Adding Custom Templates

Summary

The eGovFrame lets you create your own templates to use. Since the Code Generation is a plug-in tool, it can be extended by anyone to create different types of templates and use them in building projects.

Description

eGovFrame’s Code Gen uses Velocity Template Engine by default. In order to create new templates the following are required.

* Template: Create templates based on Velocity Template syntax.
* Wizard XML File: XML file for wizard UI.
* Wizard list XML File: XML file for template list.
* Plug-in project: leverages extendability of eGovFrame Code Gen., and comprises of Template, Wizard XML, and Wizard list XML.
* Feature, Update site project: Update the plug-in projects in batches.

Creating templates

See the website below on how to create Velocity templates.

http://wiki.apache.org/velocity/FrontPage [http://wiki.apache.org/velocity/FrontPage]

How to create wizards

Wizards are created using XML syntax, and is as follows.

Composition

There are model-pages and component-pages for custom template wizard pages, and they let you create one or more pages.

Model-page: database table selection

Component-page: manually enter individual components

Output: describes template file mapping informatino

Creating model pages

<model-page type="table" required="true">

<!-- SelectSomeTable

choose a table -->

<description>Select some Table</description>

</model-page>

Examples of component pages

<component-page>

<description>Input resource information</description>

<group label="" required="true">

<textfield name="author" label="Author:" required="true" value="" />

<container name="voPackage" label="VO Package: " required="true" type="package" />

</group>

</component-page>

(Description)

* Component-page: create one or more <component-page> XML code for

each wizard page

* Label: title of the wizard dialog
* Name: internal component name
* Required: true or false
* Value: default value if any

Creating UI input components

Group: Groups one or more UI components

<group name="" label="" checkbox="" >

...

</group>

(Descripton)

* Label: title of the wizard dialog
* Checkbox: apply checkboxes to components within groups

Container: enables choosing packages or folders

<container name="actionPackage"

label="Package:"

required="true"

type="package|folder"

value="" />

(Definition of types above)

* Package: enables choosing Java package list for current project
* Folder: enables choosing resources folder for current project

textField: text input field

<textfield name=""

label=""

required="true/false"

extension="" />

checkbox:

<checkbox name="createValidation"

label="Create the Action-validation.xml ?"

value="false" />

radio:

<combo name="radio" label="View:" value="FreeMarker">

<elements>

<option value="freemarker">FreeMarker</option>

<option value="velocity">Velocity</option>

<option value="dispatcher">JSP</option>

<option value="xslt">XSLT</option>

</elements>

</combo>

combo:

<combo name="radio" label="View:" value="FreeMarker">

<elements>

<option value="freemarker">FreeMarker</option>

<option value="velocity">Velocity</option>

<option value="dispatcher">JSP</option>

<option value="xslt">XSLT</option>

</elements>

</combo>

Example output

- In order to generate code, <output> needs to be set so that inputs from the wizard dialog will be mapped to appropriate templates for it.

<output>

<template component="txtFileName" velocity="sequenceId.vm"

extension="xml" container="txtPath">

</template>

<condition variable="checkService" value="true" />

</output>

(Description)

Template: creates one or more <template> XML codes for each files that need to be created

Component: component name

Velocity: Velocity template to be used

Extension: default file extension to be used

Container: directory path for files to be saved

Condition: for conditional code generation (if “variable” equals this “value”, will generate code)

How to use database table information in wizards/templates

Following is how to use database tables to generate code.

${model.vendor}: database vendor type

${model.entity.name}: table name

${model.entity.pcName}: Pascal-style table name (EGOV\_SAMPLE →

EgovSample)

${model.entity.ccName}: Camel-style table name (EGOV\_SAMPLE →

egovSample)

${model.entity.ucName}: table name in all upper-case

${model.entity.lcName}: table name in all lower-case

${model.attributes}: column names

${model.primaryKeys}: primary keys

${model.attributes.get(0)}: first column

${model.attributes.get(0).name}: name of first column

${model.attributes.get(0).javaType}: Java type of first column

Example

Following is an example of how to use database tables for generating code.

