

eHealth Framework

Customizing an Application

Notice

The wording in this document applies equally to women and men. The masculine form was selected to ease the comprehensibility and legibility of the text.

All company logos are a registered trademark of InterComponentWare AG.

The product names mentioned in this documentation are either trademarks or registered trademarks of the respective owners and are stated for identification purposes only.

This documentation and the software components are protected by copyright © 2006-2009 InterComponentWare AG.



Note:

The current version of this document has a draft status and various chapters are still in review.

The document is collaboratively built with the use of the Darwin-Information-Typing-Architecture (DITA) and has therefore a draft status concerning styles and layout. The necessary adaptations are currently also in a developmental stage.

All rights reserved.

Contents

1 Overview	1
2 Customizing an Application	2
2.1 Creating a new Customization Project	2
2.2 Adding a Config Artifact	4
2.3 Adding a Runtime Artifact	5
2.4 Adding a Property	6
3 References	8

1 Overview

Purpose

The purpose of this howto is to provide a step-by-step guide of how an eHF-based application can be customized. This document leads the reader beginning with an empty directory all the way towards using various features of the eHF Customization Plugin.

Scope

The current scope of this howto covers selected use cases. In order to get a complete overview on the eHF Customization Plugin consult the [eHealth Framework Reference Documentation](#) on page 8.

2 Customizing an Application

2.1 Creating a new Customization Project

The application that needs to be customized is available.

For customizing an existing application a new customization project is required.

1. Open a console window

On windows operating systems this can be accomplished using the classic start menu and selecting the `Run . . .` menu item. In the dialog you simply have to type `cmd` and a Windows console will be presented.

On unix-based operating systems you open a new shell or console.

2. Change into your workspace directory

On windows operating systems this can be accomplished by first changing to the drive the workspace is located. E.g. this is the C drive you type `c :`. Then you simply use the `cd <foldername>` command substituting the `foldername` with the name of the folder or a complete path until you are inside the workspace folder.

On unix-based operating systems you normally do not have to deal with different drives. They should have been mounted using an appropriated and known alias.

3. Create the customization project structure using maven genapp.

Execute the following command in the workspace directory:

```
maven genapp
  -Dmaven.genapp.template.dir=<project-templates>/ehf-customization-template
```

(where `<project-templates>` is replaced with the path to the ehf-project-templates project. This path can either be expressed as a relative or absolute path.)

Maven prompts for further information.

4. Please specify the project root directory

The root directory is the directory the project will be created in. The default is not useful in this case as it points directly to your workspace. Specify the name of the folder here that should be used. The folder can be relative and must i.e. not provide an absolute path.

5. Please specify an id for your project

The id has many purposes. It will be the `id` of the project as well as the artifact id for distribution.

6. Please specify a groupId for your application

The `groupId` is propagated in the project description. It is also used to define the group the final artifact will go to in the maven repository.

7. Please specify a groupId for your application

The `groupId` is propagated in the project description. It is also used to define the group the final artifact will go to in the maven repository.

8. Please specify the version of eHF you would like to use:

This version will be used for the ehf dependencies of your project.

The initial project structure for the module is generated. The configuration files contain default configuration. The initial model is empty.

9. Please specify a artifact id for the customization target

This is the `id` of the application that needs to be customized. This parameter, together with the next three parameters are used for defining the dependency to the artifact in the `project.xml`.

10. Please specify a group for the customization target

This is the `group id` of the application that needs to be customized.

11. Please specify a version for the customization target

This is the `version` of the application that needs to be customized.

12. Please specify a customization tag

This tag would be appended to the name of the resulting archive.

13. Please specify a name for the customization folder

This parameter could be used to define a subfolder within the customization project folder, where you could place all your customized files.

14. Import the project into the IDE

Finally, you can import the project into your IDE. Maven provides plug-ins for different IDEs. For eclipse, e.g., run

```
maven eclipse:eclipse
```

. This will create the Eclipse specific `.classpath` and `.project` configuration files. Open Eclipse and import the project using the **New Project Wizard**.

