Seam Validation Module

Reference Guide

Gunnar Morling

Introduction

The Seam Validation module provides CDI support for Hibernate Validator ...

Installation

This chapter describes the steps required to getting started with the Seam Validation Module.

2.1. Prerequisites

Not very much is needed in order to use the Seam Validation Module. Just be sure to run on JDK 5 or later, as the Bean Validation API and therefore this Seam module are heavily based on Java annotations.

2.2. Maven setup

The recommended way for setting up Seam Validation is using *Apache Maven* [http://maven.apache.org/]. The Seam Validation Module artifacts are deployed to the JBoss Maven repository. If not yet the case, therefore add this repository to your `settings.xml` file (typically in `~/.m2/settings.xml`) in order to download the dependencies from there:

Example 2.1. Setting up the JBoss Maven repository in settings.xml

```
...
<profiles>
    <profile>
        <repositories>
            <repository>
                <id>jboss-public</id>
                <url>http://repository.jboss.org/nexus/content/groups/public-jboss/</url>
                <releases>
                    <enabled>true</enabled>
                </releases>
                <snapshots>
                    <enabled>false</enabled>
                </snapshots>
            </repository>
        </repositories>
    </profile>
</profiles>

<activeProfiles>
    <activeProfile>jboss-public</activeProfile>
</activeProfiles>
...
```

General information on the JBoss Maven repository is available in the *JBoss community wiki* [http://community.jboss.org/wiki/MavenGettingStarted-Users], more information on Maven's `settings.xml` file can be found in the *settings reference* [???].

Having set up the repository you can add the Seam Validation Module as dependency to the `pom.xml` of your project. As most Seam modules the validation module is split into two parts, API and implementation. Generally you should be using only the types from the API within your application code. In order to avoid unintended imports from the implementation it is recommended to add the API as compile-time dependency, while the implementation should be added as runtime dependency only:

Example 2.2. Specifying the Seam Validation Module dependencies in pom.xml

```
...
<properties>
    <seam.validation.version>x.y.z</weld.version>
</properties>

...

<dependencies>
    ...
    <dependency>
        <groupId>${project.groupId}</groupId>
        <artifactId>seam-validation-api</artifactId>
        <version>${seam.validation.version}</version>
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>${project.groupId}</groupId>
        <artifactId>seam-validation-impl</artifactId>
        <version>${seam.validation.version}</version>
        <scope>runtime</scope>
    </dependency>
    ...
</dependencies>
...
```

Note

Replace "x.y.z" in the properties block with the Seam Validation version you want to use.

2.3. Manual setup

TODO GM: add correct links/file names

In case you are not working with Maven or a comparable build management tool you can also add Seam Validation manually to you project. Download the distribution file from http://..., un-zip it and add the JARs api and impl to the classpath of your project.

Dependency Injection

The Seam Validation module provides enhanced support for dependency injection services related to bean validation. This support falls into two areas:

- Retrieval of `javax.validation.ValidatorFactory` and `javax.validation.Validator` via dependency injection in non-Java EE environments
- Dependency injection for constraint validators

3.1. Retrieving of validator factory and validators via dependency injection

As the Bean Validation API is part of Java EE 6 there is an out-of-the-box support for retrieving validator factories and validators instances via dependency injection in any Java EE 6 container.

The Seam Validation module provides the same service for non-Java EE environements such as for instance stand-alone web containers. Just annotate any field of type `javax.validation.ValidatorFactory` with `@Inject` to have the default validator factory injected:

Example 3.1. Injection of default validator factory

```
package com.mycompany;

import javax.inject.Inject;
import javax.validation.Validator;
import javax.validation.ValidatorFactory;

public class MyBean {

    @Inject
    private ValidatorFactory validatorFactory;

    public void doSomething() {

        Validator validator = validatorFactory.getValidator();
        //...
    }
}
```

Note

The injected factory is the default validator factory returned by the Bean Validation bootstrapping mechanism. This factory can customized with help of the configuration file `META-INF/validation.xml`. The Hibernate Validator Reference Guide *describes in detail* [http://docs.jboss.org/hibernate/stable/validator/reference/en-US/html/validator-xmlconfiguration.html] the available configuration options.

It is also possible to directly inject a validator created by the default validator factory:

Example 3.2. Injection of a validator from the default validator factory

```
package com.mycompany;

import java.util.Set;

import javax.inject.Inject;
import javax.validation.ConstraintViolation;
import javax.validation.Validator;

public class MyBean {

    @Inject
    private Validator validator;

    public void doSomething(Foo bar) {

        Set<ConstraintViolation<Foo>> constraintViolations = validator.validate(bar);
        //...
    }
}
```

3.2. Dependency injection for constraint validators

The Seam Validation module provides support for dependency injection within `javax.validation.ConstraintValidator` implementations. This is very useful if you need to access other CDI beans within you constraint validator such as business services etc.

Warning

Relying on dependency injection reduces portability of a validator implementation, i.e. it won't function properly without the Seam Validation module or a similar solution.

To make use of dependency injection in constraint validators you have to configure `org.jboss.seam.validation.InjectingConstraintValidatorFactory` as the constraint validator factory to be used by the bean validation provider. To do so create the file `META-INF/validation.xml` with the following contents:

Example 3.3. Configuration of InjectingConstraintValidatorFactory in META-INF/validation.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<validation-config
                    xmlns="http://jboss.org/xml/ns/javax/validation/configuration" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://jboss.org/xml/ns/javax/validation/configuration validation-configuration-1.0.xsd">

   <constraint-validator-factory>
      org.jboss.seam.validation.InjectingConstraintValidatorFactory
   </constraint-validator-factory>

</validation-config>
```

Having configured the constraint validator factory you can inject arbitrary CDI beans into you validator implementions. Listing *Example 3.4, “Dependency injection within ConstraintValidator implementation”* shows a `ConstraintValidator` implementation for the `@Past` constraint which uses an injected time service instead of relying on the JVM's current time to determine whether a given date is in the past or not.

Example 3.4. Dependency injection within ConstraintValidator implementation

```
package com.mycompany;

import java.util.Date;

import javax.inject.Inject;
import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;
import javax.validation.constraints.Past;

import com.mycompany.services.TimeService;

public class CustomPastValidator implements ConstraintValidator<Past, Date>
{

   @Inject
   private TimeService timeService;

   @Override
   public void initialize(Past constraintAnnotation)
   {
   }

   @Override
   public boolean isValid(Date value, ConstraintValidatorContext context)
   {

      if (value == null)
      {
         return true;
      }

      return value.before(timeService.getCurrentTime());
   }

}
```

Note

If you want to redefine the constraint validators for built-in constraints such as `@Past` these validator implementations have to be registered with a custom constraint mapping. More information can be found in the *Hibernate Validator Reference Guide* [http://docs.jboss.org/hibernate/stable/validator/reference/en-US/html/validator-xmlconfiguration.html#d0e2024].

Chapter 4.

Method Validation

The Seam Validation module provides ...