The newest OSGi specs: CDI integration and Release 8
Carsten Ziegeler & David Bosschaert
About David

- Senior Computer Scientist @ Adobe
- Member of the Apache Software Foundation
- OSGi Enterprise Expert Group co-chair
About Carsten

- Principal Scientist @ Adobe
- Member of the Apache Software Foundation
- OSGi Expert Groups + Member of the Board
Agenda

- The OSGi CDI integration spec
- New R8 specs under way
  - Messaging
  - Connect
  - Condition Service
  - Feature Model
  - ... more ...
OSGi Specification Process

- Requirements, Use Cases -> RFP
- Technical Specification -> RFC
- Reference Implementation
- Compatibility Testsuite
- Spec Chapter
- Books: Core, Compendium, Enterprise
OSGi CDI Integration
OSGi CDI Extender

- Connect CDI with OSGi
- Supports CDI Annotations and Extensions
- Constructor, Field, Method Injection
- Supports Component Property Types
Why CDI Extender?

- Declarative Services are great, so why?
- CDI Extender Use Cases
  - Existing Code
  - Enhanced annotation processing
  - Private services/beans
OSGi Service Model

- Service Registry
  - Publish/find/bind

- Service Scopes
  - Singleton
  - Bundle
  - Prototype
Component Container Integration

Declarative Services

CDI Integration

Framework API

OSGi Service Registry

Others like iPojo, Dependency Manager, Blueprint ....
Register a Service using CDI

```java
@Bean
@Service(value = GameController.class)
@ServiceDescription
    ("This is the core game component")
public class GameControllerImpl
    implements GameController {
}
```
@Bean
@Service(Servlet.class)
public class GameServlet extends HttpServlet {

@Inject @Reference
private GameController game;

@Inject
private HighscoreService highscore;
Service Properties

@Bean
@SingleComponent
@AppServletContext.ContextConfig
public class AppServletContext {

    @Retention(RetentionPolicy.RUNTIME)
    @Target(ElementType.TYPE)
    @BeanPropertyType
    public @interface ContextConfig {
        String service_name() default AppServletContext.NAME;
        String service_path() default "/guessinggame";
    }
}
Declarative Services vs CDI in OSGi

<table>
<thead>
<tr>
<th></th>
<th>CDI</th>
<th>DS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declare a service</td>
<td>@Service (+@Bean)</td>
<td>@Component</td>
</tr>
<tr>
<td>OSGi Reference</td>
<td>@Inject @Reference</td>
<td>@Reference</td>
</tr>
<tr>
<td>Properties</td>
<td>@BeanPropertyType on new annotation class Annotation on class</td>
<td>Properties in @Component</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New annotation class Usage of class</td>
</tr>
</tbody>
</table>

While the annotations share the name they are in different packages
How to use CDI?

- Apache Aries CDI Extender Implementation
- Bnd Tooling >= 4.2.0
Messaging (RFP 192)
Messaging RFP 192

- Remote async messaging API for OSGi
- Publish-Subscribe
- Point-to-point
- Lightweight
- QoS & Intents
  - require and advertise capabilities
Connect (RFP 196)
Reboot OSGi Connect

- Connect OSGi to the „outside"
- Popular Application Frameworks
- JPMS
- JARs on the classpath
- Native compilation
Condition Service (RFC 242)
Condition Service

- When is your asynchronous system fully ready?
- Generally very app-specific
- Sometimes multiple levels
- Introducing:
  
  `org.osgi.service.condition.Condition`

service interface
Waiting for a condition

- A client can just listen for a condition:
  
  ```java
  @Reference(target="(osgi.condition.id=mycondition)")
  
  Condition ready
  ```

- Or default to the predefined true condition:
  
  ```java
  (osgi.condition.id=true)
  
  ... change via config by setting target property
  ```
Registering a condition

- Just register a Condition service (custom ID)
- Configuration-based:

```json
"osgi.condition.factory~systemready" : {
    "osgi.condition.identifier" : "application.ready",
    "osgi.condition.match.all" : [
        "(&(objectClass=javax.servlet.Servlet)(role=main))",
        "(db.connection.url=*)"
    ],
    "osgi.condition.match.none" : [
        "(&(objectClass=javax.servlet.Servlet)(role=loading))"
    ]
}
```
Features (RFC 241)
Features RFC 241

- Design and work with entities
  - Larger than individual bundles
- Configuration
- Additional metadata
- A design artifact
  - can be mapped to runtime implementations (Sling/Karaf/Eclipse etc...)
Features – building a system

Easy!
Type-Safe Events (RFC 244)
Type-Safe Events RFC 244

- Send asynchronous **Typed Events**
- Locally in JVM
- Events are OSGi DTOs
  - Primitive types / wrappers
  - Strings
  - Collections / arrays
  - DTOs
Type-safe Events (2)

• Sending events

    TypeSafeEventBus bus = ... from Service Registry ...
    MyDTO event = new MyDTO();
    event.myField = ... // set data in DTO
    bus.deliver(event);

• Receiving events

    @Component
    @EventType(MyDTO.class)
    class MyEventHandler implements TypedEventHandler<MyDTO> {
        public notify(String topic, MyDTO event) {
            // handle the event
        }
    }
Others
Other work

- RFP 190 Resource Encoding for Java Modules
- RFP 191 Microservice Architecture
- RFP 195 Actor Runtime

... and more: https://github.com/osgi/design
Q & A
EVALUATE THE SESSIONS

Sign in and vote using the conference app or eclipsecon.org
References OSGi

- OSGi RFPs and RFCs
  - https://github.com/osgi/design
- OSGi Specifications
  - https://osgi.org/specification/osgi.core/7.0.0/
  - https://osgi.org/specification/osgi.cmpn/7.0.0/
  - https://osgi.org/specification/osgi.enterprise/7.0.0/
References Sample Code

- CDI Example
  - https://github.com/cziegeler/samples.guessinggame.cdi
- CDI Launcher
  - https://github.com/cziegeler/features.cdi
Images

- From stock.adobe.com:
  - Karton voll mit Computerschrott By peno - penofoto.de
  - Modern PC isolated on white background. 3D illustration By Destina
  - Computer spare parts store By Vlad Kochelaevskiy