Using OSGi R7 in AEM

Carsten Ziegeler, David Bosschaert, Karl Pauls – Adobe

Carsten Ziegeler

- Principal Scientist @ Adobe
- Member of the Apache Software Foundation
- PMC Member of Apache Felix and Sling
- OSGi Expert Groups and Board member

David Bosschaert

- Senior Computer Scientist @ Adobe
- Member of the Apache Software Foundation
- OSGi Enterprise Expert Group (EEG) co-chair

Karl Pauls

- Computer Scientist @ Adobe
- Member of the Apache Software Foundation
- PMC Member of Apache Felix and Sling (VP Apache Felix)
- Co-Author OSGi in Action

Outline

- What is new in OSGi R7
  - OSGi R7 Highlights
  - OSGi R7 and AEM 6.4
- Java 9 support
  - Java 9 and OSGi R7
  - Anticipating Java 11 support in AEM

What is new in OSGi R7

OSGi Release 7

- Core and Compendium R7 released in April 2018
- Major release since R6 (April 2015)
- Improvements
- New Specifications
- Developer Experience
- Most specifications are implemented at Apache Felix and Apache Aries

Declarative Services R7 Highlights

- Improved activation
  - Activation objects assigned to fields
  - Constructor injection
- Component Property Type annotations

Declarative Services – Field Activation Objects

```
private Config configuration;

@Activate
protected void activate(final Config config) {
                                this.configuration = config;
}
```

```
@Activate
private Config configuration;


@Activate
private BundleContext bundleContext;
```

Constructor Injection

```
@Component
public class MyComponent {

    @Activate
    public MyComponent(
                       BundleContext bundleContext,

                       @Reference EventAdmin eventAdmin,

                       Config config,

                       @Reference List<Receivers> receivers) {
    // store in final fields
}
```

Component Property Type Annotations

- Simplify Component Configuration

```
@ComponentPropertyType
public @interface ServiceDescription {
    String value();
}
```

```
@Component
@ServiceDescription("Best service in the world")
public class MyComponent {
}
```

Developer Experience – Bundle Annotations I

- *All* manifest entries through annotations
- Package Exports and Versioning
  - @Version, @ProviderType, @ConsumerType
  - **@Export**

Developer Experience – Bundle Annotations II

@Capability, @Requirement, @Header

```
@Requirement(namespace="osgi.implementation",
        name="osgi.http",
        version="1.1.0")

@Header(name=Constants.BUNDLE_CATEGORY,
        value="assets")
```

Infer Requirements from Feature Usage

- Requiring Declarative Services:

```
@Requirement(namespace = ExtenderNamespace.EXTENDER_NAMESPACE,
       name = ComponentConstants.COMPONENT_CAPABILITY_NAME,
       version = ComponentConstants.COMPONENT_SPECIFICATION_VERSION)
```

- Infered by using DS annotations:

```
@Component
```

Infer Requirements from Feature Usage

```java
@Requirement(namespace = ExtenderNamespace.EXTENDER_NAMESPACE,
        name = ComponentConstants.COMPONENT_CAPABILITY_NAME,
        version = ComponentConstants.COMPONENT_SPECIFICATION_VERSION)
public @interface RequireServiceComponentRuntime {


@RequireServiceComponentRuntime
public @interface Component {
    ...
}
```

OSGi R7 Highlights - Web

- Http Whiteboard
  - Improvements (Global Filters)
  - Component Property Types
- JAX-RS
  - A whiteboard model for JAX-RS

Http Whiteboard Annotations

```
@Component(service = Servlet.class)

@ServiceRanking(200)
@ServiceDescription("Best Servlet in the World")

@HttpWhiteboardServletPattern("/game")
@HttpWhiteboardContextSelect(
  "(" + HttpWhiteboardConstants.HTTP_WHITEBOARD_CONTEXT_NAME
        +"=" + AppServletContext.NAME + ")")
public class GameServlet extends HttpServlet {
```

JAX-RS with DS

```
@Component(service = TestService.class)
@JaxrsResource
@Path("service")
public class TestService {

    @Reference
    private GameController game;

    @GET  @Produces("text/plain")
    public String getHelloWorld() {
        return "Hello World";
    }
}
```

JAX-RS Support

- Get, Post, Delete with Parameters
- Application support
- JAX-RS extension support (Filters, Interceptors, etc)
- Annotations for Declarative Services

