



OSGi Alliance Community Event

June 26th - 27th, 2007

Siemens AG Campus - Munich, Germany



OSGi Alliance Community Event

Extensions vs Services: Digging Deeper



OSGi Alliance Community Event

June 26th - 27th, 2007

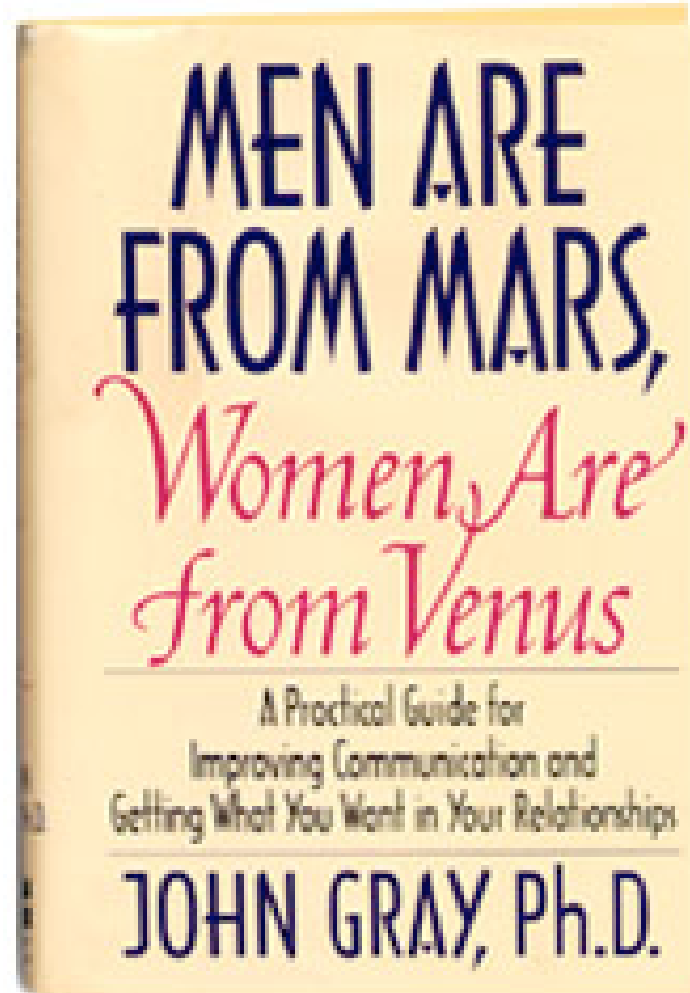
Siemens AG Campus - Munich, Germany

A photograph of a wedding couple walking away from the camera on a beach at sunset. The man is in a dark suit and the woman is in a white wedding dress. The ocean and sky are visible in the background.

When Eclipse met OSGi..



However...





Eclipse, before OSGi:

- Modules based on Classloader graph
- Versioned dependencies
- Side-by-side versions
- Inter-module glue



Then why OSGi???



Dynamic!





Some open heart surgery required



Now obsolete:

- ~~Modules based on Classloader graph~~
- ~~Versioned dependencies~~
- ~~Side by side versions~~

But *not*:

- Inter-module glue



The Extension Registry





Extensions are *Declarative*



plugin.xml

- Plug-in name & version
- Imports
- Classpath
- Extension points
- Extensions



plugin.xml

- ~~Plug-in name & version~~
- ~~Imports~~
- ~~Classpath~~
- Extension points
- Extensions



Extension Point Declaration

```
<extension-point  
  id="xyz"  
  name="..."  
  schema="...exsd"/>
```



Extension Point Schema

```
<element name="extension">
  <complexType>
    <sequence>
      <element ref="command" minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="point" type="string" use="required">
      <annotation>
        <documentation>

          </documentation>
        </annotation>
      </attribute>
      <attribute name="id" type="string">
        <annotation>
          <documentation>

            </documentation>
          </annotation>
        </attribute>
        <attribute name="name" type="string">
          <annotation>
            <documentation>

              </documentation>
            </annotation>
          <appInfo>
            <meta.attribute translatable="true"/>
          </appInfo>
        </element>
      </pre>
```



“The main problem of XSD is not that it gratuitously uses XML as its concrete syntax, but the fact that it is completely over-engineered for the problem it attempts to solve.”

– Erik Meijer, Microsoft



Extension Declaration

```
<extension  
  point="org.eclipse.ui.views">  
  <view  
    class="org...MyView"  
    icon="icons/view.gif"  
    id="..."  
    name="My View">  
  </view>  
</extension>
```



Content
according
to ext.
point
schema



Beware! This is an Extension *Point*:

`<extension-point...`



Beware! This is an *Extension*:

<extension point...



In case you missed it:

<extension-point...
<extension point...



