

# Wicket Forms with Flair



Alastair Maw

London Wicket User Group  
4 September 2007

# Good Usability

- ★ Make it obvious that a field is required.
- ★ Locate validation feedback messages alongside their actual components, rather than in a standalone FeedbackPanel.
- ★ Provide good visual hints that a field is invalid and needs attention.

# Photoshop Mock-Up

Name:  \*

You must fill this in.

Description:

Description:

# Easily Reusable

- ★ Should be trivial to apply the look & feel and functionality to any of your forms.
- ★ No manual configuration of each form component - error prone and tedious.

# ComponentBorder

- ★ Surrounds a component with markup.
- ★ Added with  
`Component.setComponentBorder(...)`.

# MarkupComponentBorder

```
<html>
<body>
  <wicket:border>
    <wicket:body/>
    <span class="requiredHint">*</span>
  </wicket:border>
</body>
</html>
```

# MarkupComponentBorder

```
public class RequiredBorder extends MarkupComponentBorder
{
    public void renderAfter(Component component)
    {
        FormComponent fc = (FormComponent)component;
        if (fc.isRequired())
        {
            super.renderAfter(component);
        }
    }
}
```

# Validation Messages

- ★ You can only have one `IComponentBorder` per Component.
- ★ `MarkupComponentBorder` just lets you wrap things with static HTML.
- ★ For our validation messages, we want to output the error, so this isn't quite going to cut it.

# Behaviors

- ★ Have various callbacks that are invoked while that component's lifecycle progresses.
- ★ Can append to and modify the output of a component.
- ★ Implement `IBehavior`.
- ★ Usually extend `AbstractBehavior`.

# ValidationMsgBehavior

```
public class ValidationMsgBehavior extends AbstractBehavior {
    public void onRendered(Component c) {
        FormComponent fc = (FormComponent)c;
        if (!fc.isValid()) {
            String error;
            if (fc.hasFeedbackMessage()) {
                error = fc.getFeedbackMessage().getMessage().toString();
            } else {
                error = "Your input is invalid.";
            }
            fc.getResponse().write(
                "<div class=\"validationMsg\">" + error + "</div>"
            );
        }
    }
}
```

# Error Highlighting

- ★ Add a red background to the form components that are in error.
- ★ Do this with Cascading Style Sheets.
- ★ Initially, we're going to make things simple, and just add a `class="error"` to the `<input>` tags that have problems.

# ErrorHighlightBehavior

```
public class ErrorHighlightBehavior extends AbstractBehavior
{
    public void onComponentTag(Component c, ComponentTag tag)
    {
        FormComponent fc = (FormComponent)c;
        if (!fc.isValid())
        {
            tag.put("class", "error");
        }
    }
}
```

# Adding the Extras

- ★ Adding the behaviors and component border manually to each FormComponent in a Form would be tedious.
- ★ Do this automatically with an IVisitor.

