

# Modular Java Development with OSGi and Spring DM

Duration:	1 day
Type:	advanced

## Description

This course enables developers to use OSGi to develop modular, service oriented applications via the Spring Dynamic Modules framework. The course begins with a review of class-loading and versioning issues in standard Java, progresses to an in-depth discussion of the OSGi specification and finishes with examples of building and deploying bundles via Spring DM. Delegates create both client side Swing based applications and server-side Spring MVC based Web Applications.

## Prerequisites

Delegates must be accomplished Java developers who have used Spring for dependency injection and written Spring configuration files using schema extensions. Experience of Spring MVC is helpful but not essential.

## List of Modules

### Review of Java Type Loading (Optional)

- How type names are generated in Java
- The class loading architecture within the JVM
- How the defining class loader affects type names
- Class loading and dependencies between JAR files
- Support for versioning in the JAR standard
- Problems with type loading in large applications

### Introducing OSGi and Spring DM

- Why the OSGi Alliance was formed
- Objectives of the OSGi standard
- Eclipse as an example of applying OSGi
- Current implementations of OSGi
- The relationship between Spring and OSGi
- Extra features provided by Spring DM

### Core Concepts of OSGi

- Components of an OSGi implementation
- The static and dynamic parts of OSGi
- How bundles create separate class-spaces
- Consumer Bundles verses Service Bundles
- The lifecycle for bundles defined by the OSGi spec
- How conflicts between class versions are resolved
- Services provided by an OSGi implementation
- OSGi enhancements to standard Java security

## Creating OSGi Bundles and Services

- Adding essential information in *MANIFEST.MF*
- Exporting and importing packages from bundles
- Handling dependencies to non-OSGi JAR files
- Writing a *BundleActivator* implementation
- Creating and publishing services via the activator
- Building a Consumer Bundle to use the published service

## Introducing Spring DM

- Problems using OSGi at the lowest level
- How Spring DM enables POJO based OSGi
- Understanding the Spring Dynamic Modules Extender bundle
- Using application contexts based on OSGi bundles
- Exporting Spring beans to OSGi via schema extensions
- Importing OSGi services via dependency injection
- Support for transparently replacing missing services
- Issues with references to services in collections
- Attaching listeners to monitor the lifetime of services
- Support for integration testing OSGi modules
- Using Spring DM to deploy Spring MVC based web apps