DS, BP, EJB, CDI, WTF!? 

Graham Charters, IBM
Agenda

- Background
- Criteria
- Component models
- How to choose
Why?
Approach

- Provide some criteria to consider when making a component model choice
- Explain a little bit about the technologies
- Describe how each rates against the criteria

- Reasons aren’t always technical
- We probably won’t always agree

My beans are dynamic and use annotations.

Your beans smell!!
Technical Criteria

- Extra-bean definition (e.g. XML-based declarative model)
- Intra-bean definition (e.g. bean annotation-based declarative model)
- Extensibility
- Enterprise Qualities of Service
- Quality of Integration with OSGi
Non-technical criteria

- Open Standards
- Open Source implementations
- Choice of implementations
- Availability of tools
- Existing assets
- Existing skills
The technologies: DS, BP, EJB, CDI, WTF

- Declaratives Services (DS)
- Blueprint (not commonly known as BP)
- Enterprise JavaBeans (EJB)
- Contexts and Dependency Injection (CDI)
- World Taekwondo Federation (WTF)

There are a number of other component models, but I’ve focused on the ones I heard discussed most often. They’re also all based on open standards.
**Declarative Services**

<table>
<thead>
<tr>
<th>Criterion</th>
<th>DS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inspiration</strong></td>
<td>Popularity if DI frameworks, but designed for OSGi</td>
</tr>
<tr>
<td><strong>Approach</strong></td>
<td>Extra-bean definition (XML-based declarative model)</td>
</tr>
<tr>
<td></td>
<td>Build-time intra-bean definition (annotations compiled to XML)</td>
</tr>
<tr>
<td><strong>Extensibility</strong></td>
<td>No</td>
</tr>
<tr>
<td><strong>Enterprise QoS</strong></td>
<td>No</td>
</tr>
<tr>
<td><strong>OSGi Integration</strong></td>
<td>Designed by OSGi for OSGi. Includes Configuration Admin</td>
</tr>
<tr>
<td><strong>Open Standards</strong></td>
<td>OSGi Alliance</td>
</tr>
<tr>
<td><strong>Implementations</strong></td>
<td>Eclipse Equinox &amp; Apache Felix</td>
</tr>
<tr>
<td><strong>Tools</strong></td>
<td>BND &amp; BNDTools</td>
</tr>
</tbody>
</table>

```java
@Component
public class DiscountCalculatorImpl implements DiscountCalculator {
    private Logger logger;

    @Reference(policy=DYNAMIC,
                policyOption=GREEDY,
                cardinality=MANDATORY)
    void setLog( LogService log) { ... }
    void unsetLog( LogService log) { ... }
    void updatedLog( Map<String,?> ref ) { ... }
}
```

```xml
<component name="DiscountCalculator"
    implementation=
        class="com.acme.utils.impl.DiscountCalculatorImpl"/>
<service>
    <provide>
        interface="com.acme.utils.DiscountCalculator"/>
    </service>
<reference name="log"
    interface="org.osgi.service.log.LogService"
    bind="setLog" unbind="unsetLog"
    updated="updateLog"/>
</component>
```
<table>
<thead>
<tr>
<th>Criterion</th>
<th>Blueprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspiration</td>
<td>Spring &amp; Spring DM (standardized by SpringSource)</td>
</tr>
</tbody>
</table>
| Approach           | Extra-bean definition (XML-based declarative model, includes bean-to-bean wiring)  
                      | Intra-bean definition (Annotations – Apache Aries Open Source only)       |
| Extensibility      | Through XML namespaces – no runtime standard                              |
| Enterprise QoS     | Transactions and Persistence (Apache Aries), Security (WebSphere Application Server), Bus (CXF), Camel, etc… |
| OSGi Integration   | Some life-cycle compromises Config Admin (open source)                    |
| Open Standards     | OSGi Alliance Updates in pipeline                                          |
| Implementations    | Eclipse Gemini & Apache Aries                                              |
| Tools              | WebSphere Developer Tools (no charge), Rational Application Developer     |

```xml
<?xml version="1.0" encoding="UTF-8"?>
<blueprint ...>
  <service id="DiscountCalculatorBeanService"
           ref="DiscountCalculatorBean"
           interface="com.acme.utils.DiscountCalculator" />
  <bean id="DiscountCalculatorBean"
        class="com.acme.utils.impl.DiscountCalculatorImpl">
    <property name="logger" ref="loggerService" />
  </bean>
  <reference id="loggerService"
             interface="com.acme.utils.Logger" />
</blueprint>
```
Enterprise JavaBeans

