PTC® the product development company

WC10 Customizing

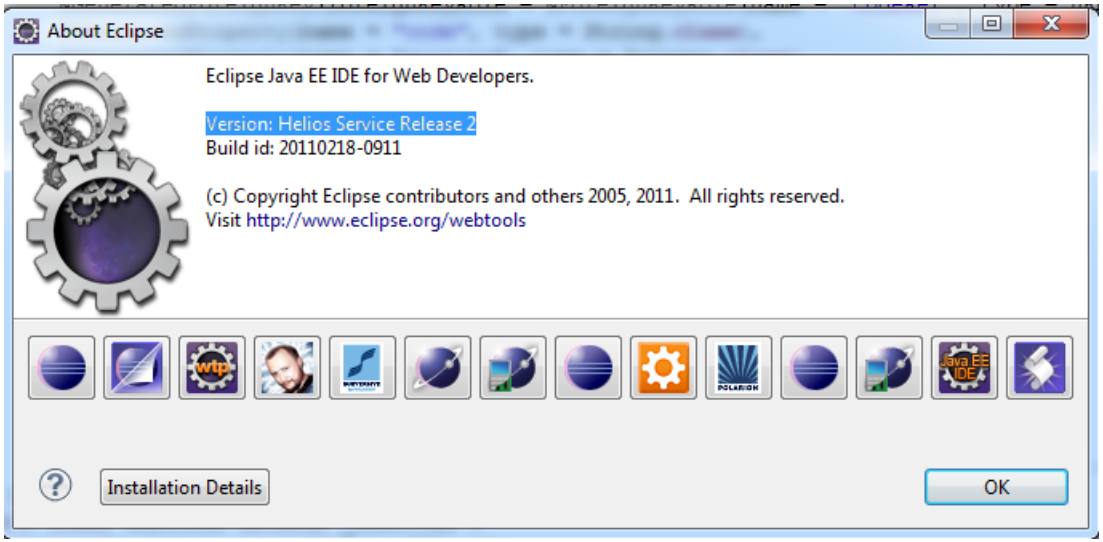
Eric Kim(yckim@ptc.com)

Eclipse configuration

1. Download & install eclipse

Version list and download site

- Eclipse (Version: Helios 3.6 R2)
  - http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/helios/SR2/eclipse-jee-helios-SR2-win32-x86_64.zip - Windows 64Bit
  - http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/helios/SR2/eclipse-jee-helios-SR2-win32.zip - Windows 32Bit

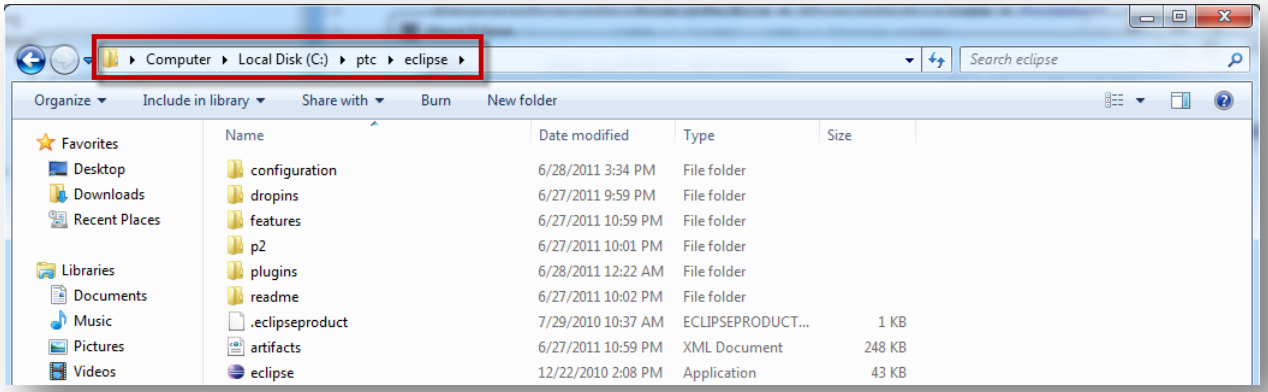


  - We need to install add-on program for WC10.
    - Aptana
    - FileSync
    - SVN Subversive

1. Download & install eclipse

Install Eclipse

- Extract compress file and copy to any directory
  - Eclipse doesn't need to execute install program, you just copy comfortable directory



- Make shortcut and set virtual memory for performance
  - Open INI file in same directory of eclipse.exe
  - Update memory size for eclipse

```
-startup
plugins/org.eclipse.equinox.launcher_1.1.1.R36x_v20101122_1400.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.2.R36x_v20101222
-product
org.eclipse.epp.package.jee.product
--launcher.defaultAction
openFile
--launcher.XXMaxPermSize
256M
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
256m
--launcher.defaultAction
openFile
-vmargs
-Dosgi.requiredJavaVersion=1.5
-Xms512m
-Xmx512m
```

2. Set environment for eclipse

PTC®

Make sure eclipse environment for performance and correct excuting

1. Set Java path for eclipse

- ✓ Add Java path $WT_INSTALL_PATH/Java/bin to system path

2. Set memory size for eclipse

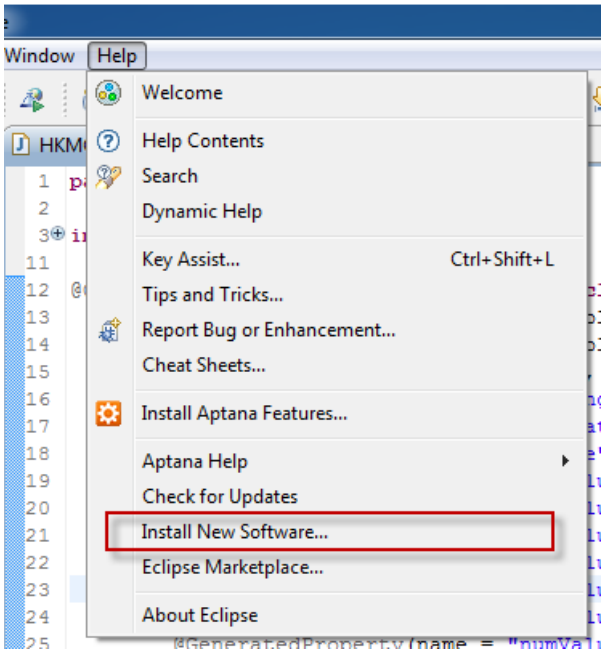
- ✓ Change directory to installed eclipse
- ✓ Open eclipse.ini file for editing mode.
- ✓ Change correct memory size on INI file.

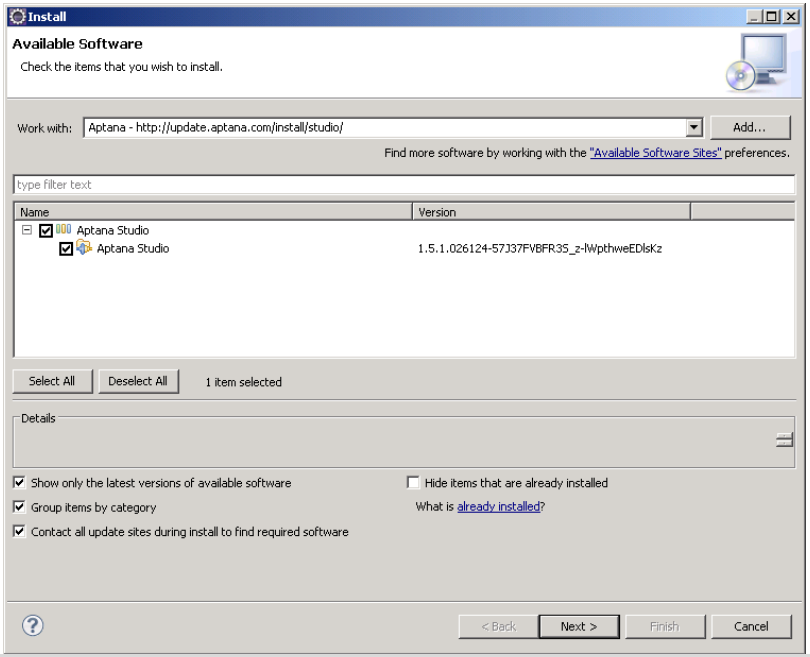
```
-startup
plugins/org.eclipse.equinox.launcher_1.1.1.R36x_v20101122_1400.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.2.R36x_v20101222
-product
org.eclipse.epp.package.jee.product
--launcher.defaultAction
openFile
--launcher.XXMaxPermSize
512M
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
512m
--launcher.defaultAction
openFile
-vmargs
-Dosgi.requiredJavaVersion=1.5
-Xms40m
-Xmx512m

```

3. Install Aptana

Install Aptana using eclipse

- Aptana makes updating ExtJS simple
  - Similar to how Eclipse simplifies Java coding
  - Install using Eclipse
  - **Help > Install New Software ...**
  - Point to the Aptana URL to install this Eclipse
  - Plug-in
  - Editing ExtJS on a Web Page with Aptana





