Fungal Developer's Guide

A High-Performance Java Kernel

Table of Contents

1

About Fungal

The goal of the Fungal project is to provide a high-performance Java kernel environment.

1.1. The team

Jesper Pedersen acts as the lead for the Fungal project. He can be reached at jesper (dot) pedersen (at) comcast (dot) net.

2

Introduction

The Fungal kernel is a Plain Old Java Object (POJO) based kernel which can build a container environment for Java services.

The focus of the Fungal kernel is simplicity and high performance.

The Fungal kernel has no dependencies on any 3rd party libraries as the kernel is built directly on the Java 6 API.

2.1. Highlights

Highlights of the Fungal kernel:

- 100% Pure Java 6 implementation
- Bean definition language
- Lifecycle of beans
- Parallel deployment
- Dependency management
- Classloader architecture
- Deployer framework
- Hot deployment
- Communication protocol
- Core services
- Netboot support
- Management (JMX) utility
- Remote access to management layer (JSR-160)
- Support multiple kernels in same VM

3

Building

3.1. Prerequisites

3.1.1. Java Development Kit (JDK)

You must have one of the following JDKs installed in order to build the project:

- Sun JDK 1.6.x
- Sun JDK 1.7.x

Remember to ensure that "javac" and "java" are in your path (or symlinked).

```
JAVA_HOME=/location/to/javahome
export JAVA_HOME

PATH=$JAVA_HOME/bin:$PATH
export PATH
```

3.1.2. Apache Ant

You must have Apache Ant 1.8.1+ installed on your system.

Remember to ensure that "ant" are in your path (or symlinked).

```
ANT_HOME=/location/to/anthome
export ANT_HOME

PATH=$ANT_HOME/bin:$PATH
export PATH
```

3.1.3. Apache Ivy

The Fungal project uses Apache Ivy for dependency management.

Apache Ivy is automatically downloaded and included in the development environment, so no additional setup is required.

3.1.4. Git

You must have Git 1.6+ installed on your system.

Remember to ensure that "git" are in your path (or symlinked).

3.2. Obtaining the source code

3.2.1. Anonymous Git access

The anonymous Git repository is located under:

```
git checkout git://github.com/jesperpedersen/fungal.git
```

3.2.2. Developer Git access

The developer Git repository is located under:

```
git checkout git://github.com/fungal.git
```

3.2.3. Git branches

We have the following branches for the project:

- master

  The head of development targeting the next upcoming release.

3.3. Compiling the source code

In order to build the Fungal project you execute:

```
ant <target>
```

where target is one of

- jars

  Builds the JAR archives in the distribution.

- test

  Builds the JAR archives in the distribution and runs all the test cases.

- docs

  Builds the API documentation for the project.

- release

  Builds a release of the project.

- clean

  Cleans the project of temporary files.

- clean-cache

  Cleans the Apache Ivy repository.

See the full list of targets in the main build.xml file.

An example to build the Fungal JAR files:

```
ant clean jars
```

4

Issue tracking

4.1. Location

The issue tracking for the project is located at http://github.com/jesperpedersen/fungal/issues
.

4.2. Components

The project is divided into the following components:

Table 4.1. Project components

Component	Description
Core	The core implementation of the project.
CLI	The command line client interface.

4.3. Categories

The system contains the following categoies:

Table 4.2. JIRA categories

Category	Description
Feature Request	Request for a feature made by the community.
Bug	Software defect in the project.
Task	Development task created by a member of the team.
Release	Issue which holds informations about a release.
Component upgrade	Identifies a thirdparty library dependency.

4.4. Life cycle

All issues folows the following life cycle:

Table 4.3. Issue lifecycle

Lifecycle	Description
Open	An issue currently not implemented.
Coding in Progress	An issue currently being worked on.
Reopen	An issue that needs further work after it has been resolved.
Resolved	An issue which has been implemented.
Closed	An issue that has been resolved and is include in a release.

Note: Thirdparty issues can't be resolved nor closed during a development cycle. These are resolved and closed as part of the release procedure of the project. The reason for this is that the library in question can receive further updates during the active development cycle.

4.5. Priorities

All issues are assigned one of the following priorities:

Table 4.4. Issue priorities

Priority	Description
Blocker	An issue that needs to be fixed before the release.
Critical	An issue that is critical for the release.
Major	The default priority for an issue.
Minor	An issue that is optional for a release.
Trivial	An issue that is optional for a release and have a lower priority than Minor.

5

Testing

5.1. Overall goals

The overall goals of our test environment is to execute tests that ensures that we have full coverage of the code in Fungal.

TBD

5.2. Quality Assurance

In addition to the test suite the Fungal project deploys various tools to increase the stability of the project.

The following sections will describe each of these tools.

5.2.1. Checkstyle

Checkstyle is a tool that verifies that the formatting of the source code in the project is consistent.

This allows for easier readability and a consistent feel of the project.

The goal is to have zero errors in the report. The checkstyle report is generated using

```
ant checkstyle
```

The report is generated into

```
reports/checkstyle
```

The home of checkstyle is located here: http://checkstyle.sourceforge.net/.

5.2.2. Findbugs

Findbugs is a tool that scans your project for bugs and provides reports based on its findings.

This tool helps lower of the number of bugs found in the Fungal project.

The goal is to have zero errors in the report and as few exclusions in the filter as possible. The findbugs report is generated using

```
ant findbugs
```

The report is generated into

```
reports/findbugs
```

The home of findbugs is located here: http://findbugs.sourceforge.net/.

5.2.3. Cobertura

Cobertura generates a test suite matrix for your project which helps you identify where you need additional test coverage.

The reports that the tool provides makes sure that the Fungal project has the correct test coverage.

The goal is to have as high code coverage as possible in all areas. The Cobertura report is generated using

```
ant cobertura
```

The report is generated into

```
reports/cobertura
```

The home of Cobertura is located here: http://cobertura.sourceforge.net/.

5.2.4. Tattletale

Tattletale generates reports about different quality matrix of the dependencies within the project.

The reports that the tool provides makes sure that the Fungal project doesn't for example have cyclic dependencies within the project.

The goal is to have as no issues flagged by the tool. The Tattletale reports are generated using

```
ant tattletale
```

The reports are generated into

```
reports/tattletale
```

The home of Tattletale is located here: http://www.jboss.org/tattletale.