Switching Cases

Switching Cases

With Tapestry's `If` component you can only test one condition at a time. In order to distinguish multiple cases, you'd have to write complex nested if/else constructs in your page template and have a checker method for each test inside your page class.

In cases where you have to distinguish multiple cases, the `Delegate` component comes in. It delegates rendering to some other component, for example a `Block`. For each case you have, you basically wrap the content inside a `Block` that doesn't get rendered by default. You then place a Delegate component on your page and point it to a method inside your page class that will decide which of your Blocks should be rendered.

JumpStart Demo

If, Not, Negate, Switch, Else, Unless

Imagine for example a use case, where you want to distinguish between 4 cases and you have an int property called `whichCase` that should be tested against. Your page template would look as follows:

SwitchMe.tml

```
<html xmlns:t="http://tapestry.apache.org/schema/tapestry_5_4.xsd">
    <body>
        <h1>Switch</h1>

        <t:delegate to="case"/>

        <t:block t:id="case1">
            Here is the content for case1.
        </t:block>

        <t:block t:id="case2">
            Here is the content for case2.
        </t:block>

        <t:block t:id="case3">
            Here is the content for case3.
        </t:block>

        <t:block t:id="case4">
            Here is the content for case4.
        </t:block>
    </body>
</html>
```

You can see, that the `Delegate` component's `to` parameter is bound to the case property of your page class. In your page class you therefore have a `getCase()` method that is responsible for telling the `Delegate` component which component should be rendered. For that we are injecting references to the `Block}}s defined in your page template into the page class and return the according {{Block` in the `getCase()` method.

SwitchMe.java

```
public class SwitchMe
{
    @Persist
    private int whichCase;

    @Inject
    private Block case1, case2, case3, case4;

    public Object getCase()
    {
        switch (whichCase)
        {
            case 1:
                return case1;
            case 2:
                return case2;
            case 3:
                return case3;
            case 4:
                return case4;
            default:
                return null;
        }
    }
}
```

Happy switching!