D

OSGi R7 Highlights

- Configurator and Configuration Admin
  - Configuration Resources
  - Improved factory configuration handling
  - Configuration Plugin improvements

Reusable Configuration Format

```
{
  "my.special.component" : {
    "some_prop": 42,
    "and_another": "some string"
  },
  "and.a.factory.cmp~foo" : {
    ...
  },
  "and.a.factory.cmp~bar" : {
    ...
  }}
```

OSGi R7 Highlights - Basics

- Converter
  - Object conversion
- Promises and Push Streams
  - Asynchronous programming model
  - Streams

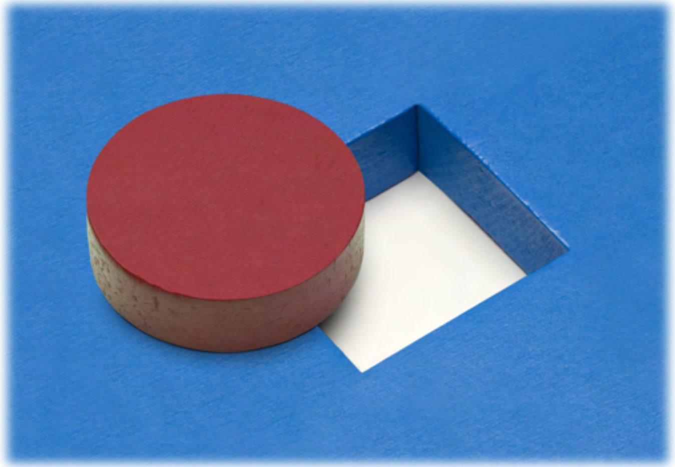
Object Conversion

- Wouldn't you like to convert anything to everything?

- Convert
  - Scalars, Collections, Arrays
  - Interfaces, maps, DTOs, JavaBeans, Annotations

- Need predictable behaviour – as little implementation freedom as possible
  - Can be customized

Implementation in Apache Felix



Using the Converter

```
Converter c = Converters.standardConverter();

// Convert scalars
int i = c.convert("123").to(int.class);
UUID id = c.convert("067e6162-3b6f-4ae2-a171-2470b63dff00").to(UUID.class);

List<String> ls = Arrays.asList("978", "142", "-99");
short[] la = c.convert(ls).to(short[].class);
```

Convert untyped maps into typed information with defaults

```
// Convert map structures
@interface MyAnnotation {
  int refresh() default 500;
  String temp() default "/tmp";
}

Map<String, String> myMap = new HashMap<>();
myMap.put("refresh", "750");
myMap.put("other", "hello");

MyAnnotation myAnn = converter.convert(myMap).to(MyAnnotation.class)

int refresh = myAnn.refresh();  // 750
Strine temp = myAnn.temp();     // "/tmp"
```

OSGi Promises

Javascript-style promises

- Asynchronous chaining
- Very simple programming model

Promises can be used outside of OSGi framework

```
public class PromisesTest {

    public static void main(String... args) {
        System.out.println("Starting");
        takesLongToDo(21)
            .then(p -> intermediateResult(p.getValue()))
            .then(p -> finalResult(p.getValue()));
        System.out.println("Async computation kicked off");
    }

    public static Promise<Long> intermediateResult(Long l) {
        System.out.println("Intermediate result: " + l);
        return takesLongToDo(l * 2);
    }

    public static Promise<Void> finalResult(Long l) {
        System.out.println("Computation done. Result: " + l);
        return Promises.resolved(null);
    }

    public static Promise<Long> takesLongToDo(long in) {
```

Push Streams

Like Java 8 streams, but data is pushed

For event-based data, which may or may not be infinite

- Async processing and buffering
- Supports back pressure
- Mapping, filtering, flat-mapping
- Coalescing and windowing
- Merging and splitting

Example:

- Humidity reader/processor stream
  - **sends an alarm when over 90%**

Implementation in Apache Aries

Asynchronous Push Streams example

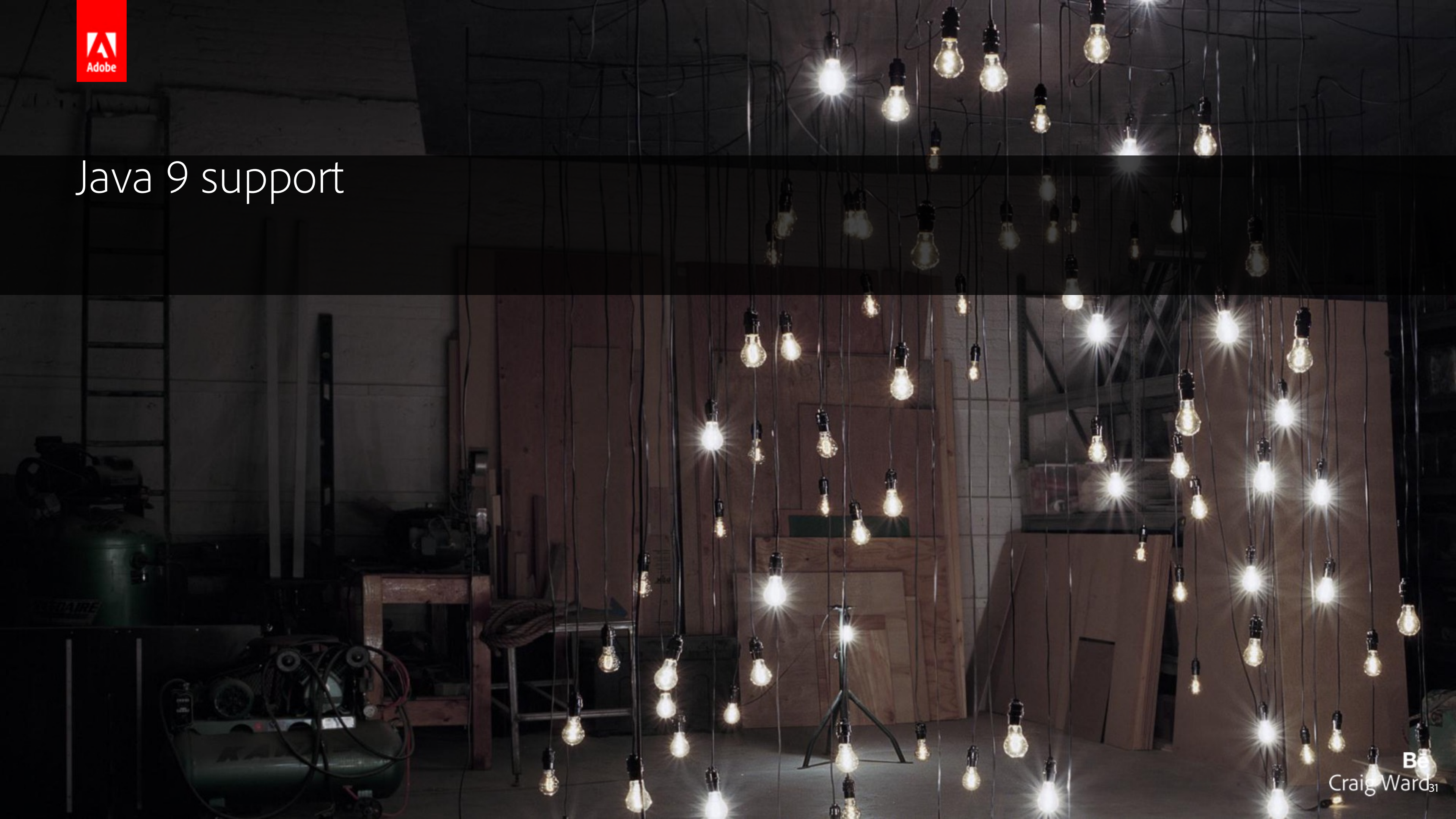
```
PushStreamProvider psp = new PushStreamProvider();

try (SimplePushEventSource<Long> ses = psp.createSimpleEventSource(Long.class)) {
  ses.connectPromise().then(p -> {
    long counter = 0;
    while (counter < Long.MAX_VALUE && ses.isConnected()) {
      ses.publish(++counter);
      Thread.sleep(100);
      System.out.println("Published: " + counter);
    }
    return null;
  });

  psp.createStream(ses).
    filter(l -> l % 2L == 0).
    forEach(f -> System.out.println("Consumed even: " + f));
```

Additional OSGi R7 Highlights

- Cluster Information
  - Support for using OSGi frameworks in clustered environments.
- Transaction Control
  - An OSGi model for transaction life cycle management.