<output>

<template component="dao" expression="${model.entity.pcName}DAO"

velocity="java/pkg/service/impl/Sample2DAO.vm" extension="java"

container="daoPackage">

<condition variable="checkDataAccess" value="true" />

</template>

</output>

Following is an example of how to generate database queries using templates

SELECT

#set($i=0)

#foreach($attribute in $model.attributes)

#if($i == 0)

${attribute.name}

#else

, ${attribute.name}

#end

#set($i=$i+1)

#end

FROM ${model.entity.name}

ORDER BY

#set($i=0)

#foreach($attribute in $model.primaryKeys)

#if($i == 0)

${attribute.name} DESC

#else

, ${attribute.name} DESC

#end

#set($i=$i+1)

#end

Following is an example of how to generate other query types using database vendor information.

#set($vender="$model.vender")

#if($vender == "HSQLDB")

.....

#elseif($vender == "Oracle")

.....

#end

Creating wizard lists

Wizard lists per category types can be defined as below, and the template path needs to specify relative path of the wizard list XML file.

Example

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

<templates>

<category name="CRUD">

<wizards>

<wizard-def description="CRUD Program" template="crud/wizard.xml"/>

</wizards>

</category>

</templates>

How to create custom template plug-in projects

1. Create a plug-in project, and refer to examples above to create a template, wizard, and wizard list file.

2. Open MANIFEST.MF using Plug-in Manifest Editor, then click the Extensions tab.

3. Click the Add… button, then double-click the egovframework.dev.imp.codegen.template.templateWizards to add to the Extensions.

4. Enter Extensions details. Choose an ID, name, and wizard list XML path.

<Edited plugin.xml>

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

<?eclipse version="3.4"?>

<plugin>

<extension

point="egovframework.dev.imp.codegen.template.templateWizards">

<wizards

id="CustomTemplates.wizards"

name="Custom Templates"

wizardsFile="templates/wizards.xml">

</wizards>

</extension>

</plugin>

5. Create a Feature project and an Update site project, and package your template plug-in.

6. Update your IDE with your template plug-in by choosing Help > Software Updates > Add Site ... > Update site address or file from the menu.

7. Template listing: Choose Window>Show View>eGovFrame Templates from the menu.

8. Test the custom template.

Manual

Following is an example of custom template project source code.

Refer to the manual and create any kind of template you wish to use.

Custom template project source code

Download:

CustomTemplateSample.zip

(Description)

The sample project consists of the following three project.

CustomTemplates: custom template project

CustomTemplates.feature: custom template feature project

CustomTemplates.update: custom template update site project

All three projects can be bundled together to become a single plug-in for loading and updating purposes.

Custom template update file

Download:

CustomTemplate.update.zip

(Description)

The example build is for updating custom templates.

1. Download the file, and choose Help > Software Updates from the menu.

2. Click the Add Site... button on the top right from the Available Software tab.

3. Click the Archive... button on the right from the Add Site dialog.

4. Choose the downloaded file, then click OK.

5. Choose Custom Template from the Available Software screen, then click Install.

6. Afterwards, restart the plug-in, and list the templates to check that the custom template was added correctly.

Tips on how to create custom templates

If you are not used to manipulating plug-in project, there is an alternative way as below.

1. Create a Velocity template and a wizard files.

2. Save the files into the plugins folder:

Location: \plugins\egovframework.dev.imp.codegen.template.templates\_<version

>\eGovFrameTemplates\

3. wizards.xml: Modify the wizards list file to add custom template information by referring the earlier sections in this document.

4. Template listing: Choose Window>Show View>eGovFrame Templates from the menu.

5. Confirm that the custom template has been added and is working properly.

6. However, one downside is that after every version update, the file gets reset, so you need to repeat these steps again manually.