Observation

- More complex meta-data
- Extensions need not have a **class** attribute
- 100% declarative (code-free) contributions



Querying the Extension Registry

```
IExtensionRegistry registry;
```

```
// ...
```

```
IExtensionPoint point =  
    registry.getExtensionPoint("xyz");  
IExtension[] extensions = point.getExtensions();
```



Querying the Extension Registry

```
for(IExtension ext : extensions) {  
    IConfigurationElement[] elements =  
        ext.getConfigurationElements();  
    // ...  
}
```



Querying the Extension Registry

```
for(IConfigurationElement element : elements) {  
    String name = element.getAttribute("name");  
    String icon = element.getAttribute("icon");  
    // ...  
}
```



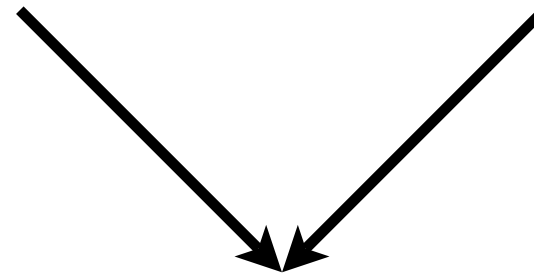
Getting the Extension Object

```
Object o = element.createExecutableExtension("class");
```

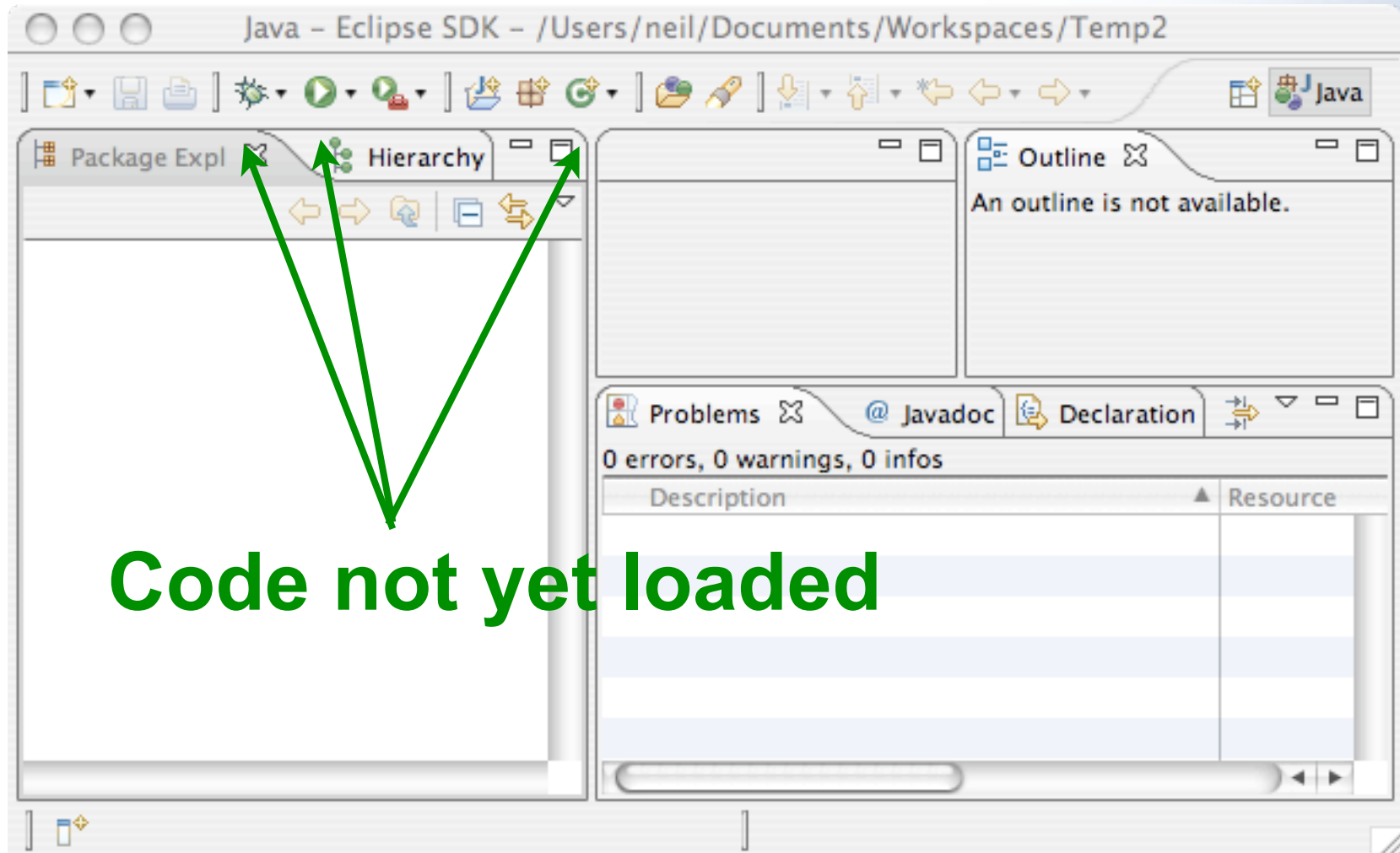


**Rich
Meta-data**

**Late
Instantiation**



Lazy Loading





Extension Discovery

Bundle Resolved



Has plugin.xml?



