Pepper

Developer's Guide for PepperModules

Florian Zipser <saltnpepper@lists.hu-berlin.de>
INRIA
SFB 632 Information Structure / D1 Linguistic Database
Humboldt-Universität zu Berlin
Universität Potsdam

Pepper: Developer's Guide for PepperModules

by Florian Zipser, , , , and

Version 1.1.4-SNAPSHOT

Table of Contents

Foreword

The aim of this document is to provide a helpful guide to create own PepperModules to plug them into the pepper framework. Currently this document is not in a very good state so far. It is not more than a loose collection of hopefully interesting information for developers. But are working on it, to fullfill its aim and to make a helpfull and readable guide out of it. You are very welcome to help us improving this document by reporting us bugs, requests for more information or by writing sections. Please write an email to `<saltnpepper@lists.hu-berlin.de>`.

Chapter 1. Documenting your PepperModule

One of the most important but often forgotten tasks when creating a PepperModule is to document the behaviour of it and its functionalities. Therefore the template SampleModules contains a template for creating a documentation. The documentation in SaltNPepper in general is done in DocBook (see docbook.org/). DocBook is a documentation language written in XML and enables to transform the documentation into several target formats like html, pdf, odt, doc etc..

Note

We recommend to use the template, for a uniform view to all PepperModules. That makes it possible not to forget important issues to be mentioned and makes it easier for the user to have a good understanding of what the PepperModule is doing.

You will find the template in `SAMPLE_MODULES_HOME/src/main/docbkx/manual.xml` among other directories containing files necessary for the transformation. Tor refer to images from the manual, put the into the image folder `SAMPLE_MODULES_HOME/src/main/docbkx/images/` and make a relative reference.

1. Transformation

The standard transformation which is configured for SampleModules is the transformation to html and pdf. The configuration is done in the `pom.xml` of the pepper-parentModule project. To add further output formats, just copy the respective plugins to your pom and change them.

When executing the maven goal site

```
mvn clean site
```

, maven will create a manual folder under `SAMPLE_MODULES_HOME/target/manual`, where you can find the pdf documentation and the html documentation.

Note

The current configuration does not need to be changed, just write your documentation by overriding the template. The rest shall work automatically. In some cases it might be necessary to adopt the transformation, than please take a look to the xsl transformation files in `SAMPLE_MODULES_HOME/docbook-xsl`.

Chapter 2. Customizing behaviour of your PepperModule

via properties

When creating a mapping inside to either map any format or model to salt or a salt model to another salt model or a salt model to any model or format, it is often a matter of choice to map some data this way or another. In such cases it might be clever not to be that strict and allow only one possiblity. It could be a good idea to leave this decision to the user. Customizing a mapping will increse the power of a PepperModule enormously, because than it can be used for many purposes without rewriting parts of it. Therefore the pepper framework provides a property system to access such user customizations. Nevertheless, a PepperModule shall not be dependant on user customization, the past showed, that it is very frustrating, when a PepperModule breaks up, because of not specifified properties. There should always be a default behaviour in case of the user has not specified one.

1. Property

A property is just an attribute-value pair, consisting of a name so called property name and a value so called property value. Properties can be used for customizing the behaviour of a mapping of a PepperModule. Such a property must be specified by the user and determined in the pepper workflow description. The pepper frsamework will pass all customization properties direct to the instance of the PepperModule.

Note

In the current version of pepper one has to specify a property file by its location in the pepper workflow description file (.pepperParams) in the attribute @specialParams inside the <importerParams>, <exporterParams> or <moduleParams> element. In the next versions this will change to a posibility for adding properties directly to the pepper workflow description file.

2. Property registration

The pepper framework provides a kind of a registration for customization properties. This registry is called `PepperModuleProperties` and can be accessed via `getProperties()` and `setProperties()`. This class only represents a container object for a set of `PepperModuleProperty` objects and provides accessing methods. An instance of `PepperModuleProperty` represents an abstract description of a property and the concrete value at once. In the registration phase it belongs to the tasks of a `PepperModule` to specify the abstract description which consists of the name of the property, its datatype, a short description and a flag specifying if this property is optional or mandatory. To create such an abstract description of a property use the constructor:

```
PepperModuleProperty(
                        String
                        name
                    ,

                        Class>T<
                        clazz
                    ,

                        String
                        description
                    ,

                        Boolean
                        required
                    );
```

and pass the created property object to the property registry by calling the method `addProperty`. The pepper framework uses the registry to first inform the user about usable properties for customization and second to fullfill the property objects with the property values set by the user.

The value of a specific property can be accessed by passing its name to the registry. The method to be used is the following one:

```
getProperty(
                        String
                        propName
                    );
```

Chapter 3. Testing your PepperModule

For running your own PepperModule in a test environment, the pepper framework provides a special project for doing this called pepper-testSuite. This project first contains an environment to run your module called pepper-testEnvironment and contains a second project called pepper-moduleTest for checking the correctness of your PepperModule. Correctness means, that your module can be pluged into the environment and does not mean the logical correctness of its functionality. This project directly runs in the OSGi container and therefore has to be started in it. Because of there is starter outside the OSGi environment, you have to pass necessary resource locations and test corpora via environment variables.

1. environment variables

- PEPPER_TEST to a folder, where the resources of the peppermodules are and the temprorary stuff can be stored
- PEPPER_TEST_WORKFLOW_FILE to the .pepperparams file

2. Configure OSGi environment

For running pepper and also its test environment it is very important to use the correct start order for plugins. For instance it is of special interest to start a logger as early as possible to get a logging and not to hide important messages like warnings or errors. For pepper it is important to run the the pepper framework after all plugins are started, otherwise, they will not be registrated to the pepper plugin registry and can therefore not be resolved.

To avoid this, set the start level (in OSGi) of "pepper-framework" to default-value + 2 and set the start value of the pepper-logReader to 0.

To make it easier, we provide a preconfigured file, which can be used for this, but only for eclipse. This file contains information for the run configurations of the pepper-testEnvironment project and can be found under `PEPPER_TEST_ENVIRONMENT_HOME/pepper-testEnvironment.launch`. When using this configuration, it will not run correcly out of the box, because you have to enable your project first. Therefore open the run configuration in eclipse open the pepper-testEnvironment entry and click the box left to the name of your project. For checking not to forget one necessary dependency you can click the vlidate button and the eclipse will retrieve all dependencies and list missing ones in case they exist.