PageReferences

WARNING: This page applies to an older version of Tapestry. For Tapestry 5 see http://tapestry.apache.org/component-libraries.html

Referring to pages in a Tapestry library from within the library can be somewhat confusing. Intuitively, you might think that if a page within a library references another page in the same library, that the page names are interpreted relative to the library itself, and so need no kind of prefixing.

After a few moment's thought, you will realize that if you could skip mentioning a prefix for pages within a library, you would need another mechanism to refer to pages at the application (which have no obvious prefix to use by default). So what Tapestry does is to require references to pages within a library to be prefixed with the name of the library.

At this point, you may be thinking: "Why should I have to hard code the library's name/prefix in my page, especially since it's not set by me, but by the client of the library, so I won't even know what it's set to". Never fear: Tapestry provides a mechanism for you to refer to pages within the library without having to explicitly mention the library's id.

Each Tapestry library specification, when parsed, defines a namespace INamespace which contains much of the information provided by the library-specification. Of particular interest here is the fact that the *id* of the namespace is the name that was used by the <library> element in the containing . application or .library file.

In HTML templates, when using components such as PageLink, you can refer to other pages within the same library by supplying a value for the namespace parameter, namely, the namespace of the component or page itself. As an example:

```
    <a href="#" jwcid="@PageLink" page="NextPage" namespace="ognl:namespace" >Go to next page </a>
```

Without this extra parameter, the PageLink would refer to a page called NextPage at the top level, i.e. defined (implicitly or explicitly) in the main . application file.

This works for page references in HTML templates, but what about in Java files? The normal IRequestCycle.getPage(*page*) will return a page reference, but again, it is a reference to an application level page, and not an intralibrary page.

The solution is simple, but involves a few more method invocations. Instead of the normal

```
    IPage page = cycle.getPage("NextPage");
```

you need to ask the namespace to construct a page name within the name space and use that:

```
    String nextPageName = getNamespace().constructQualifiedName("NextPage");
    IPage page = cycle.getPage(nextPageName);
```

Of course, if you found that you were frequently needing to perform this operation, you could factor these two lines into a utility method that you put in a helper class or your own global subclass of BasePage (or BaseComponent).