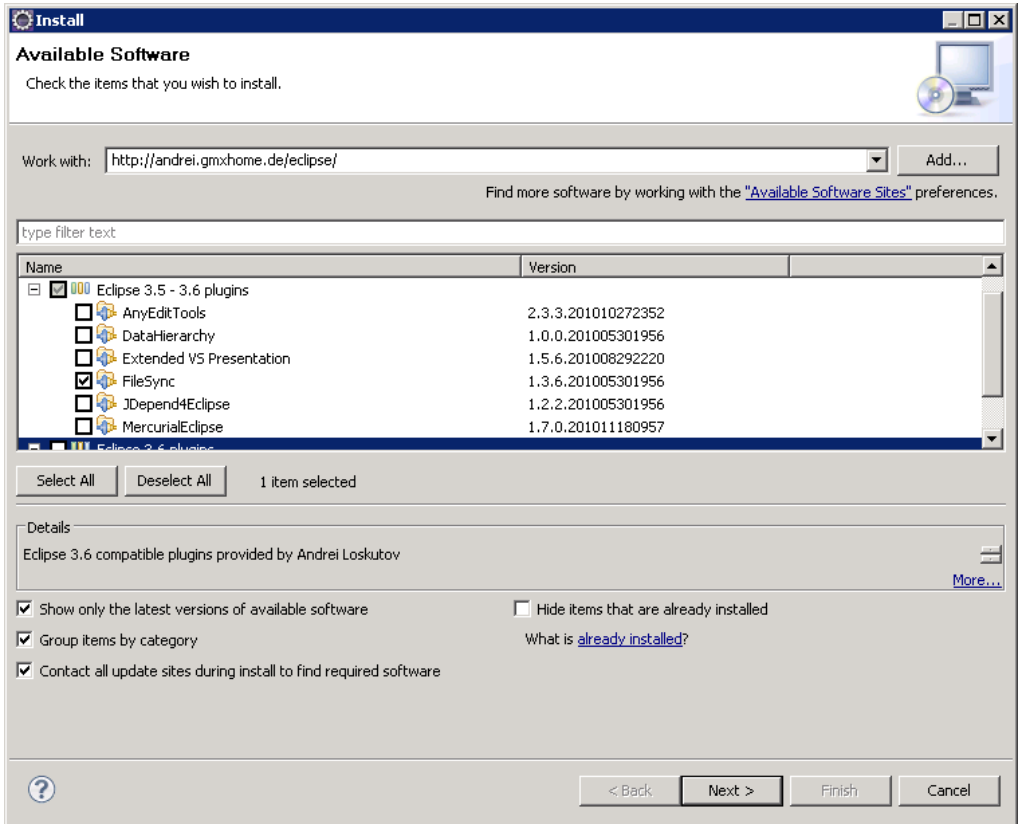
Although pages on the web will indicate an older URL, the correct URL is http://download.aptana.com/tools/studio/plugin/install/studio

The older version will display an error when Eclipse starts.

4. Install FileSync

FileSync is used to Synchronize Directories

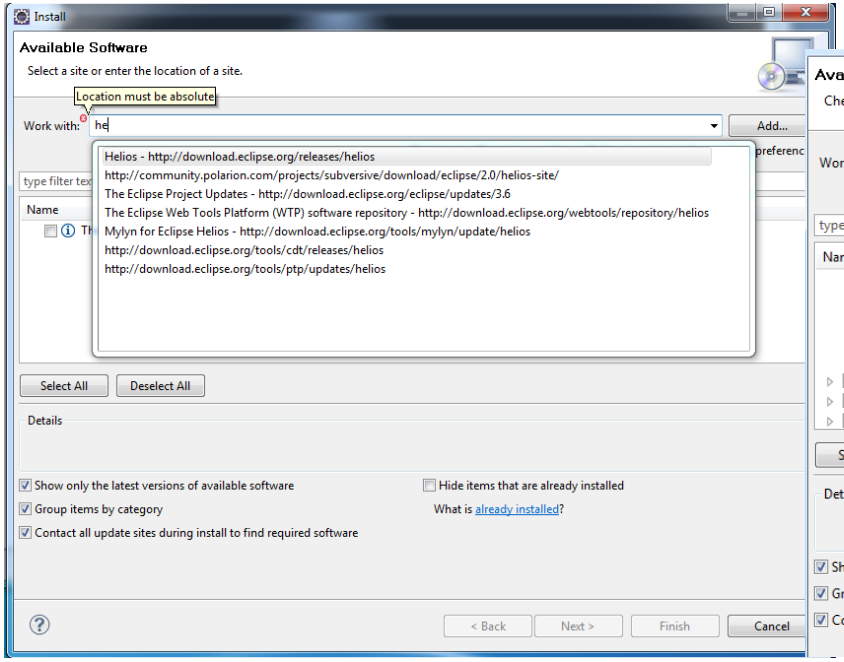
- In this case we will use it to synchronize our build directory with the Windchill codebase
  - **Help > Install New Software**
  - Enter the following URL http://andrei.gmxhome.de/eclipse/

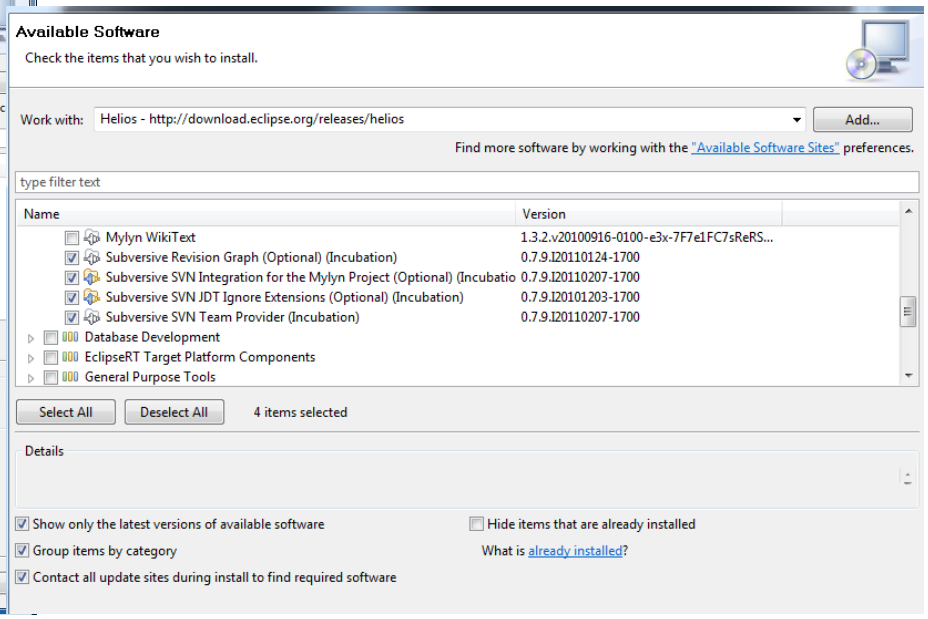


5. Install SVN Repository

Subversive

- **Help > Install New Software**
- For Subversive, it doesn't need to enter URL. You have to use eclipse update site
- On the work with line, you enter "Helios"
- Select Collaboration > Subversive software
- After finished install and restart, eclipse will ask to install connector for Subversive. In this time, if you are using windows operating system, you just select toolkit.





6. Eclipse configuration – (1/2)

Miscellaneous Configurations

Window > Preferences

- **Java > Compiler > Building**
  - Unselect ***Output folder > Scrub output folders when cleaning projects***
- **Java > Code Style > Formatter > Import...**
  - select the following file: %wt_viewroot%\Windchill\src\devtools\eclipse\eclipseFormatterProfile.xml
- **General > Editors > Text Editors**
  - Check "***Insert spaces for tabs***"
  - Optionally check ***show print margin*** and set to 120
  - Optionally check ***show line numbers***
- **General > Editors > File Association**
  - Add "***.jsfrag***" and add associated editors ***Javascript Editor***

XML 문서

eclipseFomatterProfile,xml

6. Eclipse configuration – (2/2)

Configure Aptana as Follows

If you have the Aptana plugin

- **Aptana > Editors > Javascript > Formatting > Import…**
  - select the following file: %wt_viewroot%\Windchill\src\devtools\eclipse\eclipseAptanaFormatterProfile.xml
- **General > Editors > File Association**
  - update “*.jsfrag” and “*.js” to use associated editor Aptana JS Editor by default
- **Aptana > Editors > General**
  - Under Tab Insertion select Use Spaces
- **XML > XML Files > Editor**
  - Select Indent using spaces and set Indentation size to 4

