");
                if (pageResource != null)
                        return cycle.getInfrastructure().getSpecificationSource().getPageSpecification
(pageResource);
                }

                return null;
        }

        private void init(INamespace namespace) {
                isCacheDisabled = "true".equals(System.getProperty("org.apache.tapestry.disable-caching"));

                while (!namespace.isApplicationNamespace())
                        namespace = namespace.getParentNamespace();
                if (namespace.isApplicationNamespace()) {
                        Resource namespaceLocation = namespace.getSpecificationLocation();
                        pagesBaseLocation = namespaceLocation.getRelativeResource(pagePath + "/");
                        componentsBaseLocation = namespaceLocation.getRelativeResource(componentPath + "/");
                        isInitialized = true;
                }
        }

}
```

And here's the necessary application configuration. The values of 'pagePath' and 'componentPath' are the path to your pages and components folder under WEB-INF:

```
        <extension name="org.apache.tapestry.specification-resolver-delegate"
                class="foo.bar.MySpecificationResolverDelegate">
                <configure property="pagePath" value="pages"/>
                <configure property="componentPath" value="components"/>
        </extension>
```

OR if you prefer hivemodule.xml:

```
        <implementation service-id="tapestry.page.SpecificationResolverDelegate">
                <invoke-factory>
                        <construct class="foo.bar.MySpecificationResolverDelegate">
                                <set property="pagePath" value="pages"/>
                                <set property="componentPath" value="components"/>
                        </construct>
                </invoke-factory>
        </implementation>
```

Example: Now you can access the page stored under 'WEB-INF/pages/foo/Test.page' and 'WEB-INF/pages/foo/Test.html' by linking to the page as follows:

```
<a jwcid="@PageLink" page="foo/Test" href="">test page</a>
```