OSGi R7 and AEM

- AEM 6.4 contains
  - Declarative Services, Configuration Admin, Http Whiteboard
- Additional installation possible
  - Converter, Promises, Push Streams
- Potentially installable
  - Configurator, JAX-RS
- Tooling
  - Apache Felix maven bundle plugin 4.x
  - Bnd maven plugin 4.x



Java 9 support

JPMS – Java Platform Module System

- Modularized JDK
  - 24 modules (e.g., logging, xml, desktop, rmi,…)
  - 6 modules deprecated for removal
    - java.activation, java.corba, java.transaction, java.xml.bind, java.xml.ws, java.xml.annotation
    - Not available by default (needs –add-modules)
    - Not available anymore at all in java11
  - Deprecation of Unsafe

JPMS – Java Platform Module System

- Module system for jvm based applications
  - Modulepath along side classpath
  - Meta-data for exports, requires, and services (module-info.java)
  - Module level accessibility
    - Public no longer public (only public and exported and readable is accessible)
    - Includes reflection
  - No split packages
- ModuleLayer for recursive use cases
- Allows developers to build custom platforms based only on the required modules (via jlink tool)

Multi-Release JAR

- New type of JAR called multi-release JAR
  - Allows the JAR to support multiple major Java versions
- In a nutshell
  - Simple JAR with „Multi-Release: true“ in Manifest
  - Can provide version dependent resources in
    META-INF/versions/N (for N>=9)
  - Highest matching versioned resource overrides

Java 9 and OSGi R7

Java imports

- Until now, osgi.ee was used to define required Java version
- Modularized Java enables to build custom platforms
  - Includes java.* packages
- Subsequently,
  - OSGi R7 now allows imports for java.*
  - osgi.ee should only be used for bytecode level
- Java exports still only possible by the system bundle
  - Effectively, still bootdelegated
- System packages will now be calculated based on available modules

Multi-Release JAR files in OSGi

- OSGi R7 adds support for multi-release JAR files
  - An OSGi bundle file can be a multi-release JAR
  - Bundle class path entries can be multi-release JAR files
- R7 Framework supports supplemental manifest files
  - Supplement “Import-Package“ and “Require-Capability“ for different versions
  - Via OSGI-INF/MANIFEST.MF in the versioned directories

    e.g.:

    META-INF/versions/9/OSGI-INF/MANIFEST.MF

Supporting R6

- OSGi R6 prohibits bundles from importing java.*
- Bundles that must work on OSGi R6 and earlier should:
  - Not import the java.* packages in the main Manifest
  - Package the bundle as a multi-release JAR and import java.* packages in supplemental manifests
- R6 frameworks will ignore supplemental manifests
- R7 frameworks will use them and they are only relevant starting with java >= 9

Tooling

- Maven has no support for Multi-Release JAR
  - Workarounds possible
  - https://maven.apache.org/plugins/maven-compiler-plugin/multirelease.html
- BND doesn‘t support Multi-Release JAR
  - https://github.com/bndtools/bnd/issues/2227
- BND doesn‘t support java.* dependencies
  - https://github.com/bndtools/bnd/issues/2507

Summary and Outlook

- True interoperability between JPMS and OSGi still not possible as OSGi framework has to be on the classpath for now (and not on the module classpath)
- OSGi R7 improves using OSGi on JPMS
  - Runtime discovery of packages together with java.* imports allows developers to build custom runtimes
  - Multi-release JAR supports provides path for R6 BC

Anticipating Java 11 support in AEM

What can you do today

- Prepare bundles for java11
  - Consider adding java.* imports
    - Packaged as multi release jars
    - (might want to wait until tooling catches up)
  - Don't rely on bootdelegated packages
  - Don't use packages not in java11
    - Especially not javax.rmi and org.omg.*
- AEM 6.4 should start-up on java9

OSGi R7 and beyond...

OSGi R7 and beyond…

- Upcoming OSGi R7 Enterprise release
  - CDI - Context and Dependency Injection support OSGi
  - Proposed final draft will be released 22nd of October 2018
- R8 Plans
  - App Packaging and Java 11 JPMS
  - Realtime OSGi
  - Industry 4.0
  - Microprofile I/O

OSGi R7 and beyond...

- Read more about the features added in R7 in the OSGi R7 Highlights Blog Series
  - https://blog.osgi.org/2018/09/osgi-r7-highlights-blog-series.html

Questions?

Image acknowledgements

Images licensed from https://stock.adobe.com

- Round Block Square Hole By pixelrobot
- Tabby cat drinks water from the tap By Stefano Garau