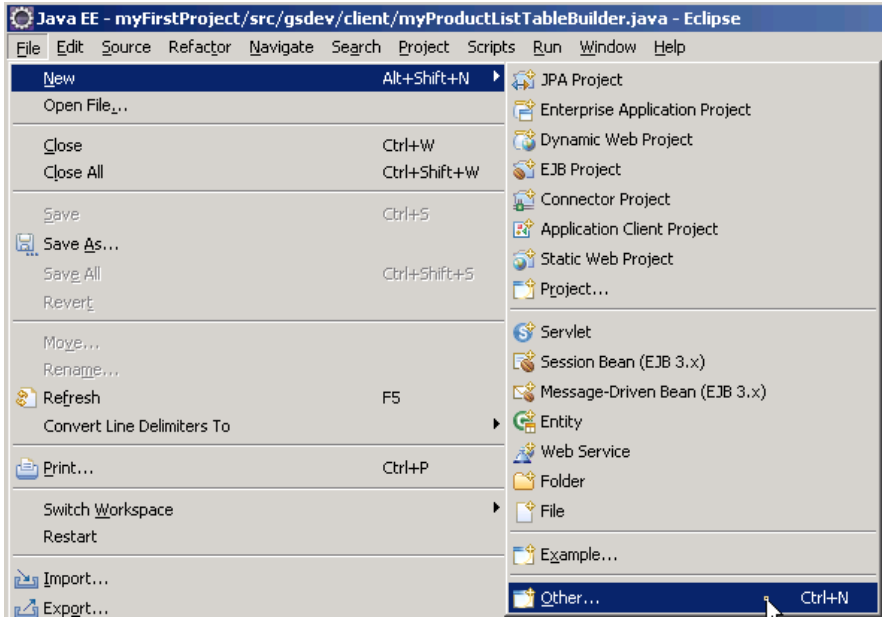
XML 문서

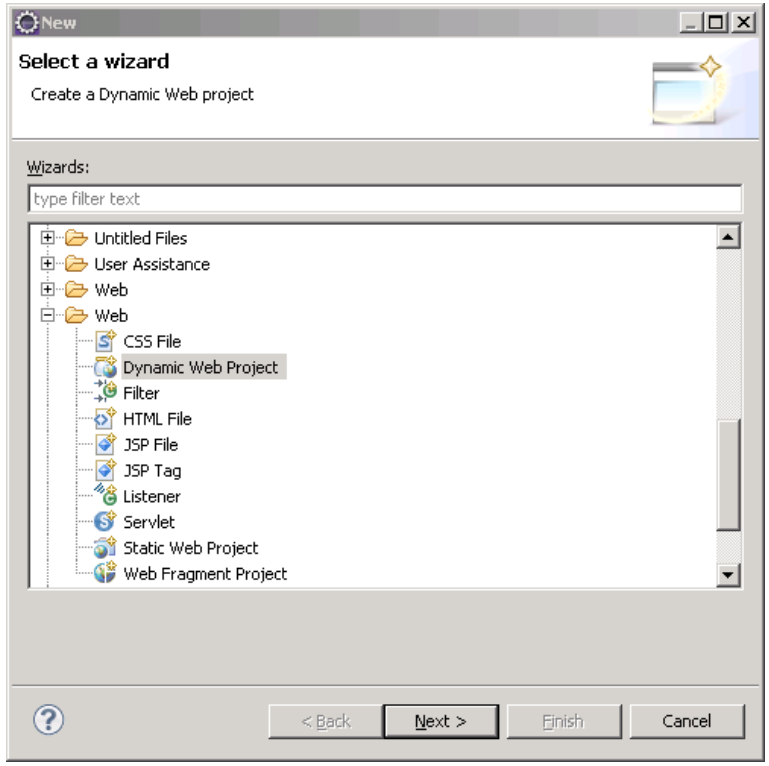
eclipseAptanaFomatterProfile,xml

Create a Dynamic Web Project

After Windchill 10, we are using Web project because it is more comfortable for tag program and JSP programming than Java project.

- File > New > Other displays a list of Project and File Types
  - ✓Varies depending on options installed
  - ✓Dynamic Web Project is useful for JSTL configurations.



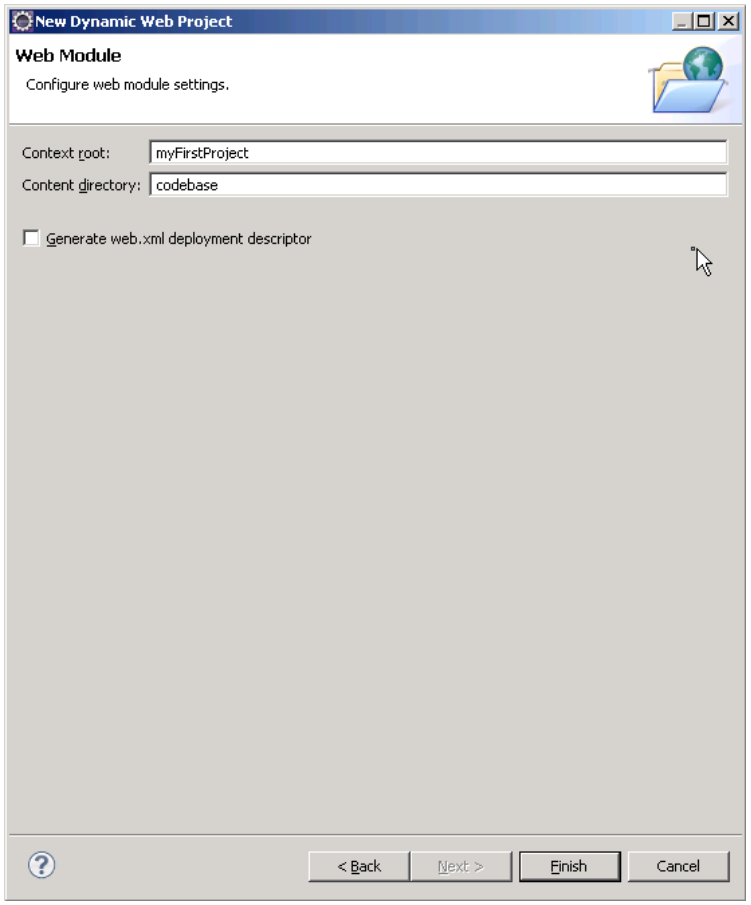


7. Create Project – (2/2)

Create a Dynamic Web Project

Every configuration will use default except on Web module.

The configuration must create codebase content.



8. Set environment of project for Windchill

Classpath makes API available

Classpath is Configured using the Java Build Path in Eclpse

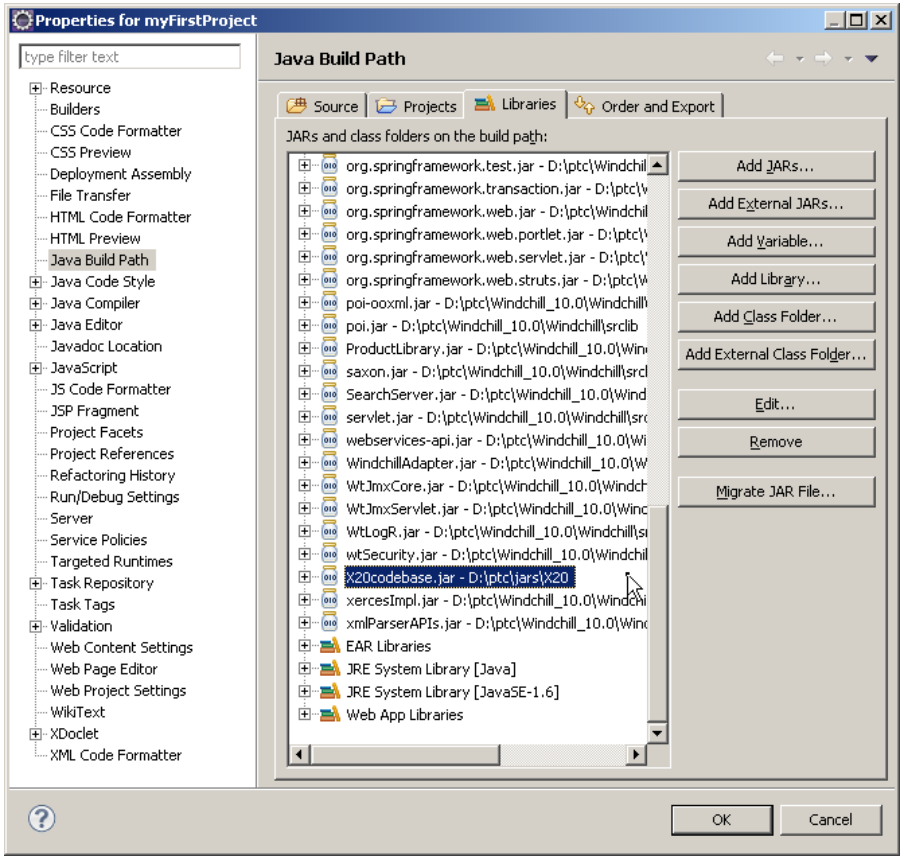
The Classpath consists of:

- Jars in the %WT_HOME%\srclib directory

  The srclib has many directories related Windchill, so you can add more Jars for especial implemetation.

  For example:
  - ✓ Info*Engine implemetation must include srclib/ie libraries.
  - ✓ Work Group Manager implementation needs to include srclib/prowt libraries.

