WildFly Core

Extending WildFly

Emmanuel Hugonnet, Red Hat
Jeff Mesnil, Red Hat

Thanks to Tomaz Cerar

Agenda

- What Are Extensions
- WildFly Core
- Write Simple Extension
- Cassandra Extension
- Demo

What Are Extensions?

- Entry point for extending WildFly
- Can provide
  - new deployment types
  - new model
  - services
  - resources

WildFly Core

- Really small (< 15 MiB)
- Consists of
  - JBoss Modules
  - Modular Service Controller (MSC)
  - Domain management (CLI, REST)
  - Deployment manager
  - Logging

JBoss Modules

- Modular class loader
- Isolated class loader
- Can be a set of many resources (jars)

Domain Management

- Manages configuration
- Backbone for all extensions
- Accessible via
  - Extensions
  - CLI
  - Admin console
  - DMR clients
  - …

MSC

- Truly concurrent service container
- Handles service dependencies
- Handles lifecycle
- “Real” functionality should be in services

DMR

- Detyped Model Representation
- JSON-like
- Internal format for all operations
- Internal model (ModelNode API)

Domain Model Definition

- Attributes
- Operations
- Resources
- Resource tree
  - PathElement
  - Single target (key=value)
  - Multi target (key=*)
- Resolvers & multi language support

Operation Handlers

- Defines operations on resources
- Execution stage
  - MODEL
  - RUNTIME
  - VERIFY
  - DOMAIN
  - DONE
- Can be chained

Resource Handlers

- Every resource needs add & remove
- In MODEL phase
  - validate and set model
- In RUNTIME phase
  - Add services
  - Add deployment processors

Deployment Manager

- Deployment repository
- Defines deployment phases
- Provides deployment processor infrastructure
- Can be used via
  - Domain management
  - Deployment scanner

