Testing Blueprint Routes with Pax Exam and Karaf

Bryan Saunders

Agenda

- Pax Exam
- Pax Exam Karaf Container
- Pax Concepts
- Testing Blueprint Routes
  - Maven Dependencies
  - Configuring the Container
  - Writing Testable Routes
- Demo
- Resources

Pax Exam

Pax Exam

- In-Container Testing Framework for OSGi, Java EE, and CDI
- Uses a Test Driver and a Test Container
  - Test Driver launches the OSGi framework and the system under test
  - Builds On-The-Fly bundles from Test Cases and Injects them into the Container, Called a Test Probe
- JUnit and TestNG Drivers
- Drivers are Annotation Based

Pax Exam

- Two Types of Containers, Native and Forked
  - Native runs in the same VM as the Test Driver
  - Forked runs in a separate VM from the Test Driver
- Supports multiple Strategies for Restarting and Reusing the Running OSGi Framework for each Test
- Supports All Major OSGi Frameworks
  - Equinox, Felix, Knoplerfish

Pax Exam Karaf Container

Pax Exam Karaf Container

- Eases Integration Testing with Pax Exam and Karaf
- Provides an Actual Karaf Container for Testing
- Supports any Karaf Based Distribution (Fuse, Service Mix, Geronimo)
- Maintained as Official Pax Exam modules in Pax Exam 3.1.0
- Added Karaf Specific Support
  - KarafDistributionOption.*

Pax Exam Concepts

Container Configuration

- Controls the Host Container
- Determines the Set of Bundles and Features provisioned to the Container
- Builds and Configures the Container Environment
- Multiple Methods of Specifying Configuration Options
  - OSGi typically uses one or more methods annotated with `@Configuration` with the return type `Option[]`

Probe

- Artifact added to the Container for Testing
- Probe is Created on the Fly
  - Uses Pax Tinybundles
- Contains the Current Test Classes and All Classes/Resources under the Same Root
- Can be Configured inside the Test Class if needed

Testing Blueprint Routes

Writing Testable Routes

- Anything that needs to Change during a Test should be Externalized
  - All Endpoints
  - Configuration Values
  - Header Names
  - Etc…
- Integrations with External Resources should be Mock able
  - Database Connections
  - Other Messaging Brokers
  - Etc…

Maven Dependencies

- The Following Maven Dependencies are Necessary
  - Testing
    - junit:junit:4.10
    - org.ops4j.pax.exam:pax-exam:3.4
    - org.ops4j.pax.exam:pax-exam-inject:3.4
    - org.ops4j.pax.exam:pax-exam-junit4:3.4
    - org.apache.camel:camel-test:2.12
    - org.apache.camel:camel-test-blueprint:2.12
  - Container
    - org.ops4j.pax.exam:pax-exam-container-karaf:3.4
    - org.apache.karaf:apache-karaf:3.0:zip
      - This could be replaced with any other Karaf distribution, such as Fuse

More Maven Dependencies

- Camel
  - org.apache.camel.karaf:apache-camel:2.12:xml:features
    - This is the Camel Feature Set that is Loaded into the OSGi Container
  - org.apache.camel:camel-core:2.12
  - org.apache.camel:camel-core-osgi:2.12
  - org.apache.camel:camel-core-xml:2.12
  - org.apache.camel:camel-blueprint:2.12
  - org.apache.camel:camel-mvel:2.12
  - org.apache.camel:camel-cxf:2.12

Generating Dependency File

- Dependency File should be Generated in Test Module to Use the `versionAsInProject()` method in our Container Configuration

```
<plugin>
        <groupId>org.apache.servicemix.tooling</groupId>
        <artifactId>depends-maven-plugin</artifactId>
        <version>1.2</version>
        <executions>
                <execution>
                        <id>generate-depends-file</id>
                        <goals>
                                <goal>generate-depends-file</goal>
                        </goals>
                </execution>
        </executions>
 </plugin>
```

Configuring the Container

- Container Configuration is Done with the `@Configuration` Annotation
- Method should Return `Option[]`
- Used for
  - Specifying the Distribution
  - Setting Distribution Properties
    - Log Level, Unpack Directory, Start Method, Etc…
  - Loading Features / Bundles
  - Adding/Modifying Distribution Environment
    - Property Files, Etc…

