# BindForge

Version 0.5

Module Framework

March 16, 2009



Roman Roelofsen

# Contents

# 1 What is BindForge?

The mission of BindForge is to ease the OSGi development. BindForge provides sophisticated dependency injection facilities and various abstraction layers for OSGi services and other compendium elements. The configuration is done via a powerful Scala-based DSL. These features are provided non-intrusively so that Java programmers do not need to learn e.g. the Scala compiler or other tools. Simply put a text file in your bundle. It's as simple as that!

BindForge relies on Guice for the dependency injection features. Unlike normally required when using Guice, BindForge users do not need to put special annotations in their classes (e.g. @Inject). Hence the Java classes are full POJOs again. At the same time, BindForge is 100% compatible with Guice. All known features, e.g. Spring beans support, are available. For the OSGi service registry interaction, BindForge uses the Peaberry dynamic service extension. Additionally, BindForge completely abstracts from various OSGi elements. Without introducing code dependencies, users can fully utilise the OSGi service platform.

# 2 Download

To use BindForge, you need to install 2 bundles in your OSGi framework:

- bindforge-*version*.jar
- scala-full-bundle-*version*.jar

Both bundles are available via a direct file download or via the BindForge Maven repository. The Scala bundle contains the complete Scala library and the Scala compiler. **Note:** In case you already have a Scala bundle installed, it is recommended to replace the existing one with the BindForge version since the existing bundle may not contain the Scala compiler.

The bundle files are available at the BindForge homepage, download section.

### 2.1 Maven Repository

Besides the direct download links, all required bundles are also available via the BindForge Maven repository. To use the repository, add the following configuration (1) to your pom.xml:

```
<repository>
    <id>bindforge.org</id>
    <name>BindForge Maven2 Repository</name>
    <url>http://repository.tuxed.de</url>
</repository>
```

**Listing 1:** BindForge Maven Repository

After you configured the repository, you will need to add the artifact dependencies to your pom.xml (2):

```
<dependency>
        <groupId>org.bindforge</groupId>
        <artifactId>bindforge</artifactId>
        <version>0.5.0</version>
</dependency>
<dependency>
        <groupId>org.scala-lang</groupId>
        <artifactId>scala-full-bundle</artifactId>
        <version>2.7.3</version>
</dependency>
```

**Listing 2:** BindForge Artifacts

## 3 Bundle Configuration

To use BindForge, you need to put a configuration file in your bundle. Once your bundle gets started, BindForge will read this file to activate the configuration.[1] A manifest entry in the bundle is used to specify the configuration file. Listing 3 shows an example MANIFEST.MF.

```
Bundle-ManifestVersion: 2
Bundle-SymbolicName: org.acme.yourapp
```

[1] This behaviour is called *Extender Pattern* and very common for OSGi frameworks. For example, the *Declarative Services* specification uses the same mechanism.

```
3 BindForge-Config: org.acme.yourapp.Config
```

**Listing 3:** MANIFEST.MF with Configuration

The header `BindForge-Config` in line 3 specifies the configuration that should be used. As you can see in the example, the value has the form of `packagename.ClassName`. BindForge configurations are full Java classes written in the Scala programming language[2].

- tell bf about the config (manifest)

- this point to class file, but you dont need to compile it (but you can)

- dir in bundle, everything will get compiled

[2]http://www.scala-lang.org