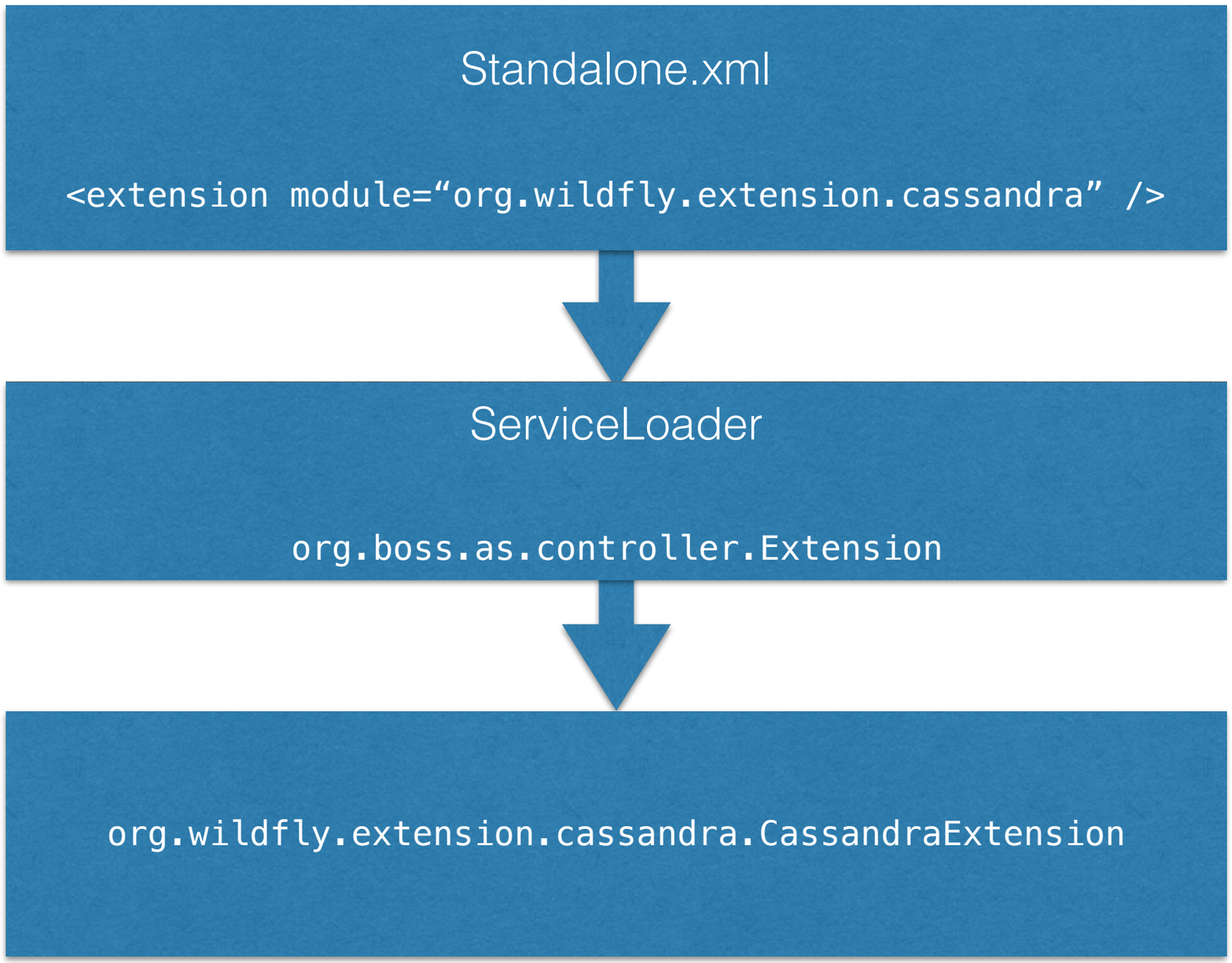
Deployment Processor

- Hooks into deployment lifecycle
- Can modify deployment behaviour
- Can define new deployment type

Extension Point

- Loaded via Service Loader
- Can define many subsystems
- Packaged as JBoss Module
- Referenced in configuration
- Can provide any new functionality

Extension Loading



Write Cassandra Extension

Building Blocks

- Define Extension point
- Define Root Model
- Define Add handler
- Define XML parser
- Define XML marshaller

Extension Class

```
public class CassandraExtension implements Extension {

    @Override
     public void initialize(ExtensionContext context) {

          …
     }


     @Override
     public void initializeParsers(ExtensionParsingContext context) {

          …
     }
}
```

Service Loader Entry

```
$ cd src/main/resources/META-INF/services/
```

```
$ cat org.jboss.as.controller.Extension

org.wildfly.extension.cassandra.CassandraExtension
```

Define Root Model

```
public class RootDefinition extends PersistentResourceDefinition {
  public static final RootDefinition INSTANCE = new RootDefinition();

  private RootDefinition() {
    super(CassandraExtension.SUBSYSTEM_PATH,
    CassandraExtension.getResourceDescriptionResolver(),
    RootAdd.INSTANCE,
    ReloadRequiredRemoveStepHandler.INSTANCE);
  }

  public Collection<AttributeDefinition> getAttributes() {
    return Collections.emptySet();
  }

  protected List<? extends PersistentResourceDefinition>
getChildren() {
   return Arrays.asList(ClusterDefinition.INSTANCE);
  }
}
```

Define Root Add Handler

```
class RootAdd extends AbstractBoottimeAddStepHandler {

    static final RootAdd INSTANCE = new RootAdd();

    private RootAdd() {

    }


    protected void populateModel(ModelNode operation, ModelNode model)
throws OperationFailedException {

        model.setEmptyObject();

    }

  }
```

Define XML Namespace

```
enum Namespace {

   // must be first

   UNKNOWN(null),

   CASSANDRA_1_0("urn:wildfly:cassandra:1.0");



   public static final Namespace CURRENT =
CASSANDRA_1_0;

}
```

