Tested Objects 1.0: FitNesse: Developers Guide

FitNesse Integration for Naked Objects 4.0.x

Version 0.1

Preface

Tested Objects is a sister project for the Naked Objects framework, providing integration with FitNesse to enable agile acceptance (or scenario) testing. The integration bundles the *FitNesse* wiki server and provides a set of generic *Fitnesse* fixtures that interact with the domain model in the same manner that a *Naked Objects* viewer does. It also provides a Maven archetype to get you started quickly.

This developers' guide explains how to build *Tested Objects* from source, allowing you to contribute back and extend the range of capabilities.

If you are simply interested in using *Tested Objects* as-is, please consult the user guide. (Note that the archetype also has a built-in user guide).

Tested Objects is hosted on SourceForge, and is licensed under Apache Software License v2. *Naked Objects* is also hosted on SourceForge, and is also licensed under Apache Software License v2.

Chapter 1 Introduction

This chapter introduces the organization of this developers' guide.

Tested Objects is one of a number of sister projects for Naked Objects. Each of these sister projects are organized along the same general lines: they have the same directory structure, the same coding conventions, a shared "corporate" Maven POM to define build artifacts, the same release process and so on.

The Star Objects project is an umbrella for all of these sister projects. As such it holds the corporate POM and a number of other shared artifacts, such as a site template so that the Maven sites for all sister projects have the same general look-n-feel. It also hosts a Maven snapshot repository and release repository.

In addition, the *Star Objects* also has a developers guide (available online here). This describes how to build any given sister project from source, how to be a contributor, and how (as a project admin) to release code artifacts to the repositories and how to deploy the site.

This developers guide therefore provides only a high level outline of the structure of the modules, and provides only summary steps for how to build and deploy the sister projects. Any variations from the standard procedures described in the *Star Objects* developers guide are also given.

Tihs guide also provides design/implementation notes, in ???. If you are thinking about or fixing a bug or contributing a new feature, you might find some starting points here (over and above reading the Javadocs, tests and code, of course).

Part I

FitNesse Integration

Currently Tested Objects provides only a FitNesse integration, so this developers guide consists of a single part.

Chapter 2

Modules

This chapter describes the modules that make up Tested Objects' FitNesse integration.

The modules that make up *Tested Objects*' FitNesse integration follow the general conventions of sister projects, with a *main* module, a *support* module and a *testapp* module. You can read more about this in the *Star Objects* developer guide. However, the directory structure this does not quite follow the standard layout for sister projects: the *main* and *support* modules are nested under `trunk/fitnesse/` (as opposed to directly under `trunk/`). That's because *Tested Objects* may be extended to support other testing frameworks (eg Concordion) in the future.

2.1. Directory Structure

The source code directory structure for *Tested Objects* is as follows:

```
trunk/
  fitnesse/           # FitNesse support
    main/               # main release for FitNesse, including Maven site
      fixtures/           # the FIT fixtures that integrate NO with FitNesse
      documentation/      # this documentation
    support/            # support - built after main release
      release/            # defines dependencies for projects using the 'fixtures' module
      archetype/          # archetype
    testapp/            # application for testing - not released
tags/
  fitnesse/main       # tags for trunk/fitnesse/main
  fitnesse/support    # tags for trunk/fitnesse/support
```

Note that this directory structure is nested under `trunk/fitnesse/` (as opposed to directly under `trunk/`). As is usual, though, to ensure that tags go into the correct location when releasing, the `mvn-release-plugin` plugin has been configured (using `<tagBase>`) to override its default location.

You can checkout the entire trunk using Subversion:

```
svn co https://testedobjects.svn.sourceforge.net/svnroot/testedobjects/trunk ~/testedobjects/
trunk
```

2.2. Main Modules

As the above shows, there are two separate released artifacts:

The *main* (`org.starobjects.tested.fitnesse:main`) is a multimodule project that defines the main artifacts that implement *Tested Objects*' FitNesse support. It contains:

- the fitnesse *fixtures* (`org.starobjects.tested.fitnesse:fixtures`)

  This contains implementations of *FitNesse*'s fixture API which in turn interact with the domain application.

  It uses `org.starobjects.tested.fitnesse:main` as its parent (and thus inherits transitively from the corporate POM).

- the *documentation* (`org.starobjects.tested.fitnesse:documentation`)

  The documentation submodule contains the user and developers' guides

  It also uses `org.starobjects.tested.fitnesse:main` as its parent.

It uses the corporate POM (org.starobjects.star:corporate) as its parent.

2.3. Support Modules

The *support* (`org.starobjects.tested.fitnesse:support`) is a multimodule project that provides a number of supporting artifacts. It contains:

- an additional *release* module (`org.starobjects.tested.fitnesse:release`)

  This is a convenience module that can be used as a parent by projects using the *FitNesse* integration provided by *Tested Objects* (for example, as generated by the archetype, below). Its primary purpose is to define a consistent set of versions in `<dependencyManagement>` tag.