Merge with global DOM



Extension Registry Startup

Load Cached DOM

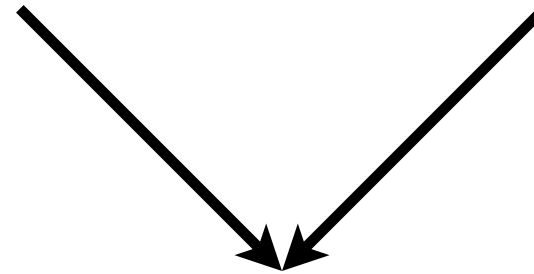


That's it!



**Lazy
Loading**

**Cached
State**



Very Fast Startup



Dynamic Extensions

- Simplest approach: re-query when needed
- Works for: Menus, Popups, etc
- Doesn't work for: Visible artifacts, Many non-GUI Requirements



Dynamic Extensions

```
public class MyExtensionChangeHandler implements IExtensionChangeHandler
{
    public void addExtension(IExtensionTracker tracker, IExtension ext) {
        // get elements
        tracker.registerObject(ext, object, IExtensionTracker.REF_WEAK);
    }

    public void removeExtension(IExtension ext, Object[] objects) {
        for(Object object : objects) {
            // clean up
        }
    }
}
```



Dynamic Extensions

```
IExtensionRegistry registry = null;
```

```
// ...
```

```
IExtensionPoint point = registry.getExtensionPoint("xyz");  
ExtensionTracker tracker = new ExtensionTracker(registry);  
IFilter filter = tracker.createExtensionPointFilter(point);  
tracker.registerHandler(new MyExtensionChangeHandler(), filter);
```



Small Problem

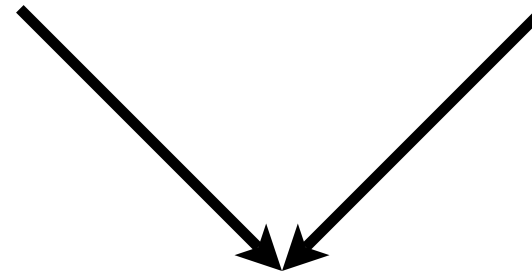
- Almost nobody does this!
- 99.99% of Eclipse plug-ins get the `IExtensionRegistry` service like this:

```
Platform.getExtensionRegistry();
```



**Complex
Code**

**Lack of
Exemplars**



**Many Eclipse Developers
*Ignore Dynamic Issues***



Example





Then why OSGi???



We're just not there yet



The Biggest Difference...

**“This is available, anybody can use it”
– OSGi Service**



The Biggest Difference...

**“I made this specially for you”
– Extension**



Service Registry

Classified Ad

Buffet

Dating Agency

Extension Registry

Visiting Salesman

Ordering from the Menu

Recruitment Agency



- PIANO LESSONS one-on-one coaching with experienced teacher £10/hr call 020 8123 4567
- SINGLE java.lang.Male with(&(tall=true)(money=lots)(sense_of_humour=great))
- HARLEY-DAVIDSON Road King, great condition, low mileage, has to be seen to be believed. £8,200 ono



- PIANO LESSONS one-on-one coaching with experienced teacher £10/hr call 020 8123 4567

- SINGLE java.lang.Male with(&(tall=true)(money=lots)(sense_of_humour=great))

- HARLEY-DAVIDSON Road King great condition, low mileage, has to be seen to be believed. £8,200 ono

NB: what if Male is only interested in Females?



Are Extensions Obsolete?



Principal Advantages of Extensions

Lazy loading

Rich metadata

Consumer Discrimination

Cached state, fast startup

Possible Services-based Replacement

Declarative Services

String properties can contain XML if desired

ServiceFactory (approximately)

???



State of Extension Registry  **The DOM**

State of Service Registry  **Opaque Internal State of Every Service**



Restoring Service Registry State

Deserialize service state



**All services must be
serializable**

Replay every
ServiceEvent, BundleEvent



Slower Startup



How Important is Start-up Time?

- For Eclipse SDK, *very*
- How about your application?



Conclusions

- Extensions are a good fit for Eclipse
- Extensions may be a good fit for parts of your application, if it is Eclipse-like
- Services are still better for most things



Questions?