Define XML Model

```
<subsystem xmlns="urn:wildfly:cassandra:1.0">

</subsystem>
```

Define Parser

```
public class SubsystemParser implements XMLStreamConstants, XMLElementReader<List<ModelNode>>,
XMLElementWriter<SubsystemMarshallingContext> {

  public final static SubsystemParser INSTANCE = new SubsystemParser();

  private static final PersistentResourceXMLDescription xmlDescription;

  static {

    xmlDescription = builder(RootDefinition.INSTANCE)
        .addChild(
          builder(ClusterDefinition.INSTANCE)
            .addAttributes(ClusterDefinition.ATTRIBUTES))
        .build();
  }

  public void readElement(XMLExtendedStreamReader reader, List<ModelNode> list) throws XMLStreamException {

    xmlDescription.parse(reader, PathAddress.EMPTY_ADDRESS, list);

  }

}
```

Define marshaller

```
public void writeContent(XMLExtendedStreamWriter writer,
SubsystemMarshallingContext context) throws XMLStreamException {

  ModelNode model = new ModelNode();
model.get(RootDefinition.INSTANCE.getPathElement().getKeyValuePa
ir())

       .set(context.getModelNode());

  xmlDescription.persist(writer,

      model,

      Namespace.CURRENT.getUriString());

}
```

Making It All Work

```
public void initializeParsers(ExtensionParsingContext context) {
  context.setSubsystemXmlMapping(SUBSYSTEM_NAME,
    Namespace.CASSANDRA_1_0.getUriString(),
    SubsystemParser.INSTANCE);
}
```

```
public void initialize(ExtensionContext context) {
  final SubsystemRegistration subsystem = context.registerSubsystem(SUBSYSTEM_NAME, 1, 0);
  final ManagementResourceRegistration registration =
subsystem.registerSubsystemModel(RootDefinition.INSTANCE);
  registration.registerOperationHandler(
    GenericSubsystemDescribeHandler.DEFINITION,
    GenericSubsystemDescribeHandler.INSTANCE);
    subsystem.registerXMLElementWriter(SubsystemParser.INSTANCE);
}
```

Module Definition

```
<module xmlns="urn:jboss:module:1.3" name="org.wildfly.extension.cassandra">
  <properties>
    <property name="jboss.api" value="private"/>
  </properties>

  <resources>
    <!-- the subsystem itself -->
    <resource-root path="wildfly-cassandra.jar"/>
    <!-- cassandra libs -->
    <resource-root path="cassandra-all-3.0.2.jar"/>
    <resource-root path="cassandra-thrift-3.0.2.jar"/>
    <!-- Datastax driver -->
    <resource-root path="cassandra-driver-core-3.0.0-rc1.jar"/>
    <!-- cassandra dependencies -->
    …
  </resources>

  <dependencies>
    …
    <module name="org.jboss.as.controller"/>
    <module name="org.jboss.msc"/>
  </dependencies>
</module>
```

Module Layout

```
$ tree modules/system/layers/base/org/wildfly/extension/cassandra/

modules/system/layers/base/org/wildfly/extension/cassandra/
└── main
    ├── module.xml
    ├── wildfly-cassandra.jar
    ├── cassandra-all-3.0.2.jar
    ├── cassandra-driver-core-3.0.0-rc1.jar
    ├── cassandra-thrift-3.0.2.jar
    ├── thrift-server-0.3.7.jar
    └── …
```

How to test it?

