Red Hat  
Domain Driven Design QuickStart

Usage Instructions

July 2013

©2013

All rights reserved. No part of this material may be reproduced without the prior written permission of Red Hat.

# Part 1: Obtain the QuickStart code and install it in your environment

Overview

The overall goal of this QuickStart is to walk you through the process of creating an example application illustrating Domain Driven Design (DDD), Behavior Driven Development (BDD) and Test Driven Development (TDD) utilizing BRMS as the domain engine.

We will start by installing the five Maven projects shown as solid boxes below. The diagram also shows

* A dotted-line box for an atm.ui project to represent the top layer of the DDD stack for completeness, although the UI layer is outside the scope of the QuickStart.
* A dotted-line box for the atm.domain project that will be created as part of the QuickStart.

Each project in the diagram is labeled with its Maven ArtifactId. The GroupId for each project is com.rhc.ddd. The four projects on the left side of the diagram represent the four layers of the DDD stack. In a full-scale application a layer might be implemented with more than one project.

Contains code that will be added to atm.domain during the QuickStart. Not modified during the QuickStart; the entire project may be deleted when atm.doman is complete.

atm.domain

Created from domain-layer-archetype during the QuickStart. During the QuickStart, code is added to this project from atm.domain.snippets.

domain-layer-archetype

atm.domain.  
snippets

Initially a skeletal project. This project will be completed during the QuickStart.

Legend

Child project Parent project

Dependent project Dependency

Archetype project for QuickStart domain layer projects. Not modified during the QuickStart.

atm.infrastructure

Initially a skeletal project. This project will be completed during the QuickStart. In a full-scale application would typically be a WAR.

atm.application

drools-reference-implementation

A useful parent project for domain-layer projects. It contains dependency-management elements to help manage dependency versions. Not modified during the QuickStart.

Shown to complete the four-layer DDD stack. This project is outside the scope of the QuickStart.

atm.ui

Requirements & Assumptions

* Access to the five Red Hat DDD QuickStart projects. For example, you may have received a zip or tar file or information on how to obtain the projects from GitHub.
* Internet access for Maven dependencies.
* This QuickStart assumes you have experience working with Maven projects and Java source code.

Instructions

1. Take the steps necessary to make the five QuickStart projects available in your development environment.

Note that some of the code requires Java SE 6.

There should be no Maven errors or compiler errors.

1. The Maven archetype domain-layer-archetype contains several useful classes. Use it to generate a Maven project with ArtifactId atm.domain and version 1.1.2.

For example, from the command line the following may be used:

mvn archetype:generate -DarchetypeGroupId=com.rhc.ddd -DarchetypeArtifactId=domain-layer-archetype -DarchetypeVersion=1.1.1 -DgroupId=com.rhc.ddd -DartifactId=atm.domain -Dversion=1.1.2

There should be no Maven errors or compiler errors.

# Part 2: Set up and test the Domain

Overview

The domain layer is central in Domain Driven Design. In this part of the QuickStart we will

* Set up entity and Value Object classes (just one to keep the QuickStart simple)
* Set up domain services
* Set up the rules engine
* Illustrate Behavior Driven Development

Requirements & Assumptions

* This part assumes that you have the atm.application and atm.infrastructure QuickStart projects available, that you have generated atm.domain from the archetype, and that you have access to the atm.domain.snippets project.

Instructions

Some numbered steps are followed by lettered sub-steps. The lettered sub-steps provide details on how to complete the numbered step.

1. Entity classes and Value Object classes are defined in the Domain Layer.
   1. Copy the Account class from the atm.domain.snippets project to atm.doman (in the src/main/java section).
2. The domain layer offers services to the application layer. In a full-scale environment these services might be exposed as REST services but in this QuickStart we will expose a service procedurally.
   1. Copy the interface AccountService from the snippets project to atm.domain and study it.
3. To perform its work, the domain layer needs access to infrastructure services. The domain layer can define its infrastructure needs in an interface (or interfaces). With this approach the domain layer has no dependency on any specific infrastructure; the application (or integration test program) can inject any objects that satisfy the interface requirements.

With this approach atm.domain project has no Maven dependency on any infrastructure project.

* 1. Copy the interface AtmRepositoryApi from snippets to atm.domain and study it.

1. For testing we can implement this interface in a simple manner.
   1. Copy the class AtmRepositoryApiTestImpl to the src/**test**/java (not **main**) section of atm.domain.
2. We can implement the domain layer service(s) using BRMS.
   1. Copy the classes TransferRequest and AccountServiceImpl to the domain project.
   2. Also copy atm.drl to src/main/resources. This file contains the rules.
3. Test the service.
   1. Copy the class AtmTest and study it.
   2. Run it as a JUnit test and verify there are no errors.
4. Cucumber.
   1. Copy the class TransferTest and the feature file transfer.feature from the snippets project to the domain project.
   2. Run TransferTest as a JUnit test.
   3. Create class TransferStepDefinitions and fill in the body with the Console output.
   4. Replace the throws statements with printlns to verify flow.
   5. Run and examine the output.
   6. Replace the step bodies with appropriate code and re-run.

# Part 3: Set up and test the application

Overview

In this part of the QuickStart we will add the application layer above the domain layer and the infrastructure layer below the domain layer.

The application uses Spring dependency injection.

Requirements & Assumptions

* This part assumes that you completed the previous part.

Instructions

Some numbered steps are followed by lettered sub-steps. The lettered sub-steps provide details on how to complete the numbered step.

1. Update the atm.infrastructure project as follows
   1. In pom.xml uncomment the dependency on the domain project.
   2. In AtmRepositoryApiImpl uncomment all commented-out code.
2. Update the atm.application project as follows
   1. In pom.xml uncomment the dependency on the domain and infrastructure projects.
   2. In AtmApplication uncomment all commented-out code.
3. Run the AtmApplication class and verify a successful result.