The module can be developed using an eclipse-based IDE.

15. Specify the customization target

The customization project template has introduced a default customization target in the configuration that was produced by genapp. The following configuration can be found in the `project.xml` project description file.

```
<dependency>
  <groupId>ehf</groupId>
  <artifactId>ehf</artifactId>
  <version>SNAPSHOT-bin</version>
  <type>zip</type>
  <properties>
    <ehf-customization-target>true</ehf-customization-target>
  </properties>
</dependency>
```

The above can be customized easily by specifying another artifact that has to function as customization target. In principle the attributes `groupId`, `artifactId` and `version` can be substituted to identify the artifact of the application that is targeted to be customized.

16. Configuring the customization

Further - as the target was changed in the last step - also the `project.properties` file has to be adapted. The genapp template has produced the following default configuration:

```
# customization target (specifies the target archive that is customized)
maven.ehf.customization.target=ehf-SNAPSHOT-bin

# customization tag (included in the name of the result archive)
maven.ehf.customization.tag=CUSTOMER-LANG

# convenience property to ease configuration
ehf.module.version=SNAPSHOT
```

```
# list of properties that can be used by the customization (extensible)
customize.product=ehf-*-bin.zip/ehf-*-bin
customize.product.modules=ehf-*-bin.zip/ehf-*-bin/modules
customize.product.lib=ehf-*-bin.zip/ehf-*-bin/lib
customize.product.war=ehf-*-bin.zip/ehf-*-bin/artifacts/ehf.war
customize.product.war.lib=ehf-*-bin.zip/ehf-*-bin/artifacts/ehf.war/WEB-INF/
lib

# sequence for specifying archived substructures that need to be extracted/
repackaged
ehf.maven.customization.archive.sequence=\
    ${maven.ehf.customization.target}.zip

# sequence of customizations to be applied
ehf.maven.customization.sequence=\
    customization-example
```

In especially the properties `maven.ehf.customization.target` and the `customize.product.*` properties must be adapted. Please note that these properties can be extended as required.

The properties `ehf.maven.customization.archive.sequence` and `ehf.maven.customization.sequence` are not of interest at this point in time and will be described later.

17. Applying the customization

Now that the project is in place the customization can be applied by executing the command

```
maven customize:all
```

The `customize:all` is composed of three goals that are executed in a sequence:

- a. `customize:prepare`: gathers all the data required for the customization. Furthermore the archive structure is prepared for the customization according to the `ehf.maven.customization.archive.sequence`.
- b. `customize:execute`: applies all customization according to the `ehf.maven.customization.sequence`.
- c. `customize:finalize`: repackages the archive by inverting the `ehf.maven.customization.archive.sequence`.

The customized archive can be found in the `target` folder. At the moment no particular customization is applied. The example customization simply adds a file named `test.txt` to the root directory of the application.

2.2 Adding a Config Artifact

The customization project is initiated as described above.

A configuration artifact - i.e. from a service module - must be included in an application.

1. Introduce a dependency to the artifact

Include the following dependency into the `project.xml`:

```
<dependency>
  <groupId>pg</groupId>
  <artifactId>pg-medicine-cabinet-config</artifactId>
  <version>SNAPSHOT</version>
  <properties>
    <ehf-artifact>config</ehf-artifact>
  </properties>
  <type>jar</type>
```

```
</dependency>
```

It is required that the `groupId`, `artifactId` and the `version` is substituted to match the config artifact that is required to be included in the application.

The introduction of the dependency represents an implicit definition of a customization.

The goal `customize:prepare` creates an implicit customization in the folder `target/extension/pg-medicine-cabinet-config-SNAPSHOT` if the above example is assumed.

2. Configuring the Customization

In order to activate the customization that is implicitly defined by the new dependency the property `ehf.maven.customization.sequence` in the `project.properties` file requires to be extended. In the above example the file could look as follows:

```
ehf.maven.customization.sequence=\
target/extension/pg-medicine-cabinet-config-${ehf.module.version}
```

All required information for adding the configuration artifact is now available.