- Test harness enables you to test
  - xml parser
  - model consistency
  - MSC services
  - Compatibility (transformers)
  - Localization resources
  - …

```
public class SubsystemBaseParsingTestCase extends
AbstractSubsystemBaseTest {
  public SubsystemBaseParsingTestCase() {
    super(CassandraExtension.SUBSYSTEM_NAME,
      new CassandraExtension());
  }

  protected String getSubsystemXml() throws IOException {
   return readResource("/test-subsystem.xml");
  }
}
```

Attribute

- Definition
- Read & Write Handlers
- XML Persistence

Attribute Definition

```
static final SimpleAttributeDefinition TRACE_QUERY =
    new
SimpleAttributeDefinitionBuilder(CassandraModel.TRACE
_QUERY, ModelType.BOOLEAN, true)
        .setAllowExpression(true)
        .setDefaultValue(new ModelNode(false))
        .build();
```

Write Handler

```
private static class TraceQueryWriteAttributeHandler extends AbstractWriteAttributeHandler<Boolean>
{

    public SimpleWriteAttributeHandler() {
        super(TRACE_QUERY);
    }
```

```
    @Override
    protected boolean applyUpdateToRuntime(OperationContext context, ModelNode operation, String
attributeName, ModelNode resolvedValue, ModelNode currentValue, HandbackHolder<Boolean>
handbackHolder) throws OperationFailedException {
        ClusterResource resource =
(ClusterResource)context.readResource(PathAddress.EMPTY_ADDRESS);
        resource.setDebugMode(resolvedValue.asBoolean());
        return false;
    }

    @Override
    protected void revertUpdateToRuntime(OperationContext context, ModelNode operation, String
attributeName, ModelNode valueToRestore, ModelNode valueToRevert, Boolean handback) throws
OperationFailedException {
        ClusterResource resource =
(ClusterResource)context.readResource(PathAddress.EMPTY_ADDRESS);
        resource.setDebugMode(valueToRestore.asBoolean());
    }
}
```

Attribute Registration

```
@Override
public void registerAttributes(final ManagementResourceRegistration
rootResourceRegistration) {

    ClusterWriteAttributeHandler handler = new
ClusterWriteAttributeHandler(ATTRIBUTES);
    for (AttributeDefinition attr : ATTRIBUTES) {
        if(!TRACE_QUERY.equals(attr)) {

rootResourceRegistration.registerReadWriteAttribute(attr, null,
handler);
        }
    }

    rootResourceRegistration.registerReadWriteAttribute(TRACE_QUERY,
null, new TraceQueryWriteAttributeHandler());
}
```

XML Persistence

```
public class SubsystemParser implements XMLStreamConstants,
XMLElementReader<List<ModelNode>>,
XMLElementWriter<SubsystemMarshallingContext> {

    public final static SubsystemParser INSTANCE = new
SubsystemParser();

    private static final PersistentResourceXMLDescription
xmlDescription;

    static {
        xmlDescription = builder(RootDefinition.INSTANCE)
                .setMarshallDefaultValues(true)
                .addChild(
                        builder(ClusterDefinition.INSTANCE)
                                .addAttributes(ClusterDefinition.ATT
RIBUTES)
                                .setMarshallDefaultValues(true))
                .build();
    }
```

Operation

```
public class CassandraQueryHandler implements OperationStepHandler, CassandraOperation {

   public static final CassandraQueryHandler INSTANCE = new CassandraQueryHandler();

   public static final SimpleAttributeDefinition QUERY = SimpleAttributeDefinitionBuilder
      ...

   public static final StringListAttributeDefinition RESULT_SET = new
StringListAttributeDefinition.Builder("result_set")
      ...

   public static final SimpleOperationDefinition DEFINITION = new
SimpleOperationDefinitionBuilder("execute",
      CassandraExtension.getResourceDescriptionResolver())
         .setReadOnly()
         .setRuntimeOnly()
         .setParameters(QUERY, KEYSPACE)
         .setReplyType(ModelType.LIST)
         .setReplyParameters(RESULT_SET).build();

   @Override
   public void execute(OperationContext context, ModelNode operation) throws
OperationFailedException {
      …
   }
}
```

Demo

Questions?

http://www.wildfly.org/

https://github.com/wildfly/wildfly

https://docs.jboss.org/author/display/WFLY10/Extending+WildFly

Thanks!

@ehsavoie / http://www.ehsavoie.com/

@jmesnil / http://jmesnil.net/