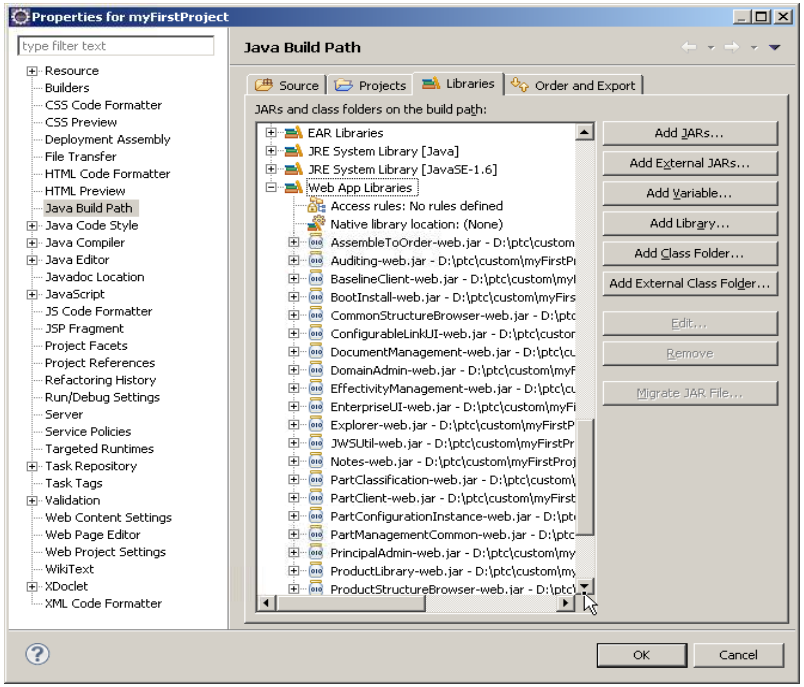
- Class files in the codebase.

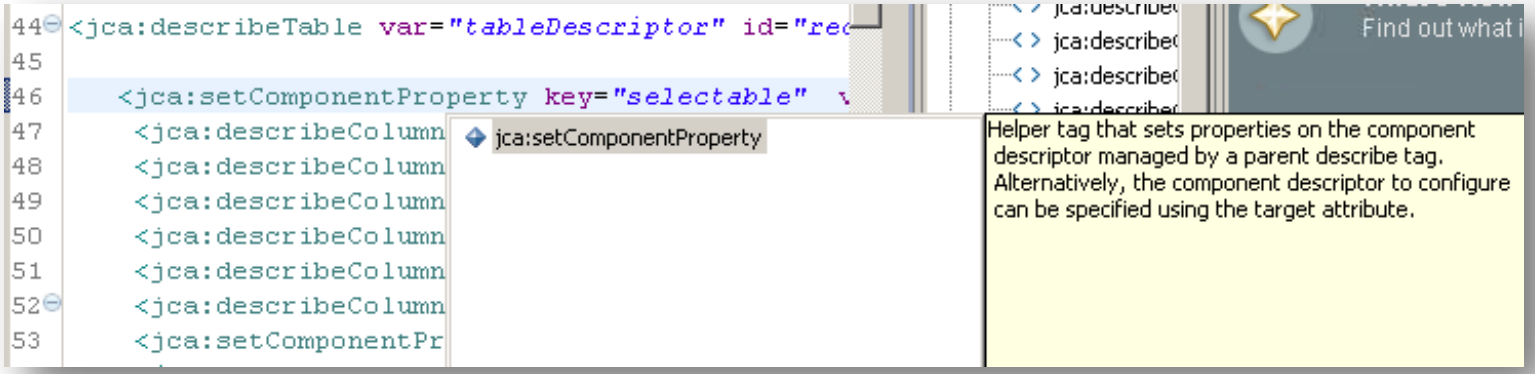


Code-assist in JSP

To enable code-assist in JSP, copy the web app libraries into the development root folders.

- *From %WT_HOME%\codebase\WEB-INF\lib*web.jar*
- *To %DEV_ROOT%\codebase\WEB-INF\lib*web.jar*





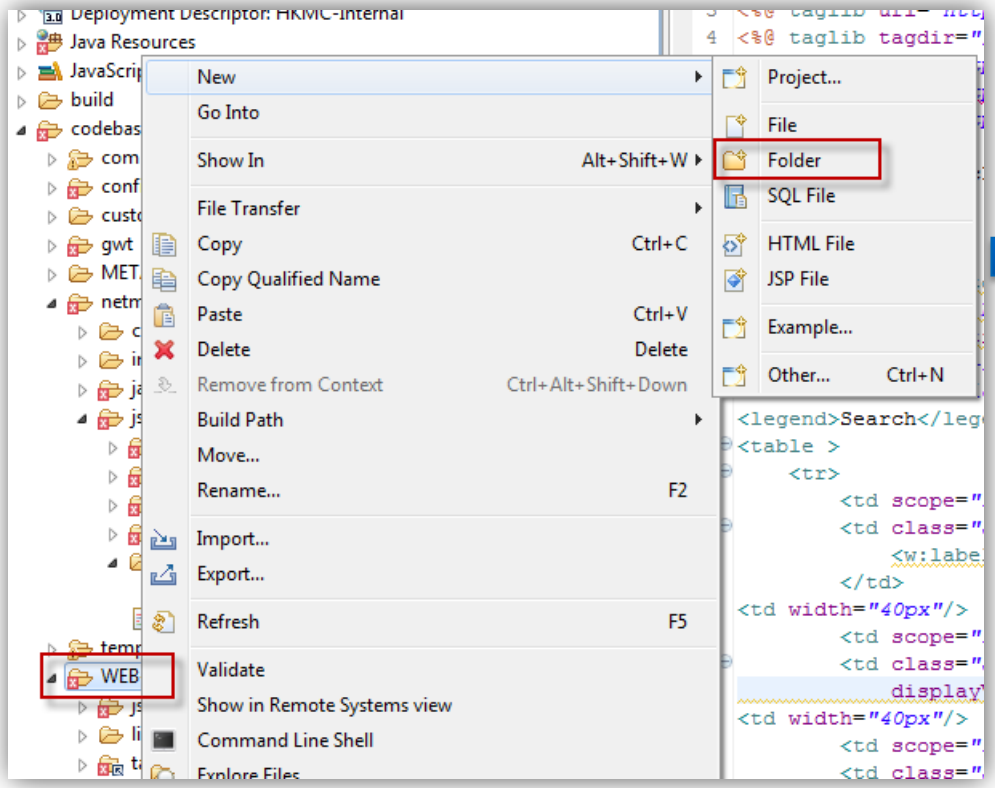
Code-Assist in JSP

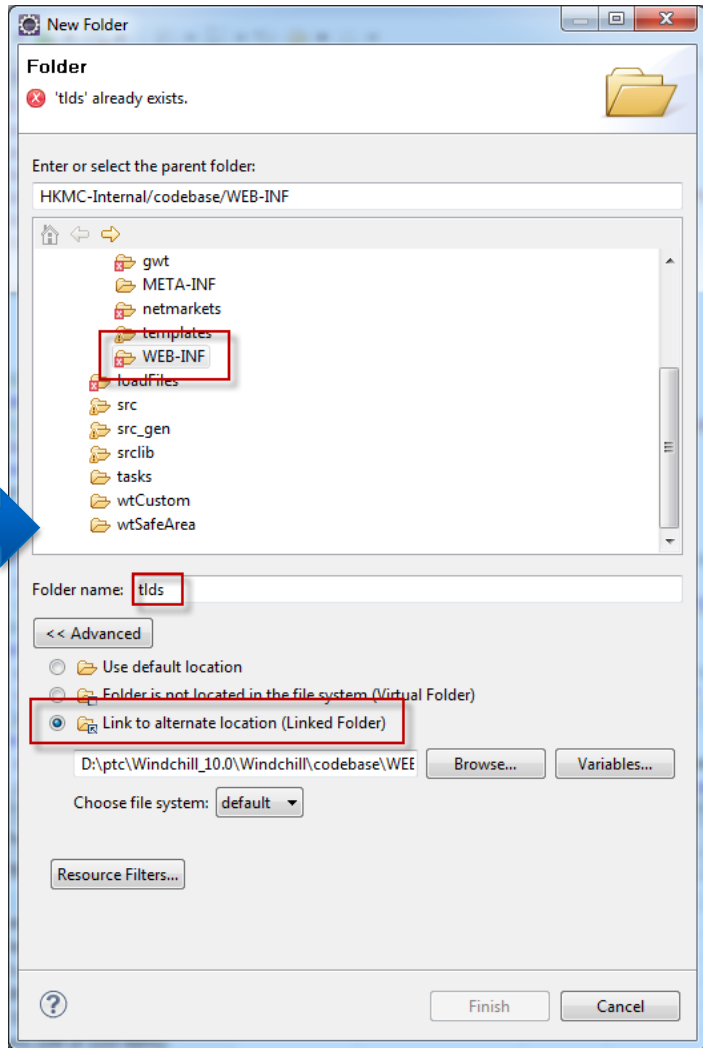
8. Set environment of project for Windchill

