A Widget for Everyone: Fine-Grained Extensions in the Eclipse UI

Jay Jay Billings
Research Staff
Oak Ridge National Laboratory
The Bredesen Center - UTK
billingsjj@ornl.gov
@jayjaybillings
+jayjaybillings

and the rest of ICE Team

and Gustavo Rodriguez, Nick Stanish, Menghan Li, Fangzhou Lin, Ji Soo Ji from Purdue (FB-OA)

20160310
Outline

• The problem
• Attempts 1 - 3
• Success with e4!
• (Bad) Examples

Additional Resources: http://www.eclipse.org/ice

YouTube
https://goo.gl/HpclLq

Ohloh.net
ohloh.net/p/eclipseice
Fine-Grained UI
What is a fine-grained UI extension?

Consider a Form

Can I customize this widget?

Can I change how it is drawn at runtime?

Can I do it without necessarily rewriting the whole code?
What could we do with a Form?

What elements could we change?
But why?

- Nuclear Energy
- Advanced Materials
- Data Analysis
- Batteries
- Quantum Computing
- Astrophysics
- Basic 3D Geometry and 2D Mesh Editing
- Advanced Manufacturing
- More 3rd Party Tools

Coming soon!
But why? (More succinctly)

ICE doesn’t know it is a temperature. Is it better like this

<table>
<thead>
<tr>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some Data</td>
</tr>
<tr>
<td>Temperature:</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

or as a widget?*

Scientists may want one, the other or both.
Can I customize this widget?

*Need focused UI extensions*

Can I change how it is drawn at runtime?

*Need simple, declarative context*

Can I do it without necessarily rewriting the whole code?

*Need “scoped” UI code*
Attempt 1-3
3.x Extension Points

Works perfectly!

But requires HUGE amounts of code.
Java EE

What if we could annotate with @Any:
1. Would pick up all classes without extension points
2. Could be dynamically allocated with one call to CDI or BeanManager
3. Smallest amount of code with annotation lookup

*Does not work.*

Could not get CDI working properly in Eclipse.
E4 with Context Functions

Works perfectly!

Add context

```java
// Create a data component
DataComponent dataComp = new DataComponent();
dataComp.setName("Data");
dataComp.setDescription("Some Data");
dataComp.setContext("demo");
// Need to set the id since geomComp is number 1
dataComp.setId(2);

// Create an Entry for the data component
IEntry entry = new StringEntry();
entry.setName("Data Entry");
entry.setDescription("An Entry with Important Data");
entry.setContext("demo-entry");
// Add the Entry to the data component
dataComp.addEntry(entry);
```

Create widget and provider

```java
public class DemoEntryCompositeProvider implements IEntryCompositeProvider {
    @Override
    public String getName() {
        // TODO Auto-generated method stub
        return "demo-entry";
    }

    @Override
    public IEntryComposite getEntryComposite(Composite parent, IEntry entry,
        int style, FormToolkit toolKit) {
        // TODO Auto-generated method stub
        return new DemoEntryComposite(parent, entry, style);
    }
}
```

But requires an OSGI service.
Attempt 4: E4 with Processor
E4 with Processor

Also works perfectly!

```xml
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0"
    name="org.eclipse.ice.demo.geometry.page"><implementation
class="org.eclipse.ice.demo.ui.DemoGeometryPageContextFunction"/>
    <property
        name="service.context.key" type="String" value="demo-geometry"/>
    <service>
        <provide
            interface="org.eclipse.e4.core.contexts.IContextFunction"/>
    </service>
</scr:component>
```

Replace OSGI service with Processor:

Consolidates services

Much cleaner

Injected directly into model.
Examples
Reflectivity Simulator: Whole Page Replacement
Facebook Open Academy
Evaluate the Sessions

Sign in and vote at eclipsecon.org

-1 0 +1