The AbstractUsecase Class

Table of contents

1. Introduction

When you implement a custom usecase, you're very likely to extend `org.apache.lenya.cms.usecase.AbstractUsecase`. This class provides a set of functionality to simplify the implementation of usecases.

2. Configuration

The usecase is configured in `cocoon.xconf`. A typical configuration looks like this:

```
<component-instance name="edit.forms" logger="lenya.publication"
                    class="org.apache.lenya.cms.editors.forms.FormsEditor">
  <transaction policy="pessimistic"/>
  <view template="edit/forms/forms" menu="false"/>
</component-instance>
```

The following configuration options are available:

- Element `<transaction>` (optional)
  - Attribute `policy = (optimistic | pessimistic)`

  This element is used to determine the transaction policy of the usecase (*optimistic* or *pessimistic offline lock*). It can be omitted, the default is optimistic behaviour. You should only use pessimistic behaviour for complex usecases when the user can't afford to lose all changes.
- Element `<view>` (optional)
  - Attribute `template` (required)
  - Attribute `menu = (true | false)` (optional)

  This element declares the view of the usecase. The `template` attribute points to the JX template to use, relatively to the `lenya/usecases` directory. The suffix `.jx` is added automatically. The attribute `menu` determines if the menu should be visible. It can be omitted and defaults to `false`.

3. Extending AbstractUsecase

The following methods of the `AbstractUsecase` class are meant to be overridden:

- `protected void initParameters()`

  This method is called to initialize the parameters of the usecase. For instance, if your usecase shall display meta data on the view screen, `initParameters()` reads the meta data from the document and puts them into the parameter map using `setParameter(String, Object)` to make them available to the JX template.
- `protected void doCheckPreconditions()`

  The method `checkPreconditions()` is a template method which calls this method. For details on `checkPreconditions()`, see section Overview (index.html) .
- `protected void doCheckExecutionConditions()`

  The method `checkExecutionConditions()` is a template method which calls this method. For details on `checkExecutionConditions()`, see section Overview (index.html) .
- `protected void doCheckPostconditions()`

  The method `checkPostonditions()` is a template method which calls this method. For details on `checkPostonditions()`, see section Overview (index.html) .
- `protected void doExecute()`

  The method `execute()` is a template method which calls this method. For details on `execute()`, see section Overview (index.html) .

- `public void advance()`

  For details on `advance()`, see section Overview (index.html) .
- `protected Transactionable[] getObjectsToLock()`

  This method is supposed to return all objects which should be locked before the usecase is started. If the transaction policy is *pessimistic offline lock*, these objects are checked out immediately.