Fornax-Platform : 2. Hello World Tutorial (CSC)

This page last changed on Mar 13, 2011 by patrik_nordwall.

Sculptor Hello World Tutorial

This hands-on tutorial will walk you through the steps of how to create a small application and explore it with some JUnit tests. This example is also used and extended in 3 other tutorials:

- Archetype Tutorial
- CRUD GUI Tutorial
- REST Tutorual

This is an introduction to Sculptor. A more extensive example is available in the Advanced Tutorial. If you would like to see something more exciting than running JUnit tests we can recommend the CRUD GUI Tutorial.

Before you start you must follow the instructions in the **Installation Guide**.

Table of Contents:

Part 1 - Setup Project

In this first part we will setup the project structure for maven and eclipse.

1. Use the following command (one line) to create a maven pom and file structure. You can change the groupId and artifactId if you like.

mvn archetype:generate -DarchetypeGroupId=org.fornax.cartridges -DarchetypeArtifactId=fornax-cartridges-sculptor-archetype-standalone -DarchetypeVersion=2.0.0 -DarchetypeRepository=http://fornax-platform.org/nexus/content/repositories/public

Fill in groupId and artifactId:

```
Define value for groupId: : org.helloworld
Define value for artifactId: : helloworld
Define value for version:  1.0-SNAPSHOT: :
Define value for package:  org.helloworld: :
```

There will be warnings like this:

```
[WARNING] org.apache.velocity.runtime.exception.ReferenceException: reference :
template = archetype-resources/pom.xml [line 40,column 42] : ${fornax-oaw-m2.ver
sion} is not a valid reference.
```

Ignore these warnings and continue with next step if you see no errors.

2. In the new directory, run `mvn eclipse:eclipse` to create an Eclipse project with the same dependencies as in the pom.

3. Open Eclipse and import the project.

Part 2 - Generate Code

In this part we will write a Sculptor DSL file and generate code from it.

1. Modify the file named `model.btdesign` in the folder `src/main/resources/`

2. Open the `model.btdesign` file with Sculptor DSL editor, double-click on it.
Add something like this to the design file.

```
Application Universe {
    basePackage=org.helloworld

    Module milkyway {
        Service PlanetService {
            String sayHello(String planetName);
            protected findByExample => PlanetRepository.findByExample;
        }

        Entity Planet {
            String name key;
            String message;

            Repository PlanetRepository {
                findByExample;

            }
        }

    }
}
```

Try the code completion, error highlight and outline view.
It is a Module containing one Entity, with a Repository. The concepts are taken from Domain-Driven Design.

3. Run `mvn clean install` to generate code and build. The JUnit test will fail.

✓ If you run maven from the command prompt you have to do a refresh in Eclipse. If you run maven as an external task in Eclipse it can refresh automatically.

4. Look at the generated code. In `src` and `test` folders the code is only generated once, and you can do manual changes. In `generated` and `test/generated` it is generated each time, i.e. don't touch.

