RESTful Services and Distributed OSGi

Friends or Foes

Roland Tritsch
Director PS FUSE
ACCU 2009 - Apr., 23rd - Oxford (UK)

Why is this relevant?

Users and usage ...

- IBM's WebSphere
- Oracle's Weblogic
- Progress's FUSE
- Red Hat's JBoss
- SpringSource's Application Platform
- Sun Microsystem's GlassFish Enterprise Server

- Linked-In, SAP NetWeaver, Telco, ... :)

Agenda

- Introduction to OSGi and REST
  - The consortium, the standards, the users
  - Why to use it and how to use it
- Exposing WebServices from an OSGi container
  - To REST or not to REST that is here the ...
- Wrap and Summary
  - Current challenges and future developments

The History

- OSGi originally stood for “Open Services Gateway initiative”
  - An initiative focused on deploying Java solutions into “residential gateways” for smart homes and building controls
  - The OSGi alliance was founded in 1999 to promote wide scale adoption of OSGi technology
- OSGi tackles the problem of deploying and administering Java “modules” (aka “bundles”)
  - Lifecycle - How to load, start, and stop Java bundles without shutting down the JVM

The Benefits

My personal hit list ...

- Mobile to Mainframe
- Managing the “CLASSPATH hell”
- With great tools/tooling around it