TLDs and TAGs under the WEB-INF folder are required by Eclipse

Configure a Second Web Content folder:

- Create a new folder under the Web Content (codebase) folder
- Link it to the WEB-INF folder
  - tlds
  - tags

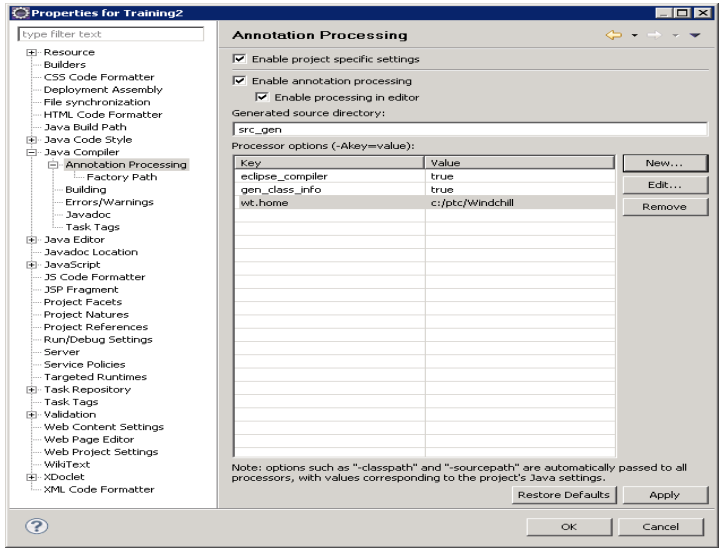


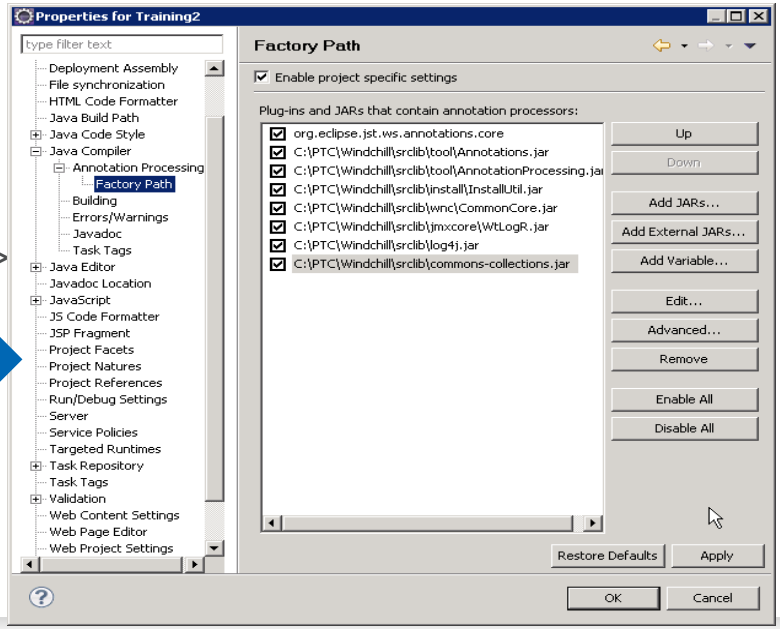


Annotation Processing

For annotation compiling, you must set following environment.

- –Set annotation configuration
  - •Goto Project properties > Java Compiler > Annotation Processing
  - •Set “Enable project specific setting”
  - •Set “Enable annotation processing”
  - •Set “Enable processing In editor”
  - •Change Generated source directory – “src_gen”
  - •Set 3 keys
    - –eclipse_compiler=true
    - –gen_class_info=true
    - –wt.home=$WT_HOME
- –Set Factory path
  - • Goto Project properties > Java Compiler > Annotation Processing > Factory Path
  - •Add JAR file like screenshot

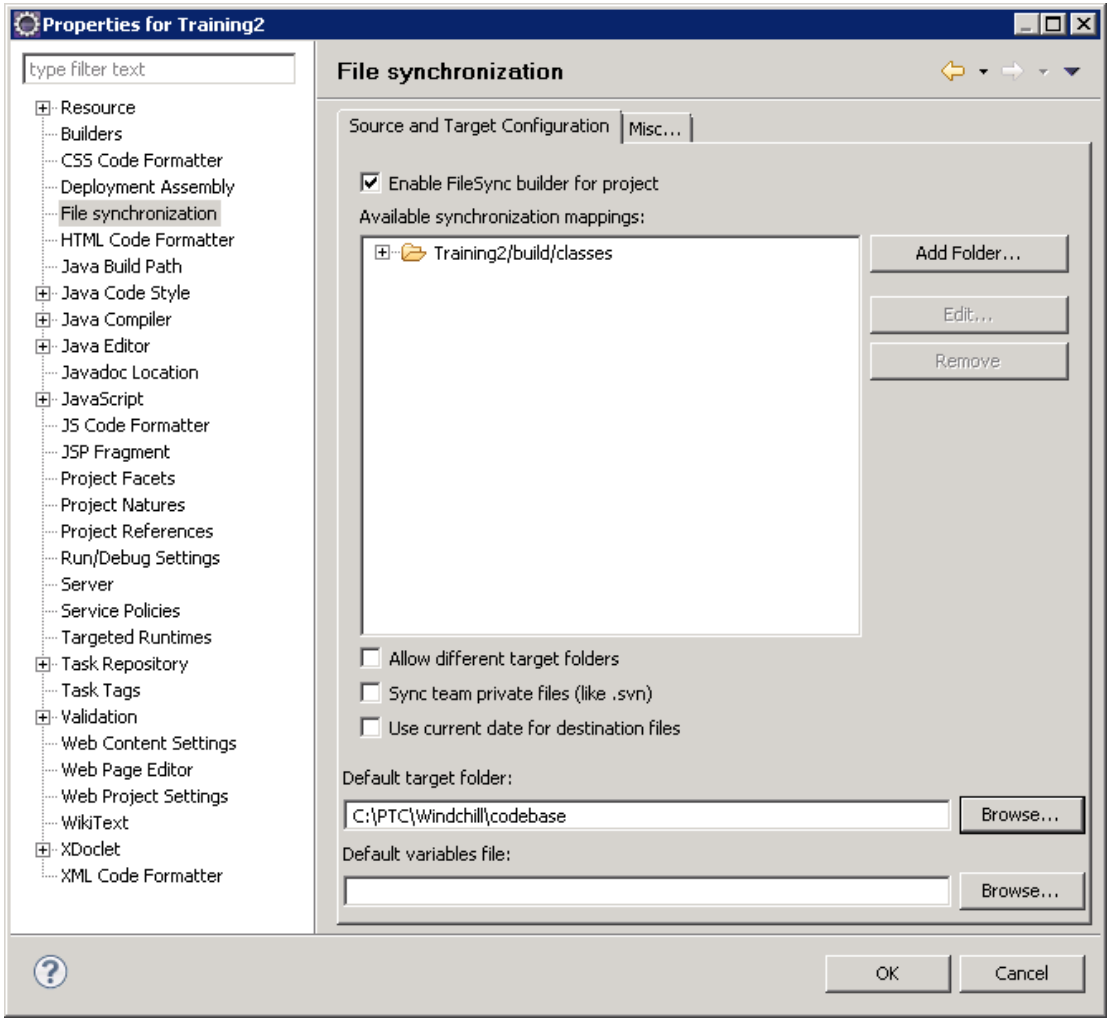




FileSync Preference

For annotation compile before using ant the based annotation class was deployed on Windchill class path.

- This configuration will be effect when you created new modeling within annotation, system will synchronize automatically to Windchill.
- It means Eclipse will also update registry files. (Must be mind)
- If you do not want automatic updating, please do not set this option.