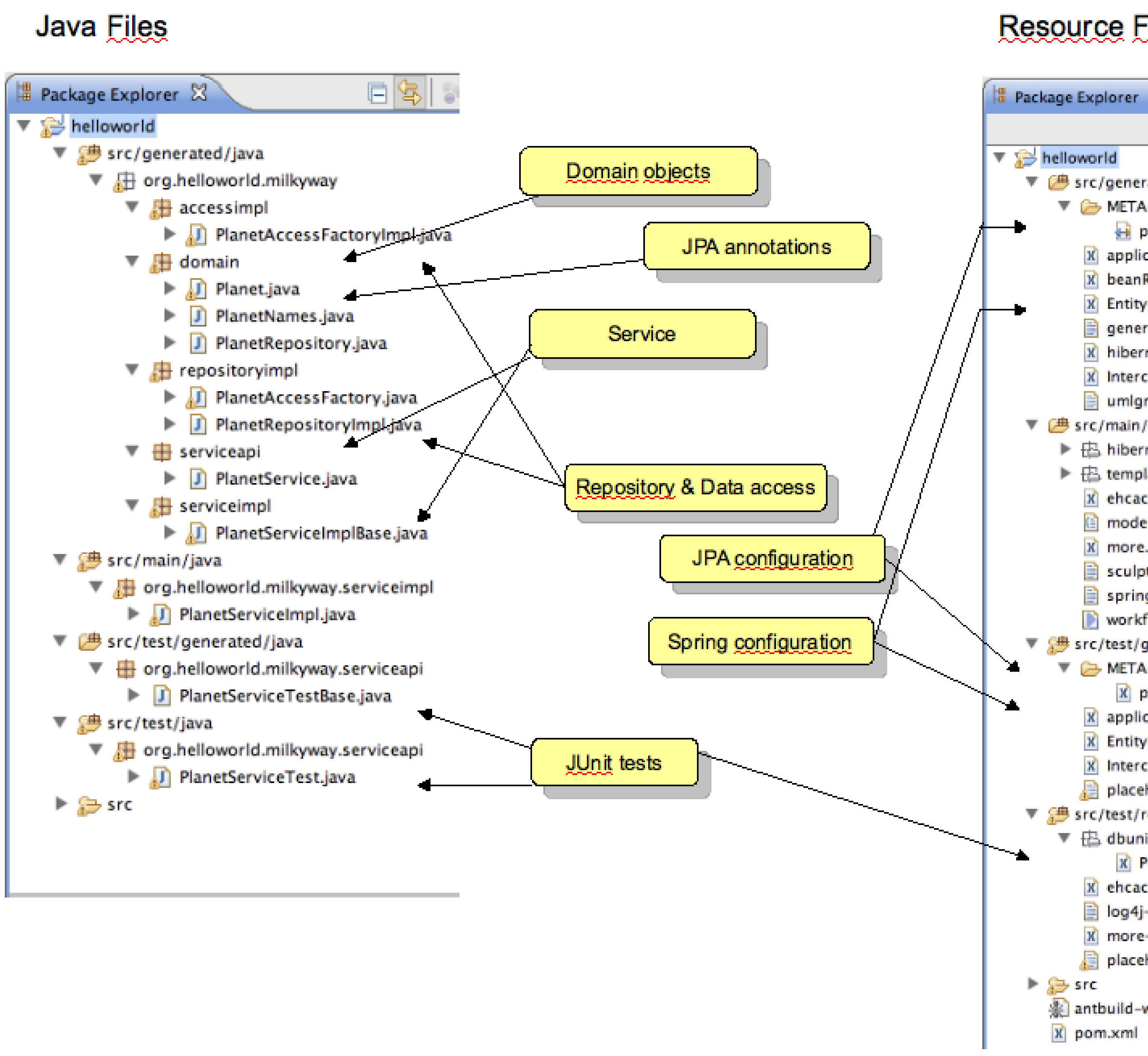

Figure 1. File structure

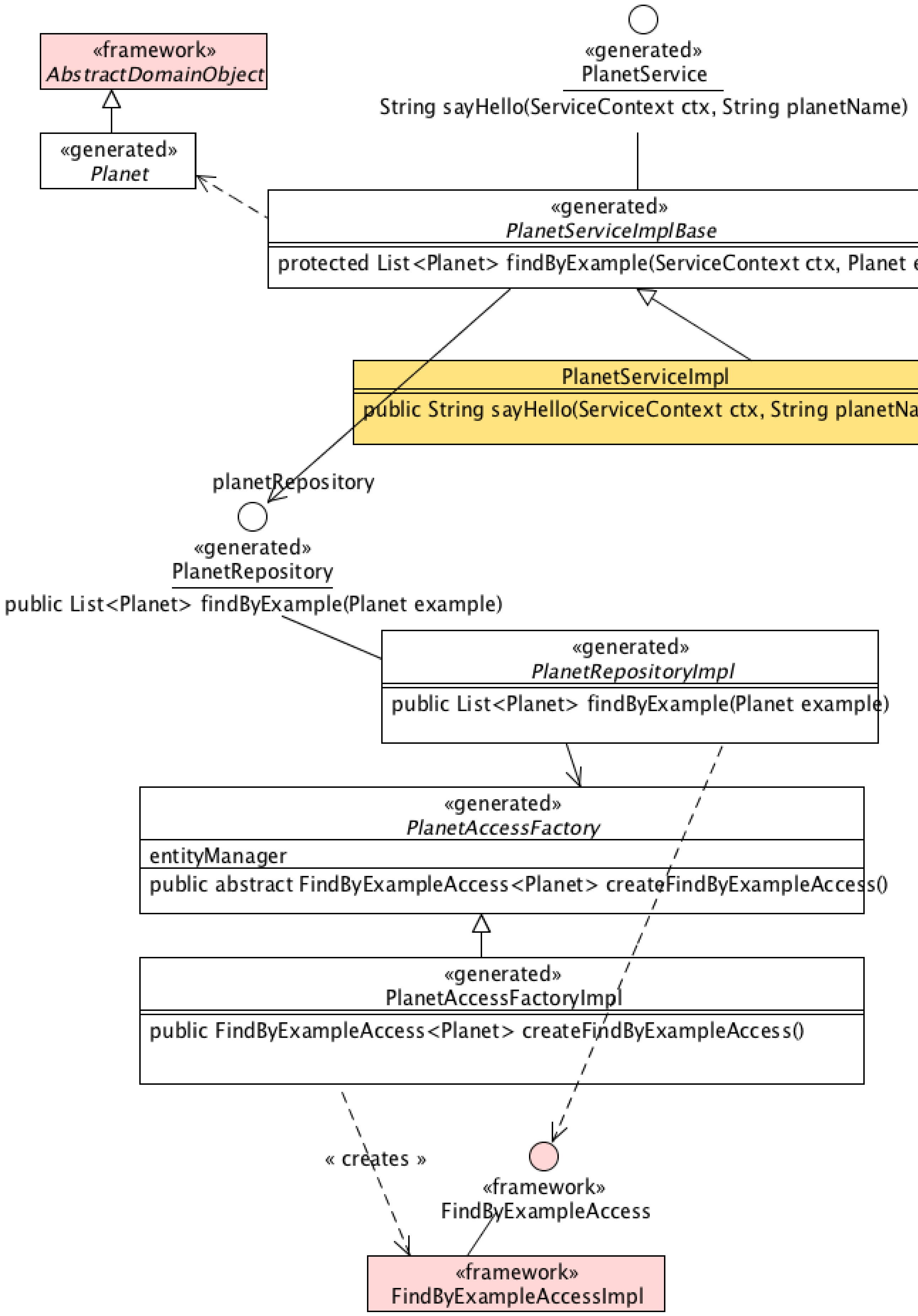
«framework»
AbstractDomainObject
«generated»
Planet
«generated»
PlanetService
String sayHello(ServiceContext ctx, String planetName)
«generated»
PlanetServiceImplBase
protected List<Planet> findByExample(ServiceContext ctx, Planet e
PlanetServiceImpl
public String sayHello(ServiceContext ctx, String planetNa
planetRepository
«generated»
PlanetRepository
public List<Planet> findByExample(Planet example)
«generated»
PlanetRepositoryImpl
public List<Planet> findByExample(Planet example)
«generated»
PlanetAccessFactory
entityManager
public abstract FindByExampleAccess<Planet> createFindByExampleAccess()
«generated»
PlanetAccessFactoryImpl
public FindByExampleAccess<Planet> createFindByExampleAccess()
« creates »
«framework»
FindByExampleAccess
«framework»
FindByExampleAccessImpl