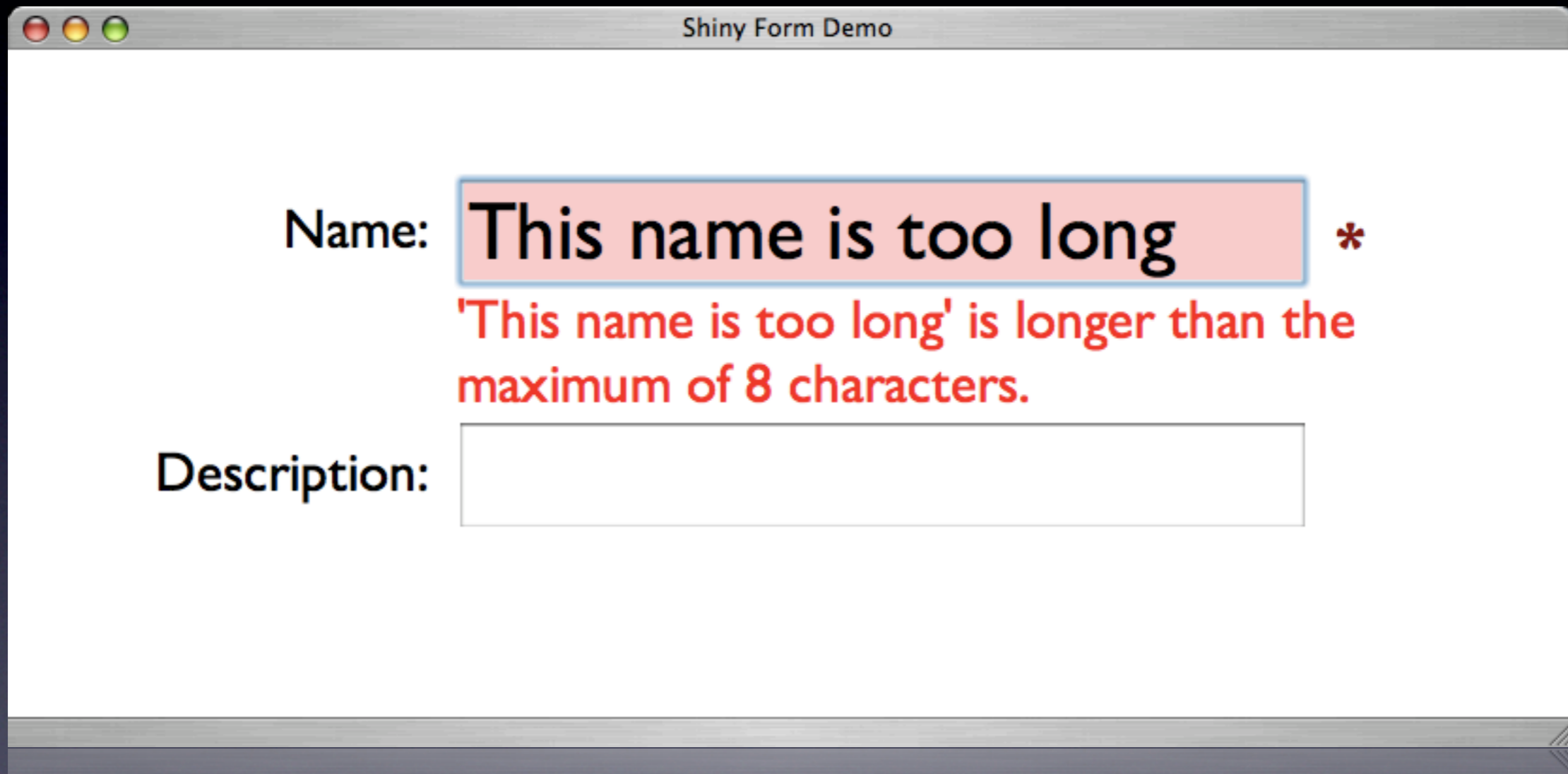
# ShinyFormVisitor

```
public class ShinyFormVisitor implements IVisitor, Serializable
{
    Set visited = new HashSet();
    public Object component(Component c)
    {
        if (!visited.contains(c))
        {
            visited.add(c);
            c.setBorder(new RequiredBorder());
            c.add(new ValidationMsgBehavior());
            c.add(new ErrorHighlightBehavior());
        }
        return IVisitor.CONTINUE_TRAVERSAL;
    }
}
```

# Running the IVisitor

```
public class ShinyForm extends Form
{
    IVisitor shinyVisitor = new ShinyFormVisitor();
    public void onBeforeRender()
    {
        super.onBeforeRender();
        visitChildren(shinyVisitor);
    }
}
```

# Putting It All Together



Shiny Form Demo

Name:  \*

'This name is too long' is longer than the maximum of 8 characters.

Description:

The image shows a screenshot of a web browser window titled "Shiny Form Demo". The window contains a form with two input fields. The first field, labeled "Name:", contains the text "This name is too long" and is highlighted with a red background. To the right of the input field is an asterisk (\*). Below the input field, a red error message reads: "'This name is too long' is longer than the maximum of 8 characters." The second field, labeled "Description:", is an empty text input box.

More London Wicket talks & source code:

<http://londonwicket.org>

More information about Wicket:

<http://wicket.apache.org>