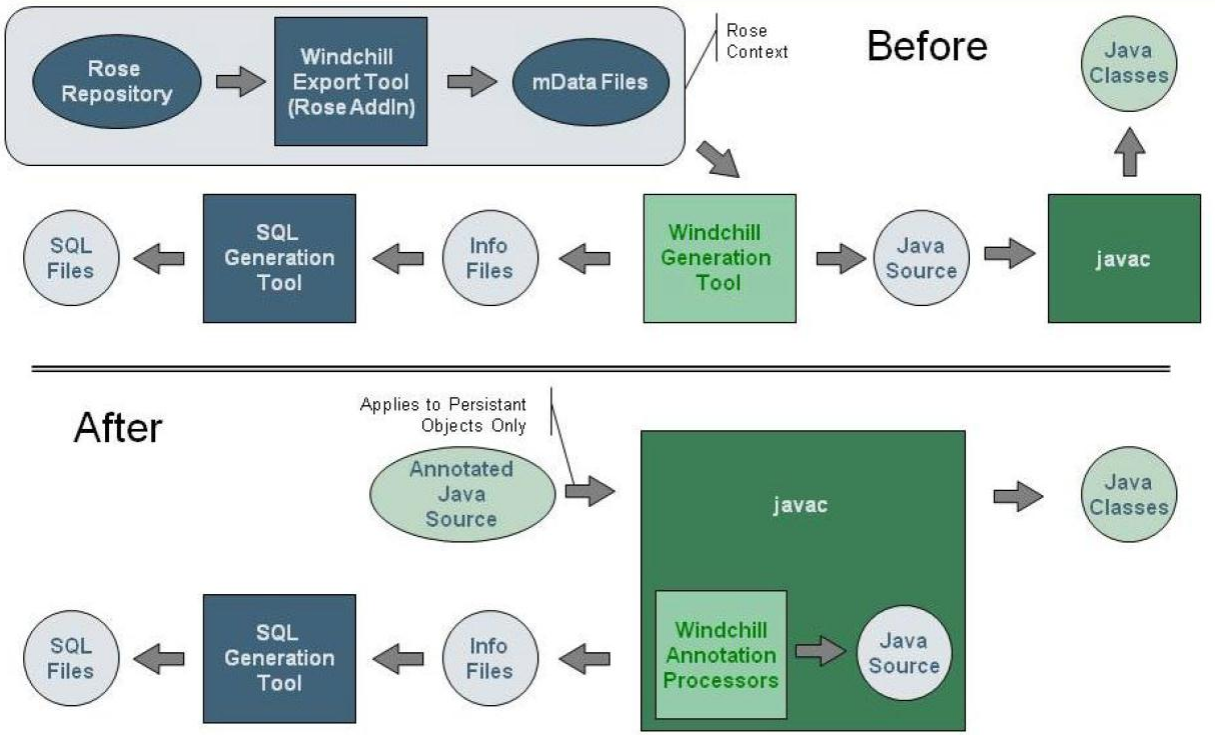


Modeling with annotation

WC10's generation

WC10 start to use annotation, so the modeling and generating process was changed. Following is the point of changed view.

- Ration Rose was gone. (No have visual tool)
- All modeled object must be generated from annotation file. (previous classes doesn't have new annotation file except some important objects)



Generation command

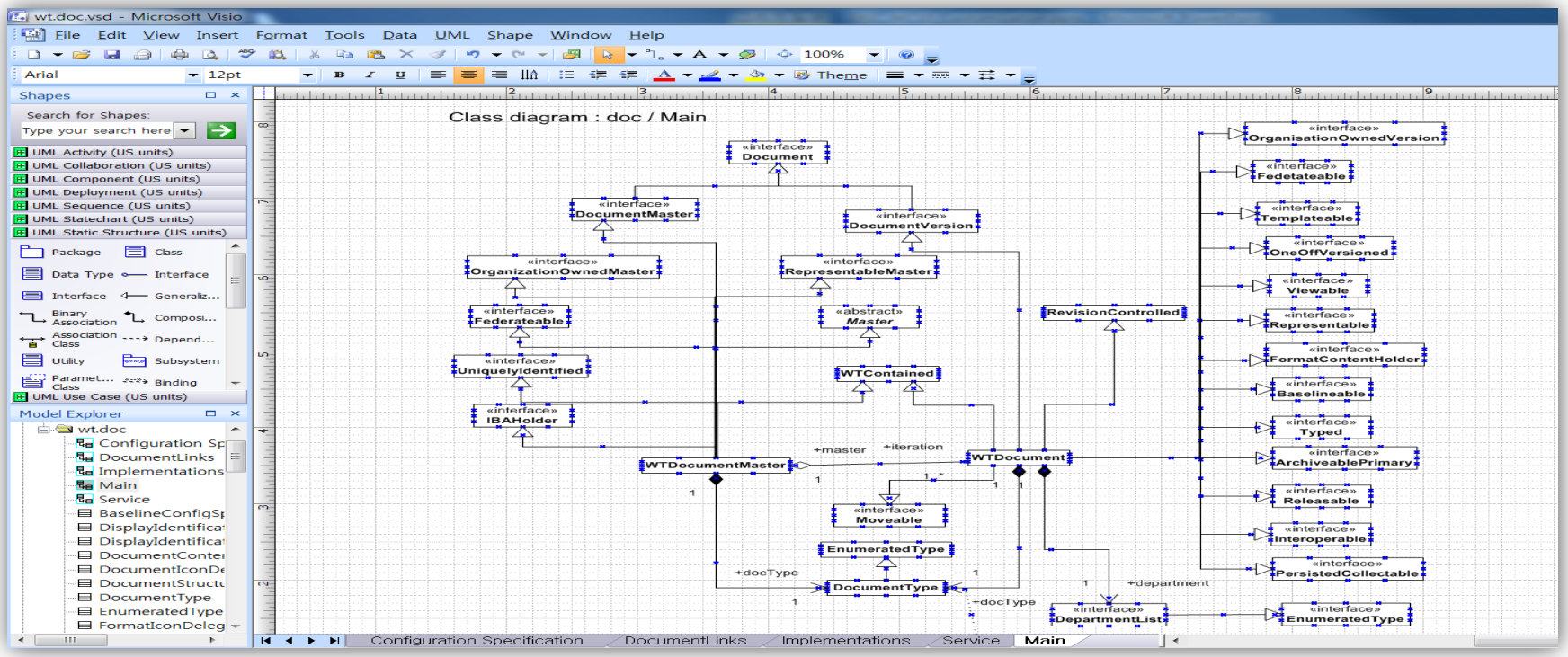
When you want to generate annotation file, you have to use Windchill command. The command is supported by tools.xml

- Help!
  - ant -f bin/tools.xml help – or – ant -f bin/tools.xml <target>.help
- Compile
  - ant -f bin/tools.xml class -Dclass.includes=com/ptc/training/**
- SQL scripts
  - ant -f bin/tools.xml sql_script -Dgen.input=com.ptc.training.*
- Bundles
  - ant -f bin/tools.xml bundle -Dbundle.input=com.ptc.training.*

All of annotation file must be existed below on “$WT_HOME/src” directory.

Visual tool (Just imagination)

Because Rational Rose was gone, PTC are using ***Microsoft Visio*** for visual tool before annotation design.



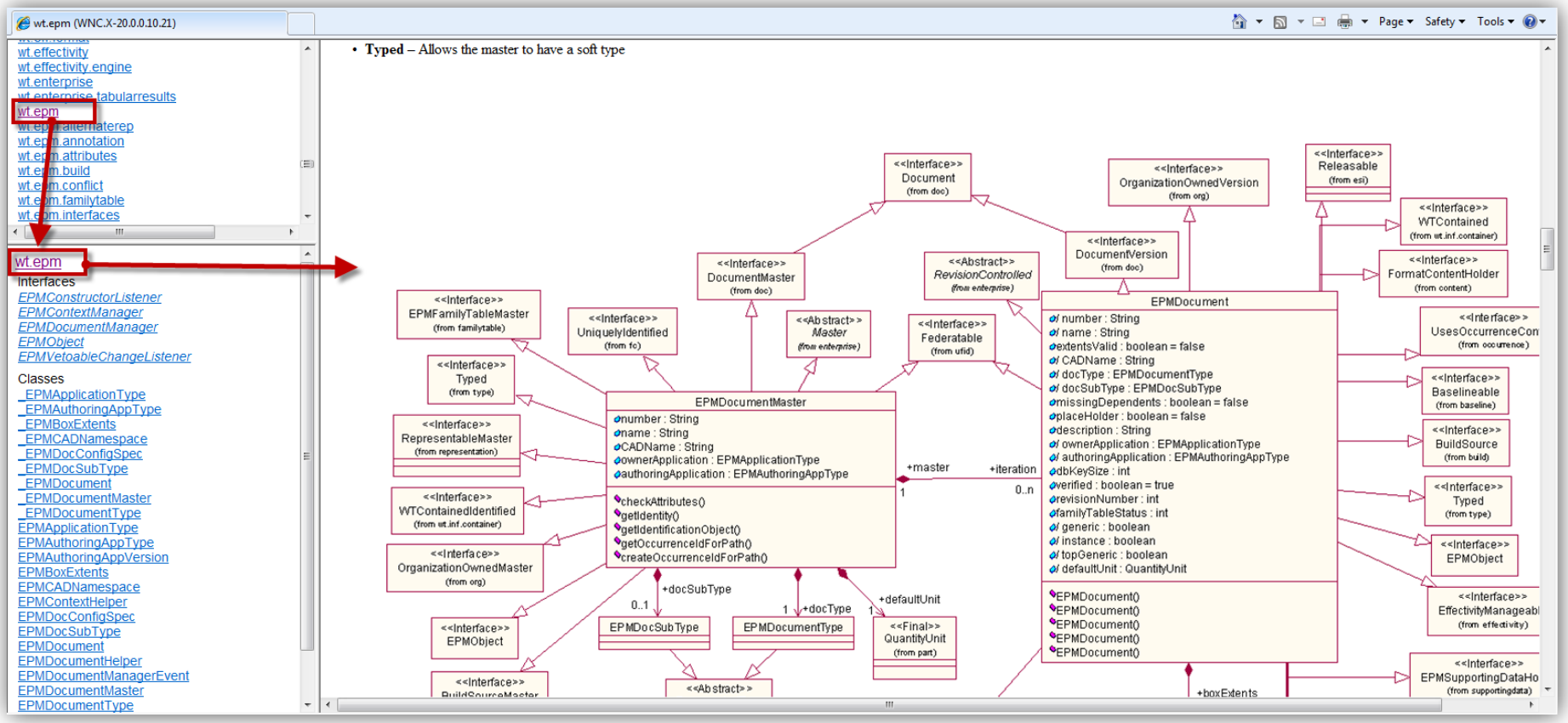
Windchill basic model can be get from OpenGrok.