Example Configuration

```
@Configuration
 public static Option[] configure() throws Exception {
     return new Option[] {
             karafDistributionConfiguration()
                     .frameworkUrl(
                             maven().groupId("org.apache.karaf").artifactId("apache-karaf").type("zip")
                                     .versionAsInProject())
                     .useDeployFolder(false)
                     .karafVersion("3.0.0")
                     .unpackDirectory(new File("target/paxexam/unpack/")),

             logLevel(LogLevel.WARN),

             features(
                     maven().groupId("org.apache.camel.karaf").artifactId("apache-camel").type("xml")
                             .classifier("features").versionAsInProject(), "camel-blueprint",
                      "camel-mvel", "camel-cxf", "camel-test"),

             KarafDistributionOption.editConfigurationFilePut("etc/org.ops4j.pax.url.mvn.cfg",
                     "org.ops4j.pax.url.mvn.proxySupport", "true"),
             keepRuntimeFolder(),
             KarafDistributionOption.replaceConfigurationFile("etc/com.walmart.mqm.store.routes.cfg", new File(
                     "src/test/resources/com.walmart.mqm.store.routes.cfg")),

             mavenBundle().groupId("com.walmart.mqm").artifactId("storeBundle").versionAsInProject() };
 }
```


RED HAT CONSULTING

Modifying the Probe

- Modified Using the `@ProbeBuilder` Annotation
- Uses the Following Method Signature

```
@ProbeBuilder
public TestProbeBuilder probeConfiguration(TestProbeBuilder probe) {
   // Do Things
}
```

- Method should return the `probe` Parameter after Modification
- Sets Probe Specific Configurations via Headers
- Dynamically Imports all Packages by Default

RED HAT CONSULTING

Using Provisional Packages in the Probe

- Provisional Packages are by Default NOT Imported
- Probe must be Modified to Import Them
- Add the Following Line to your ProbeBuilder Method

```
probe.setHeader(Constants.DYNAMICIMPORT_PACKAGE,
"*;status=provisional");
```


RED HAT CONSULTING

Getting the Camel Context

- There will potentially be Multiple Camel Contexts running in the Container
- We need to get the Correct Context for Our Tests
- Best Done in `doPreSetup()` Method that can be Overridden from `CamelTestSupport`
- Camel Context should be Looked Up by it's name in the `BundleContext`

```
@Override
 protected void doPreSetup() throws Exception {
      camelContext = PaxExamTestUtil.getOsgiService(CamelContext.class, "(camel.context.name=" + CONTEXT_NAME
              + ")", 10000, bundleContext);
      assertNotNull(camelContext);
 }
```

Sending Messages in Tests

- 2 Step Process
- Create/Start the Template then Send Message
- Different from BlueprintTestSupport

- Getting and Starting the Template

```
ProducerTemplate template = camelContext.createProducerTemplate();
template.start();
```


RED HAT CONSULTING

Sending Messages in Tests

- Sending a Message with No Headers

```
template.send("direct:gw01_in", new Processor() {
        public void process(Exchange exchange) {
            Message in = exchange.getIn();
            in.setBody("Hello from Camel");
        }
    });
```

- Sending a Message with Headers

```
final Map<String, Object> headerMap = new HashMap<String, Object>();
headerMap.put("WM_MSG_ID", null);
headerMap.put("WM_HO_WMQ_QUEUE", null);

template.send("direct:gateway_in", new Processor() {
        public void process(Exchange exchange) {
            Message in = exchange.getIn();
            in.setBody("Hello from Camel");
            in.setHeaders(headerMap);
        }
    });
```


Using Mock Queues

- Create the Queue by Getting the Endpoint from the Camel Context and Casting to a `MockEndpoint`

```
MockEndpoint mockNoHomeOfficeDlq = (MockEndpoint)
        CamelContext.getEndpoint("mock:gateway_noHomeOfficeDlq");
```

- Set Mock Endpoint's Expectations

```
mockNoHomeOfficeDlq.expectedMessageCount(1);
mockNoHomeOfficeDlq.expectedBodiesReceived("Hello from Camel");
```

- Send Messages to Mock Endpoint
- Assert Mock Endpoint is Satisfied

```
mockNoHomeOfficeDlq.assertIsSatisfied(2500);
```


Demo

Demo

- Demo Time!

RED HAT CONSULTING

Resources

Additional Resources

- Pax Exam 3.x Wiki
  - https://ops4j1.jira.com/wiki/display/PAXEXAM3/Pax+Exam
- Karaf Container
  - https://ops4j1.jira.com/wiki/display/PAXEXAM3/Karaf+Container
- Container Reference
  - https://ops4j1.jira.com/wiki/display/PAXEXAM3/Karaf+Test+Container+Reference
- Karaf Integration Testing
  - http://karaf.apache.org/manual/latest/developers-guide/writing-tests.html

Q&A

- Questions?
- Comments?
- Concerns?

RED HAT CONSULTING