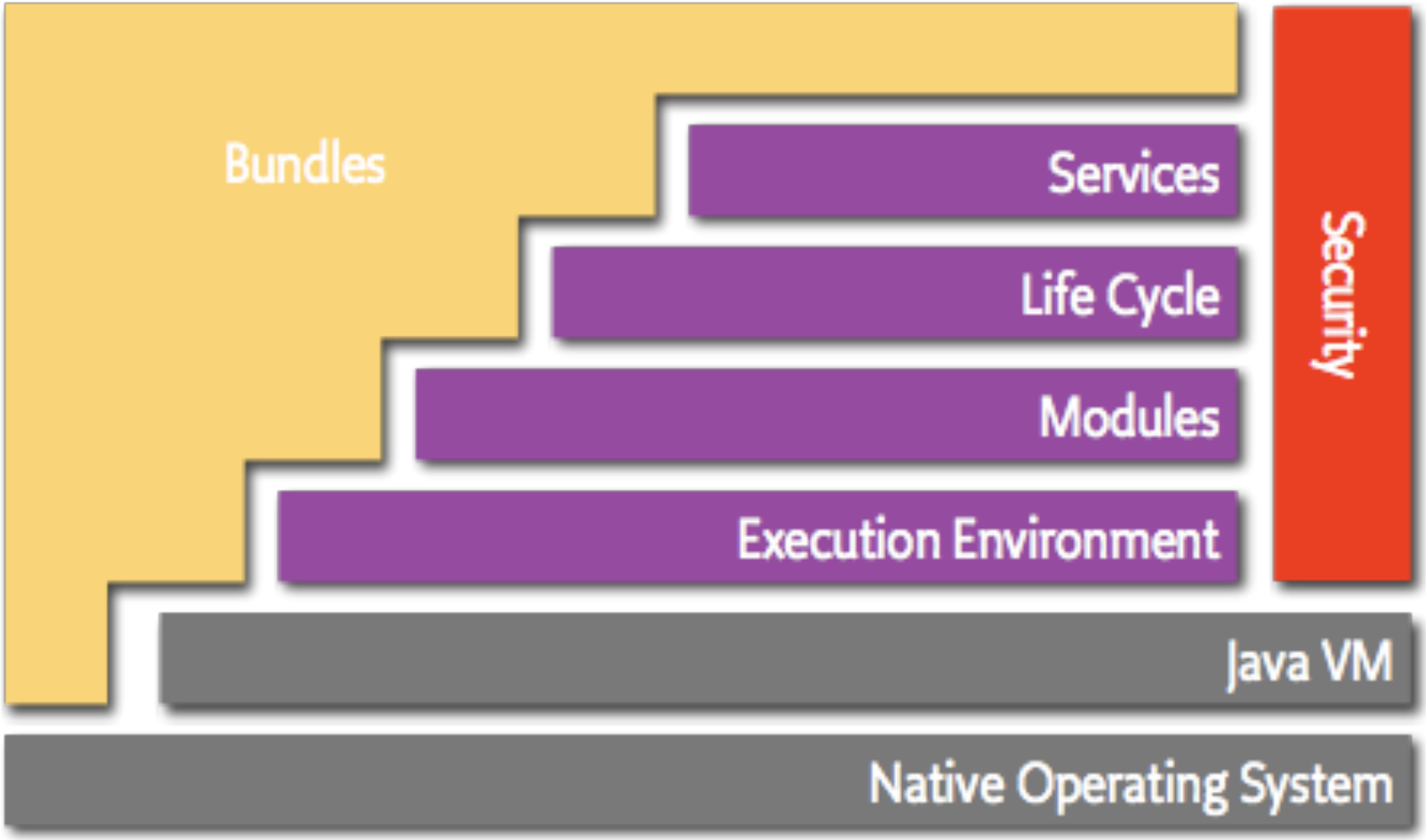
The OSGi Alliance

- The OSGi alliance is made up of over 40 members. Including ...
  - IBM, Progress, RedHat, SpringSource, Gigaspaces, Sun Microsystems ...
- Specifications are created by expert groups
  - Core Platform Expert Group (CPEG)
  - Vehicle Expert Group (VEG) - OSGi in the automotive industry
  - Mobile Expert Group (MEG) - OSGi in mobile telecommunications

The OSGi Alliance (cont.)

- Specifications are created by expert groups
  - Residential Expert Group (REG) - OSGi in consumer and residential applications
  - Enterprise Expert Group (EEG) - OSGi in enterprise IT infrastructure applications
    - Distributed OSGi/RFC 119 describes how to allow remote invocations between OSGi containers
      - A reference implementation is based on Apache CXF

OSGi - Architecture Overview



Source: www.osgi.org

OSGi - Key/Core Concepts - Bundles

- A bundle is a Java archive (JAR) with some meta-data
  - The meta-data is provided in plain-text in the META-INF/MANIFEST.MF file.
- Bundle meta-data includes the following
  - Bundle-Name, Bundle-Symbolic-Name, Bundle-Version, Export-Package, Import-Package
- Only those packages matching java.* are imported by default
  - all other packages must be imported explicitly

OSGi - Key/Core Concepts - Services

- Services in OSGi are Java objects, invoked using local method invocations
  - Used to allow dynamic (re)use of code
- The Distributed OSGi specification (EEG RFC 119) extends this concept to allow distributed communication.
  - A service in one OSGi framework instance could invoke on another service deployed in a different OSGi framework instance.
  - Services use additional OSGi properties to mark a service as “remote”.

OSGi - Key/Core Concepts - Class Loading

- Using graph-based class-loading with versioned bundles means:
  - The same JVM can host numerous bundles …
  - … including different versions of the same bundle, …
  - … that can share and re-use classes, …
  - … with no runtime class-loading conflicts, …
  - … in a standardized manner.
- This is a major contribution of OSGi, in that it removes much of the risk associated with different JARs in the same JVM container

OSGi - Implementations

The "not-complete" list ... :)

- Apache Felix (from Apache Software Foundation)
  - Distributed under ASL 2.0
  - http://felix.apache.org/
- Equinox (from Eclipse Foundation)
  - Distributed under the Eclipse License
  - http://eclipse.org/equinox
- Knoplerfish (knoplerfish.org)
  - Distributed under the Knoplerfish License
  - http://knoplerfish.org

REST - Introduction

- JAX-RS/JSR311 provides Java language support for RESTful services
  - For more information on REST, see Roy Fielding's Ph.D. thesis:
    - http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
- The approach is ideal for services that will be consumed by rich, browser- based internet client technologies like JavaScript and Google Web Toolkit (GWT)

REST - Key/Core Concepts

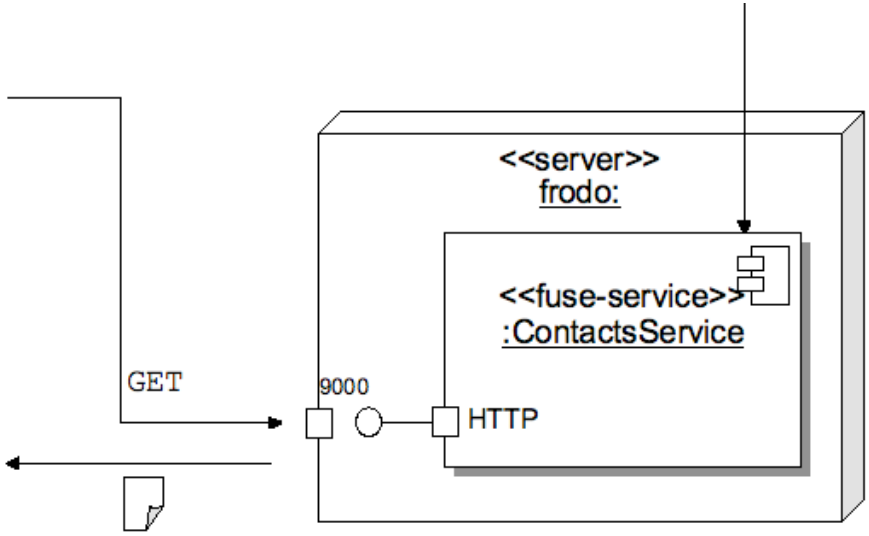
- Approach:
  - Annotate - your Java service.
  - Deploy - in Spring Framework, Tomcat, J2EE, or standalone.
  - Consume – e.g. from AJAX clients.