  Note that this module does not inherit from the *main* POM, instead it inherits from the *Naked Objects Framework*'s equivalent `org.nakedobjects:release` module (thus defining a stack of dependencies).

- the *archetype* (`org.starobjects.tested.fitnesse:archetype`)

  This is released after the main release, since it needs to be updated to depend on the released versions of `org.starobjects.tested.fitnesse:main`. Its version numbers are the same as those of `org.starobjects.tested.fitnesse:main`.

  This module also inherits from the starobjects corporate POM (`org.starobjects.star:corporate`).

Like *main*, the *support* module also uses the corporate POM (org.starobjects.star:corporate) as its parent.

2.4. TestApp Module

The testapp module is a test application for adhoc testing of FitNesse. It is not a released artifact.

Chapter 3

Building, Documenting and Deploying

This chapter outlines how to build, document and deploy Tested Objects.

The build, documentation and deployment process follows the general standard for sister projects, as documented in the *Star Objects* developers' guide. The one exception is that deploying Tested Objects also involves deploying FitNesse itself; FitNesse is not formally released into Maven central repo, so the starobjects repo hosts it instead.

The sections in this chapter correspond to the parts one, two and three of *Star Objects* developers' guide.

3.1. Building from Source

There are no special steps required for building *Tested Objects* from source.

In particular, note that the FitNesse jar is *not* prerequisite software; instead it is managed as a Maven module (see Section 3.3, “Release Process”).

You can therefore just follow the processes described in *Star Objects* developers' guide:

- build the main:

  ```
  $ cd ~/testedobjects/trunk/fitnesse/main
  $ mvn clean install
  ```

- build the support:

  ```
  $ cd ~/testedobjects/trunk/fitnesse/support
  $ mvn clean install
  ```

3.2. Contributing Changes

There are no special considerations for contributing changes for *Tested Objects*. You can therefore just follow the processes described in *Star Objects* developers' guide.

3.3. Release Process

There are one special consideration for releasing/deploying for *Tested Objects*. Specifically, *FitNesse* is not currently available through Maven central repository. It must therefore be uploaded into the *Star Objects* Maven repo, at http://starobjects.sourceforge.net/m2-repo/release/. This is what is referenced from the *Tested Objects* POMs.

FitNesse itself is available from http://fitnesse.org/FrontPage.FitNesseDevelopment.DownLoad.

To install in the *Star Objects* repository (on the actual server), use:

```
$ mvn install:install-file      \
  -D file=fitnesse-20090818.jar \
  -D groupId=org.fitnesse       \
  -D artifactId=fitnesse        \
  -D version=20090818           \
  -D packaging=jar              \
  -D generatePom=true
```

Although *FitNesse* is open source, the *FitNesse* download site does not make the source code available. So it's also manually downloaded a ZIP of the source from the GIT source code repository, and then rezip it up in the same format that Maven prepares. I installed this also:

```
$ mvn install:install-file              \
  -D file=fitnesse-20090818-sources.jar \
  -D groupId=org.fitnesse               \
  -D artifactId=fitnesse                \
  -D version=20090818                   \
  -D packaging=jar                      \
  -D classifier=sources                 \
  -D generatePom=true
```

Otherwise than these points you can just follow the processes described in *Star Objects* developers' guide, to:

- for deployments, update `~/.m2/settings.xml`:

  ```
  <servers>
    <server>
      <id>testedobjects-site</id>
      <username>xxx</username>
      <password>xxx</password>
    </server>
  </servers>
  ```

- make documentation changes to DocBook and to the site
- deploy the site locally

  ```
  $ cd ~/testedobjects/trunk/fitnesse/main
  $ mvn site-deploy -D dist=local
  ```

  This will deploy to `/tmp/m2-sites/testedobjects`.

- deploy a code snapshot

  First, deploy *main*:

  ```
  $ cd ~/testedobjects/trunk/fitnesse/main
  $ mvn clean install deploy -D dist=remote
  ```

  Then, deploy *support*:

```
$ cd ~/testedobjects/trunk/fitnesse/support
$ mvn clean install deploy -D dist=remote
```

- tag a release and then deploy a code release

TODO: ydetails required here.

- deploy a site remotely

  then, deploy the site (you'll also need a sourceforge terminal session running; see *Star Objects* developers guide for details):

```
$ cd ~/testedobjects/trunk/fitnesse/main
$ mvn site-deploy -D dist=remote
```

Chapter 4

Design Notes

This chapter will contain design notes on the implementation of the FitNesse integration.

TODO: describe the internal design/architecture here...