Default Parameter

Many of the components provided with Tapestry share a common behavior: if the component's id matches a property of the container, then some parameter of the component (usually value) defaults to that property.

This is desirable, in terms of not having to specify the component's id and then specify the same value as some other parameter.

Let's say you have created a component, `RichTextEditor`, which operates like a normal TextArea component, but provides a JavaScript rich text editor. You might start with something like:

```
public class RichTextEditor implements Field
{
  @Property
  @Parameter(required=true)
  private String value;

  . . . // Lots more code not shown here
}
```

However, the weakness here is when you make use of the component. You template may look like:

```
    <t:label for="profile"/>
    <br/>
    <t:richtexteditor t:id="profile" value="profile"/>
```

Every component has a unique id; if you don't assign one with the `t:id` attribute, Tapestry will assign a less meaningful one. Component ids can end up inside URLs or used as query parameter names, so using meaningful ids helps if you are ever stuck debugging a request.

This repetition can be avoided by adding the *autoconnect* attribute to the @Parameter annotation:

```
  @Property
  @Parameter(required=true, autoconnect=true)
  private String value;
```

This can now be written as `<t:richtexteditor t:id="profile"/>`. The unwanted repetition is gone: we set the id of the component and the property it edits in a single pass.

If there is no matching property, then a runtime exception will be thrown when loading the page because the value parameter is required and not bound.

The most common case of using autoconnect is form control components such as TextField and friends ... or this RichTextEditor.

Related Articles

- Default Parameter
- Supporting Informal Parameters
- Enum Parameter Recipe
- Component Parameters