ContactsService.java

```
@Path("contactservice")
interface ContactsService {

  @GET

  @Path("/contacts/{id}")

  public Contact getContact(
    @PathParam("id") int id);

}
```

http://frodo:9000/contacts/123 Go >

```
<ns4:Contact>
  <firstName>Ade</firstName>
  <lastName>Trenaman</lastName>
  <company>IONA Technologies</company>
  <title>Principal Consultant</title>
  <email>adrian.trenaman@iona.com</email>
  <mobile>+353-86-6051026</mobile>
```



The REST Interface

- No formal interface definition language (WSDL, IDL) is used
  - However, XSD is often used for data definition.
- A service's "parameters" are passed via payload and URL
  - http://localhost:9000/contacts/007
  - Apache CXF supports multiple payloads, including XML and JSON

The REST Interface (cont.)

- Services make use of a natural mapping from HTTP verbs to CRUD operations.
  - POST: **Create** a new item of data on the service.
  - GET: **Retrieve** data from the service.
  - PUT: **Update** data on the service.
  - DELETE: **Delete** data from services.

The REST Interface - “HTTP” tunneling

- Some client-side tooling only supports GET and POST
- Instead of using PUT and DELETE, you can encode the operation for a create, delete or update into the URL of a HTTP POST
  - POST http://frodo/deleteCustomer/{id}
  - Don’t use this approach with HTTP GET, which should obey “read-only” semantics

Testing RESTful Services

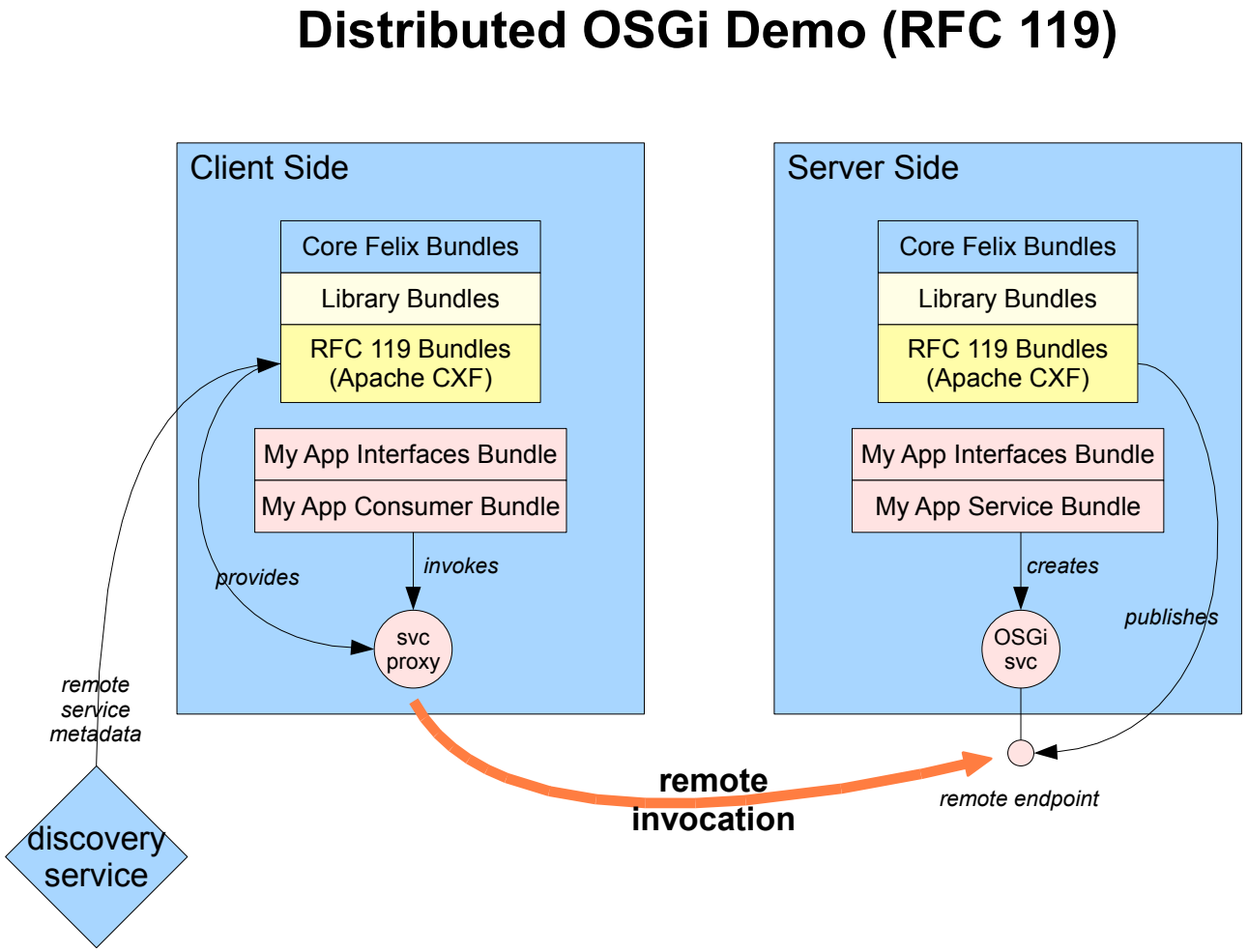
- You can test your REST services by simply pointing a browser at the URL
  - This will implicitly perform a GET on your service
- Alternatively, you can use command-line tools like wget or curl