Figure 2. Most important participating classes

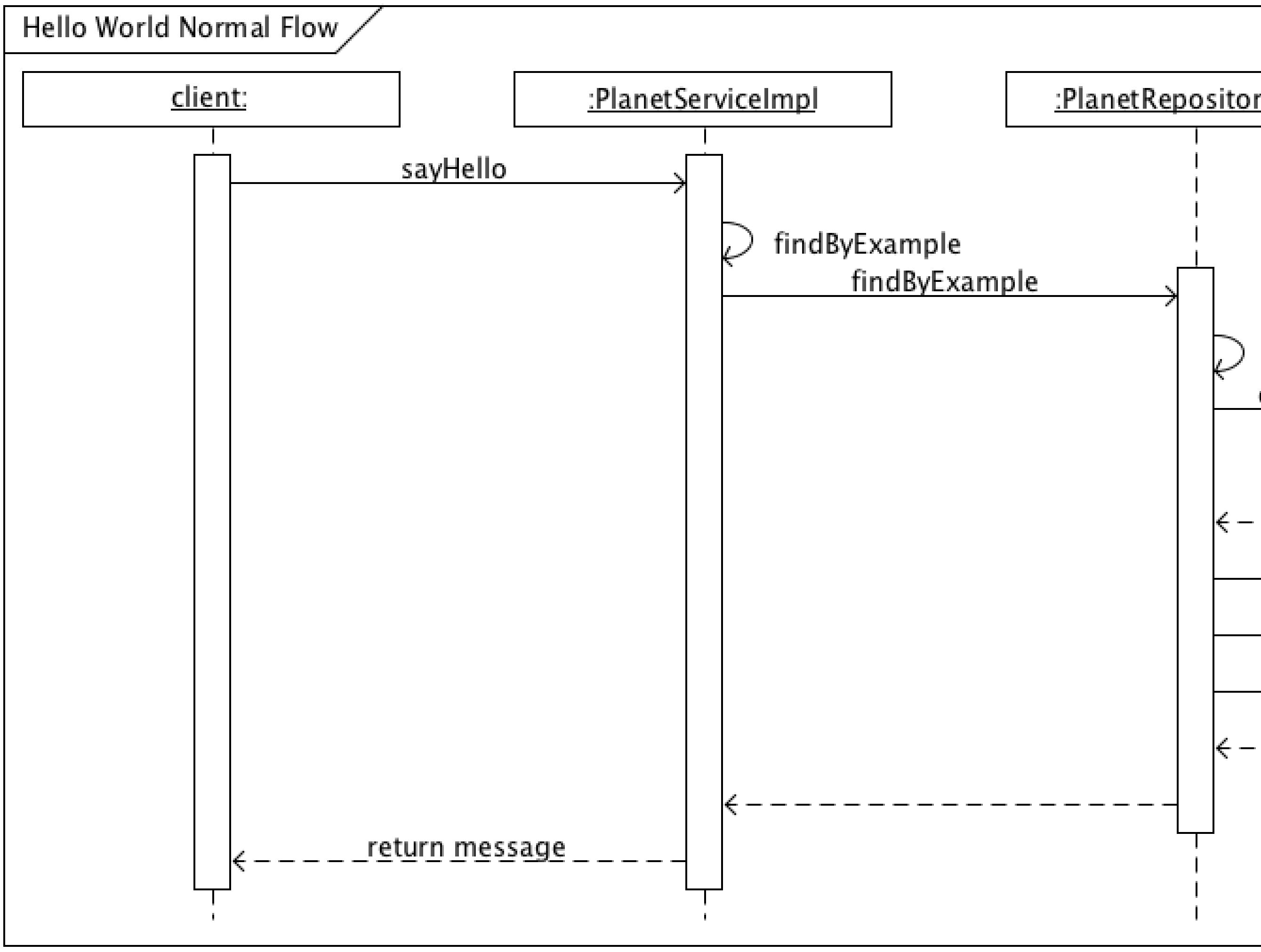

Figure 3. Normal flow for sayHello

Part 3 - Fix Failing Test

In this step we will fix the failing JUnit test and add some hand written code.

1. Run `PlanetServiceTest` as JUnit Test. Red bar.
Adjust the test method `testSayHello` to something like this:

```
public void testSayHello() throws Exception {
    String greeting = planetService.sayHello(getServiceContext(), "Earth");
    assertEquals("Hello from Earth", greeting);
}
```

2. HSQLDB is used as in memory database when running JUnit. Add test data in `src/test/resources/dbunit/PlanetServiceTest.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<dataset>
  <PLANET id="1" name="Earth" message="Hello from Earth"
    LASTUPDATED="2006-12-08" LASTUPDATEDBY="dbunit" version="1" />
  <PLANET id="2" name="Mars" message="Hello from Mars"
    LASTUPDATED="2006-12-08" LASTUPDATEDBY="dbunit" version="1" />
</dataset>
```

3. Run, still red, but another failure.

4. Implement method `sayHello` in `PlanetServiceImpl`.

```
public String sayHello(ServiceContext ctx, String planetName) {
    Planet planetExample = new Planet(planetName);
    List<Planet> foundPlanets = findByExample(ctx, planetExample);
    Planet planet = foundPlanets.get(0);
    return planet.getMessage();
}
```

5. Run. Green bar!

6. Add one more test method to test a failure scenario.

```
public void testSayHelloError() throws Exception {
    try {
        planetService.sayHello(getServiceContext(), "pluto");
        fail("Excpected PlanetNotFoundException");
    } catch (PlanetNotFoundException e) {
        // as expected
    }
}
```

7. Add `PlanetNotFoundException` in `model.btdesign`.

```
String sayHello(String planetName) throws PlanetNotFoundException;
```

8. Regenerate with

```
mvn -Dfornax.generator.force.execution=true generate-sources
```

9. Add `throws PlanetNotFoundException` in `PlanetServiceImpl.sayHello`.

10. Fix the import of PlanetNotFoundException in the test class and run it, red bar.

11. Fix the test. You need to adjust `sayHello` method.

```
public String sayHello(ServiceContext ctx, String planetName)
        throws PlanetNotFoundException {
    Planet planetExample = new Planet(planetName);
    List<Planet> foundPlanets = findByExample(ctx, planetExample);
```

```
        if (foundPlanets.isEmpty()) {
            throw new PlanetNotFoundException("Didn't find any planet named " + planetName);
        }
        Planet planet = foundPlanets.get(0);
        return planet.getMessage();
    }
```

12. Run. Green bar!

13. Run `mvn clean install`. Build success.

> You can use `mvn -o -npu install` to speed up the builds, -o == offline, -npu == no plugin upate.
> To regenerate you use `mvn -Dfornax.generator.force.execution=true -o -npu generate-sources`

Source

The complete source code for this tutorial is available in Subversion.

Web Access (read only):
http://fisheye3.cenqua.com/browse/fornax/trunk/cartridges/sculptor/sculptor-helloworld

Anonymous Access (read only):
https://fornax.svn.sourceforge.net/svnroot/fornax/trunk/cartridges/sculptor/sculptor-helloworld

2 3 4 5 6 7 8 9