JPA Objects 1.0: Developers' Guide

JPA/Hibernate Integration for Naked Objects 4.0

Version 0.1

Preface

JPA Objects is a sister project for the Naked Objects framework, allowing JPA-annotated domain objects to be persisted to a relational database (with Hibernate as the underlying JPA implementation).

This developers' guide explains how to build *JPA Objects* from source, allowing you to contribute back and extend the range of capabilities. If you are simply interested in using *JPA Objects* as-is, please consult the user guide.

JPA 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.

JPA 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).

Chapter 2
Modules

This chapter describes the modules that make up JPA Objects'.

The modules that make up *JPA Objects* 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.

2.1. Directory Structure

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

```
trunk/
  main/               # main release for JPA, including Maven site
    applib/             # the JPA application library
    metamodel/          # adds JPA semantics into the NO metamodel
    runtime/            # JPA runtime support (uses Hibernate)
    tools/              # DBA support for schema and fixture management (uses Hibernate)
    documentation/      # this documentation
  support/            # support release for JPA
    release/              # defines dependencies for projects using the 'fixtures' module
    archetype/            # archetype - released after the main release
  testapp/            # application for testing - not released
tags/
  main                # tags for trunk/main
  support             # tags for trunk/support
```

As is usual, 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://jpaobjects.svn.sourceforge.net/svnroot/jpaobjects/trunk ~/jpaobjects/trunk
```

2.2. Main Modules

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

The *main* release (`org.starobjects.jpa:main`) is a multimodule project that uses the corporate POM (org.starobjects.star:corporate) as its parent. It contains:

- the *applib* (`org.starobjects.jpa:applib`)

  The application library submodule defines ....

  The applib uses `org.starobjects.jpa:main` as its parent.
- the *metamodel* (`org.starobjects.jpa:metamodel`)

  The metamodel submodule defines implementations of the Naked Objects' `FacetFactory` API that allow JPA semantics to be incorporated into the Naked Objects metamodel.

  The metamodel also uses `org.starobjects.jpa:main` as its parent.
- the *runtime* (`org.starobjects.jpa:runtime`)

  The runtime submodule defines the implementations of the Naked Objects' `ObjectsStore` API that uses Hibernate to persist objects to the database.

  The runtime also uses `org.starobjects.jpa:main` as its parent.
- the *tools* (`org.starobjects.jpa:tools`)

  The tools submodule defines command line utilities to allow the DBA to manage schemas and to seed the database using fixtures.

  The runtime also uses `org.starobjects.jpa:main` as its parent.
- the *documentation* (`org.starobjects.jpa:documentation`)

  The documentation submodule contains the user and developers' guides

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

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

2.3. Support Modules

The *support* release (`org.starobjects.jpa:support`) is a multimodule project that provides artifacts to help application developers use *JPA Objects* in their own projects. It contains:

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

  This is a convenience module that can be used as a parent by projects using the *JPA Objects* object store (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* release (`org.starobjects.jpa:archetype`)

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

The build, documentation and deployment process follows the general standard for sister projects, as documented in the *Star Objects* developers' guide. 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 *JPA Objects* from source.

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

- build the main:

```
$ cd ~/jpaobjects/trunk/main
$ mvn clean install
```

- build the support:

```
$ cd ~/jpaobjects/trunk/support
$ mvn clean install
```

3.2. Contributing Changes

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

3.3. Release Process

There are no special considerations for releasing/deploying for *JPA Objects*. You can therefore just follow the processes described in *Star Objects* developers' guide, to:

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

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

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

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

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

- deploy a code snapshot

  First, deploy *main*:

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

  Then, deploy *support*:

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

- tag a release and then deploy a code release

> TODO: details 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 ~/jpaobjects/trunk/main
$ mvn site-deploy -D dist=remote
```

Chapter 4

Design Notes

This chapter contains notes on the architecture, design and implementation of JPA Objects.

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