Agenda

- Introduction to OSGi and REST
  - The consortium, the standards, the users
  - Why to use it and how to use it
- Exposing WebServices from an OSGi container
  - To REST or not to REST that is here the ...
- Wrap and Summary
  - Current challenges and future developments

Distributed OSGi

Exposing and Consuming WebServices



Source: http://blogs.iona.com/newcomer/OSGiCommunityEvent-DistOSGi.pdf

Exposing WebServices and ...

```
public class Activator implements BundleActivator {
  private ServiceRegistration sr;

  public void start(BundleContext context) throws Exception {
    Dictionary props = new Hashtable();
    props.put("osgi.remote.interfaces", "*");
    sr = context.registerService(
      AuctionService.class.getName(),
        new AuctionServiceImpl(), props);
  }

  public void stop(BundleContext context) throws Exception {
    sr.unregister();
  }
}
```

Source: http://coderthoughts.blogspot.com/2009/02/distributed-osgi-simple-example.html

... consuming WebServices

```
public class Activator implements BundleActivator {
  private ServiceTracker st;

  public void start(final BundleContext bc) throws Exception {
    st = new ServiceTracker(bc,
      AuctionService.class.getName(), null) {
      @Override
      public Object addingService(ServiceReference reference){
        Object svc = bc.getService(reference);
        if (svc instanceof AuctionService) {
          printServiceInfo((AuctionService) svc);
        }
        return super.addingService(reference);
      }
    };
    st.open();
  } ...
```

Source: http://coderthoughts.blogspot.com/2009/02/distributed-osgi-simple-example.html

Making D-OSGi and REST work together

- Publishing RESTful endpoints from an OSGi container is possible right now
  - Package you annotated JAVA classes in a bundle and implement a suitable Activator() to start endpoint
- Consuming RESTful services from an OSGi container is less straight forward
  - You can always consume them manually with your own RESTful client side stack, but they will not look like OSGi services
- D-OSGi wants to make distribution seamless

Making D-OSGi and REST work together

- The JAX-RS/JSR311 spec does not define a client side API for RESTful services ...
- ... but Jersey and Apache CXF do have such APIs, means ...
- ... both of these implementations are probably suitable to be plugged-in D-OSGi to allow OSGi containers to get seamless access to RESTful services

Agenda

- Introduction to OSGi and REST
  - The consortium, the standards, the users
  - Why to use it and how to use it
- Exposing WebServices from an OSGi container
  - To REST or not to REST that is here the ...
- Wrap and Summary
  - Current challenges and future developments

References

- The OSGi Alliance - http://www.osgi.org
- REST - http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
- Distributed OSGi
  - Reference Implementation - http://cxf.apache.org/distributed-osgi.html

Current Situation and Next Steps

- OSGi is on the rise
- With the efforts around D-OSGi, (RESTful) WebServices are first class citizens in the OSGi world
  - OSGi will become the de-facto deployment environment for (RESTful) WebServices
- The next steps could be to extend D-OSGi to include support for async messaging

Questions

Questions

Thank You!