3. Executing the customization

Rerun the customization using the command

```
maven customize:all
```

The customization is applied to the application and the required information for installing the application is extended by the configuration contained in the configuration artifact.

2.3 Adding a Runtime Artifact

The customization project is initiated as described above. Normally this task is always in combination of adding a config artifact. Therefore it will also be based on the result from the adding a config artifact task.

A runtime artifact - i.e. from a service module - must be included in an application.

1. Introduce a dependency to the artifact

Include the following dependency into the `project.xml`:

```
<dependency>
  <groupId>pg</groupId>
  <artifactId>pg-medicine-cabinet-runtime</artifactId>
  <version>SNAPSHOT</version>
  <properties>
    <ehf-artifact>runtime</ehf-artifact>
  </properties>
  <type>jar</type>
</dependency>
```

It is required that the `groupId`, `artifactId` and the `version` is substituted to match the runtime artifact that is required to be included in the application.

The introduction of the dependency represents an implicit definition of a customization.

The goal `customize:prepare` creates an implicit customization in the folder `target/extension/pg-medicine-cabinet-runtime-SNAPSHOT` if the above example is assumed.

2. Configuring the customization

In order to activate the customization that is implicitly defined by the new dependency the property `ehf.maven.customization.sequence` in the `project.properties` file requires to be extended. In the above example the file could look as follows:

```
ehf.maven.customization.sequence=\
    target/extension/pg-medicine-cabinet-config-${ehf.module.version},\
    target/extension/pg-medicine-cabinet-runtime-${ehf.module.version}
```

All required information for adding the runtime artifact is now available.

3. Executing the customization

Rerun the customization using the command

```
maven customize:all
```

The customization is applied to the application and the runtime artifact is added to the archive.

2.4 Adding a Property

The customization project is initiated as described above. The archive contains a property file that needs to be extended by one or more properties.

In order to customize the application the addition of a property is required.

1. Introduce a dedicated customization folder

Create a customization folder named for example `pg-pangreas` in the root directory of the customization project.

An empty customization folder is available.

2. Configure the merge property action

A customization at minimum consists of a folder in the customization project containing a `customization.properties` file. The file can be used to control certain property actions. Please consult the eHF Reference Documentation for a complete list of possible customization actions.

The `customization.properties` file should for example look as follows:

```
# description of the customization
customization.description=eHF Medicine Cabinet Glue Configuration

# pattern property for controlling the copy action
customization.copy.include=-nothing-
customization.copy.exclude=-nothing-

# pattern property for controlling the delete action
customization.delete.include=-nothing-
customization.delete.exclude=-nothing-

# pattern property for controlling the merge property action
customization.property.include=**/*.properties
customization.property.exclude=-nothing-

# pattern property for controlling the transform action
customization.transform.include=**/*.xsl
customization.transform.exclude=-nothing-

# comma separated list of targets
customization.targets=${customize.product}
```

3. Define the customization files

The patterns controlling the merge property actions have to be understood as relative to a `files` folder that has to be located in the same directory as the `customization.properties` file. Most of the actions (except e.g. the delete action) require the `files` folder.

For example the `files` folder contains a file named `configuration.password.properties`. The customization file will try to detect a file with the same name in the target folder and merge the two files.

The file may look like this:

```
connection.medicine-cabinet.pass=initinit
```

All information for adding the a property during the customization is provided in the dedicated customization folder. This folder contains a `customization.properties` file and a folder `files` containing a property file to be merged.

4. Executing the customization

Rerun the customization using the command

```
maven customize:all
```

The customization is applied to the application. The property is merged into the target file.

3 References

E

ICW eHealth Framework Reference Documentation

BAS Technology and eHF Committers, InterComponentWare AG (2008-2009)

<http://idn.icw-global.com/downloads.html> ↗