<table>
<thead>
<tr>
<th>Criterion</th>
<th>EJB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspiration</td>
<td>Version 1.0 - 15 years old this week!</td>
</tr>
<tr>
<td>Approach</td>
<td>Inheritance &amp; XML bindings (pre-3.0). Intra-bean definition (bean annotation-based declarative model) (3.0 or later)</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Nothing formal (defer to CDI)</td>
</tr>
<tr>
<td>Enterprise QoS</td>
<td>Transaction, Security, Remoting, Persistence, Messaging</td>
</tr>
<tr>
<td>OSGi Integration</td>
<td>Core capabilities (proprietary)</td>
</tr>
<tr>
<td>Open Standards</td>
<td>JCP, OSGi specification in progress</td>
</tr>
<tr>
<td>Implementations</td>
<td>All JavaEE App Servers support EJB. OSGi integration in WebSphere Application Server, GlassFish, Apache Aries, Apache OpenEJB</td>
</tr>
<tr>
<td>Tools</td>
<td>WebSphere Developer Tools (no charge), Rational Application Developer, Eclipse WTP</td>
</tr>
</tbody>
</table>

```java
@Stateless
@Local(DiscountCalculator.class)
public class DiscountCalculatorImpl
    implements DiscountCalculator {

    @Resource(lookup="osgi:service/Logger")
    private Logger logger;

    public double discount() {
        ...
    }
```
## Contexts and Dependency Injection

<table>
<thead>
<tr>
<th>Criterion</th>
<th>CDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspiration</td>
<td>JBoss Seam, Google Guice &amp; Spring</td>
</tr>
<tr>
<td>Approach</td>
<td>Intra-bean definition (annotation-based life-cycle and DI model), Extra-bean opt-in beans.xml with additional config (e.g. extensibility)</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Interceptors, Decorators &amp; Portable Extensions</td>
</tr>
<tr>
<td>Enterprise QoS</td>
<td>None. Leverage CDI in other component models (e.g. EJB)</td>
</tr>
<tr>
<td>OSGi Integration</td>
<td>Core capabilities (Open Source)</td>
</tr>
<tr>
<td>Open Standards</td>
<td>JCP, OSGi Alliance in progress</td>
</tr>
<tr>
<td>Implementations</td>
<td>Weld-OSGi, Pax-CDI, Fighterfish</td>
</tr>
<tr>
<td>Tools</td>
<td>EE Vendors tools</td>
</tr>
</tbody>
</table>

```java
@Named
@RequestScoped
@OSGiComponent
public class DiscountCalculatorImpl
implements DiscountCalculator {

    @Inject @OSGiService
    StaticLogger logger;

    public double discount() {...}

    ...
}
```
Some questions to ask when choosing

- What existing skills does my team have?
  - Choose the closest match (e.g. Spring? Consider Blueprint)

- What existing assets do I have available?
  - Choose the closest match

- Do I need Enterprise capabilities?
  - If “yes”, consider Blueprint or EJB, if “no”, consider DS

- Do I want something based on a standard?
  - If “yes”, familiarize yourself with the scope and direction of existing standards

- Do I need something based on open source?
  - Apache Felix, Eclipse Equinox, Apache Karaf, Eclipse Gemini, Apache Aries, Eclipse Virgo, ....the list goes on.....

- What are the capabilities of my target runtime?
  - If a component model isn’t supported, there’s little point choosing it

- What are the capabilities of my tool chain?
  - If a component model isn’t supported, you’re asking for pain

- What if I use more than one component model?
  - Leverage OSGi services as the common integration point

No one component model to rule them all