–http://ah-grok.ptcnet.ptc.com/search?q=&project=x-20-M020&defs=&refs=&path=vsd&hist=

Using Windchill Java Doc ($WT_HOME/codebase/wt/clients/library/api)

On the Java Doc, some visual picture of modeling is existed for each of package. (It is not supported for all of packages.)

– For example, if you want to know EPMDocument modeling, look "**wt.epm**" package.



Get annotation sample from Windchill Javadoc

You can easly get annotation sample from windchill javadoc.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

wt.vc.views

Class View

java.lang.Object
  └ wt.fc._WTObject
    └ wt.fc.WTObject
      └ wt.vc.views._View
        └ **wt.vc.views.View**

All Implemented Interfaces:

Externalizable, Serializable, wt.fc._NetFactor, wt.fc._ObjectMappable, wt.fc._Persistable, NetFactor, ObjectMappable, Persistable, DisplayIdentification

```
@GenAsPersistable(superClass=WTObject.class,
                  versions=-4333025332245130 71L,
                  properties={
                  @GeneratedProperty(name="name",type=java.lang.String.class,supportedAPI=PUBLIC,
                     javaDoc="The name of the View.  Must be unique.",constraints=@PropertyConstraints(upperLimit=30,
                     required=true),columnProperties=@ColumnProperties(unique=true)),
                  @GeneratedProperty(name="sortId",type=int.class,accessors=@PropertyAccessors(setAccess=PACKAGE))},
                  tableProperties=@TableProperties(tableName="WTView"))
public final class View
extends _View
```

General sample

When we want to make new object, designer most generally use following general designing.

> wt.part.WTPart
  - derivedProperties & foreignKeys

> wt.vc.baseline.BaselineMember
  - Example association

> wt.part.PartType
  - Example enumerated type

> wt.part.partResource
  - Example resource bundle

> wt.fc.PersistenceManager & wt.fc.StandardPersistenceManager
  - Example service

> com.ptc.windchill.annotations.metadata
  - Explanation of annotations commands

Meet annotation

Primary annotations (you'll start with these)

- **GenAsUnPersistable** non-persistent interfaces implemented by persistent classes and Evolvable classes (usage should be rare)
- **GenAsObjectMappable** persistent objects stored as columns (aka "cookies")
- **GenAsPersistable** persistent objects stored as tables (aka "persistables")
- **GenAsBinaryLink** specialize persistables associating a role A/B persistables

- **GenAsEnumeratedType** extensions of wt.fc.EnumeratedType
- **GenAsPrimitiveType** classes that can be reduced to a simple java "primitive" type

- **GenAsDatastoreSequence** database sequences
- **GenAsDatastoreStruct** database structures
- **GenAsDatastoreArray** arrays of database column types, including structs
- **GenAsDatastoreTable** non-persistable persistables

Meet annotation

Secondary annotations (properties of primary annotations)

- **GeneratedProperty** field consisting of Java “primitives” and cookies
- **GeneratedForeignKey** reference to persistable stored locally in this persistable
- **GeneratedRole** role A/B of a binary link
- **DerivedProperty** convenience accessors to above

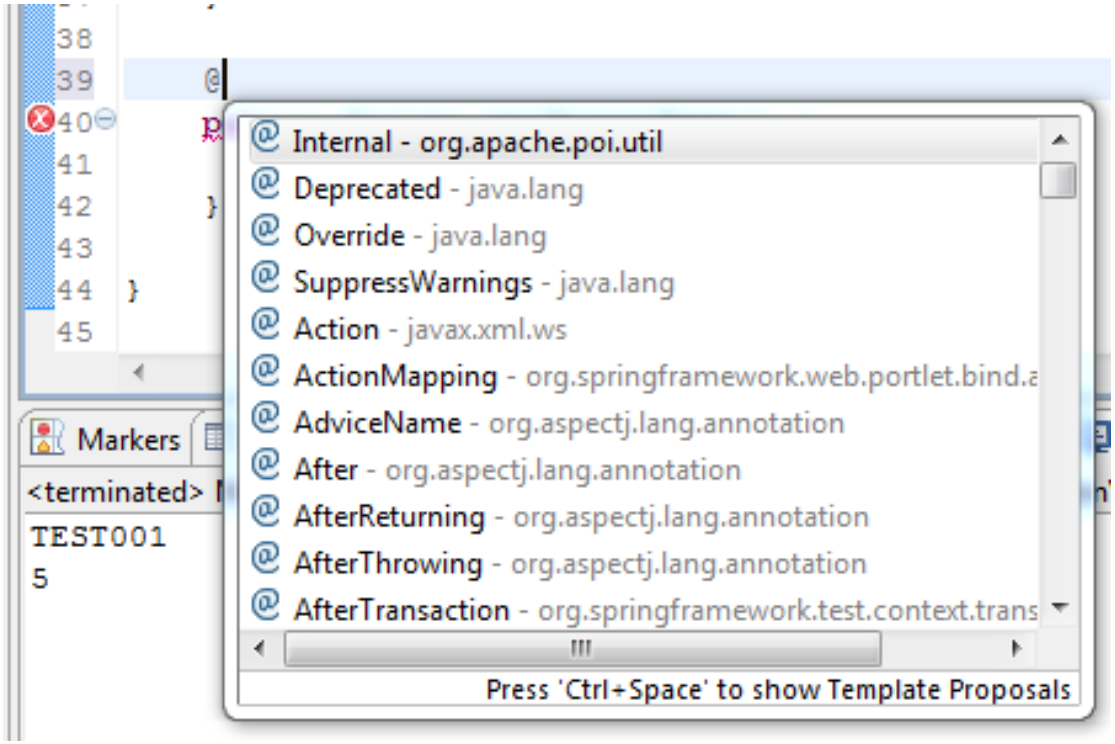
- **TableProperties** table name, composite indexes, and composite unique indexes
- **IconProperties** standard/open icon (this may be obsolete)

Tertiary annotations (properties of secondary annotations)

- **PropertyAccessors** getter/setter access/exceptions
- **PropertyConstraints** string case, upper/lower limits, required, changeability
- **ColumnProperties** column name/type, indexing, uniqueness, persistence
- **ForeignKeyRole** target persistable’s role (master in master/iteration)
- **MyRole** my role as the target sees me (iteration in master/iteration)

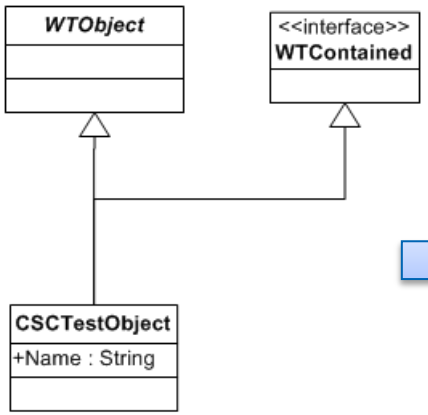
Look annotation list on the Eclipse

On the eclipse, you can also see annotation list. Using ***CTRL+Space***.



Sample of new Object

- WC10 doesn't use "preserve" marks.
- Can not change generated source code.



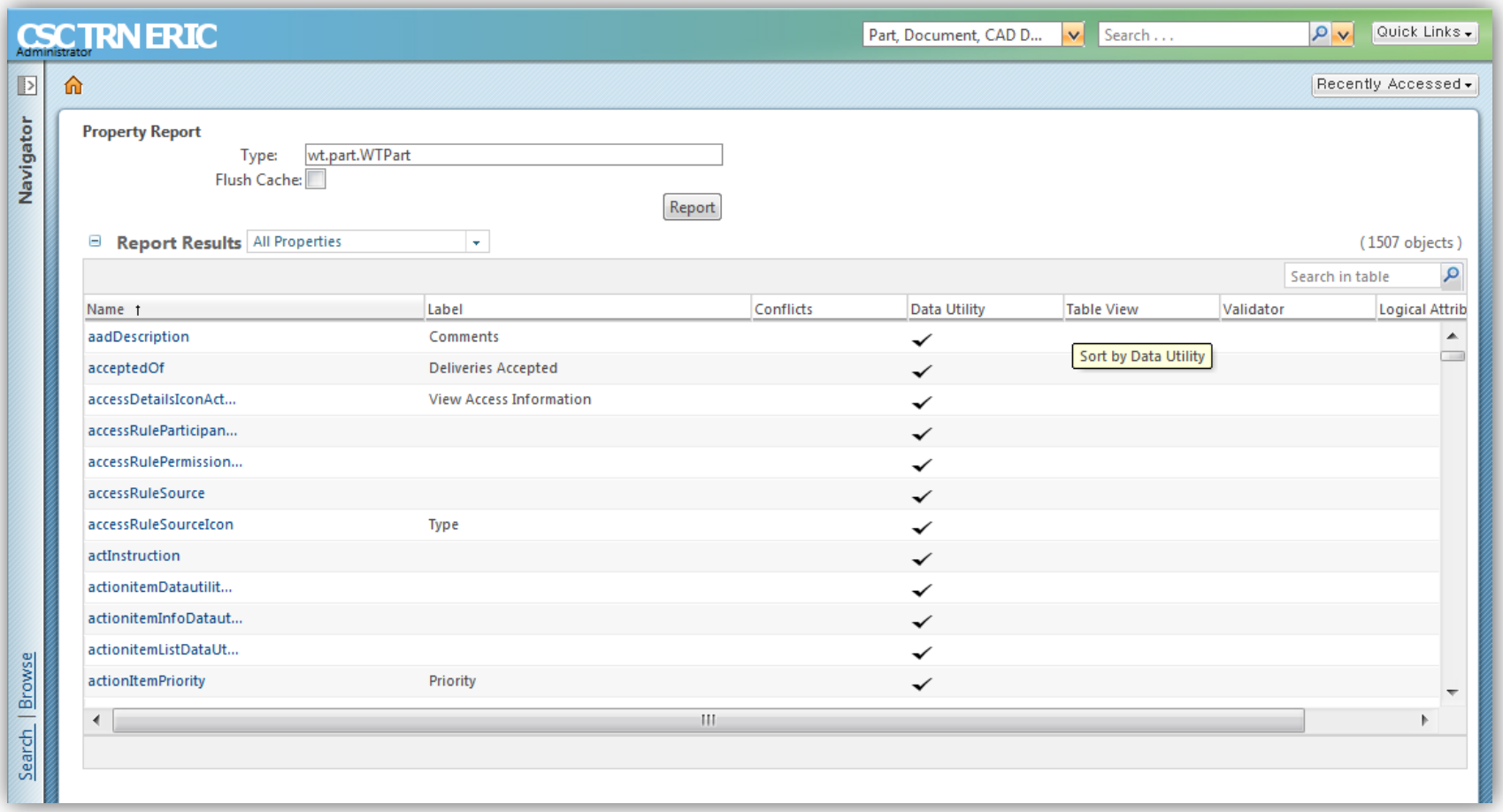

```
package ext.csc.training.modeling;

import org.apache.poi.util.Internal;
..
import com.ptc.windchill.annotations.metadata.GenAsPersistable;
import com.ptc.windchill.annotations.metadata.GeneratedProperty;
import com.ptc.windchill.annotations.metadata.PropertyConstraints;

@GenAsPersistable(superClass=WTObject.class, interfaces={WTContained.class},
    properties={
        @GeneratedProperty(name="name", type=String.class,
      constraints=@PropertyConstraints(required=true))
    }
)
public class CSCTestObject extends _CSCTestObject {
    static final long serialVersionUID = 1;

    public static CSCTestObject newCSCTestObject() throws WTException {
        final CSCTestObject instance = new CSCTestObject();
        instance.initialize();
        return instance;
    }

    @Override
    public void checkAttributes() throws InvalidAttributeException {
        try {
            nameValidate(name);
        } catch (WTPropertyVetoException wtpve) {
            throw new InvalidAttributeException(wtpve);
        }
    }
}
```

Action & Property Report

Property Report

Property report provide the detail of validate attribute lists for customization

http://localhost/Windchill/app/#ptc1/netmarkets/jsp/property/propertyReport.jsp

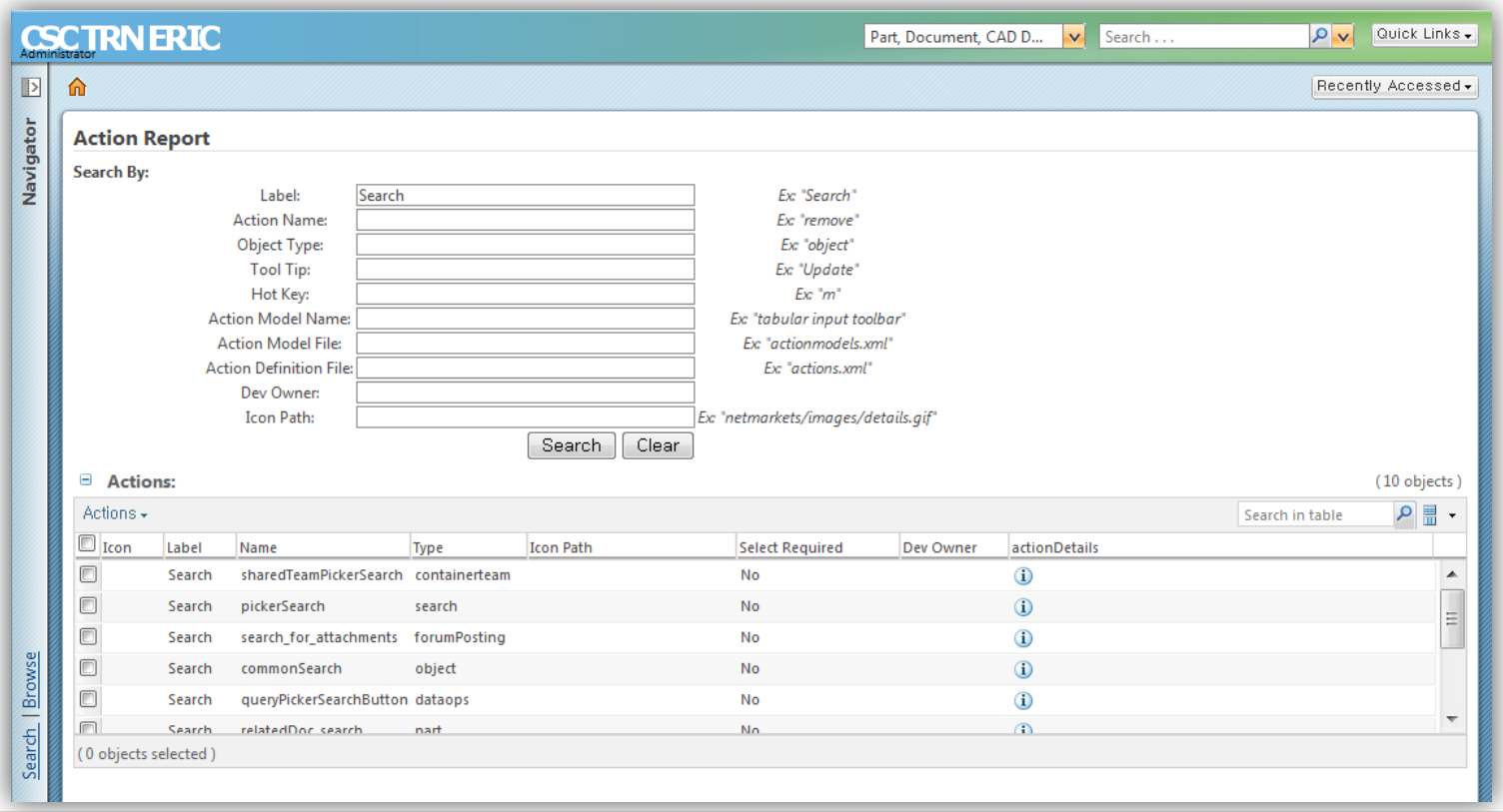


Action Report

The Action Report is a search page that returns the details of actions matching the search criteria.

The Report can be accessed at the page

http://localhost/Windchill/app/#ptc1/netmarkets/jsp/carambola/tools/actionReport/action.jsp



Action Registry

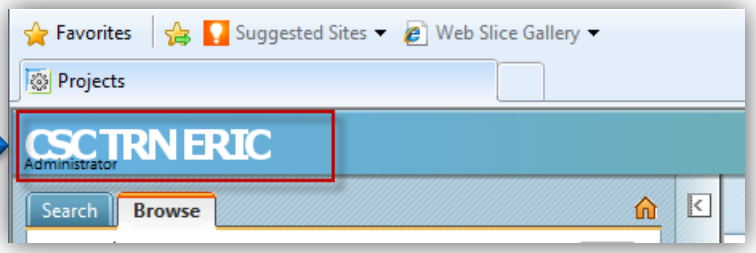
The action registry is a Windchill service — including a cache — that:

- Reads the action and action model configuration files.
- Builds individual actions.
- Builds action models.
- It is initialized when the Method Server starts.
- It can be reloaded without restarting the Method Server or Tomcat:
  - From a Windchill shell, execute: **windchill com.ptc.netmarkets.util.misc.NmActionServiceHelper**.
  - This command only reloads actions and action models; not JSP pages, or Method Server properties.

Log & Navigation

1. Custom Log

Custom Log

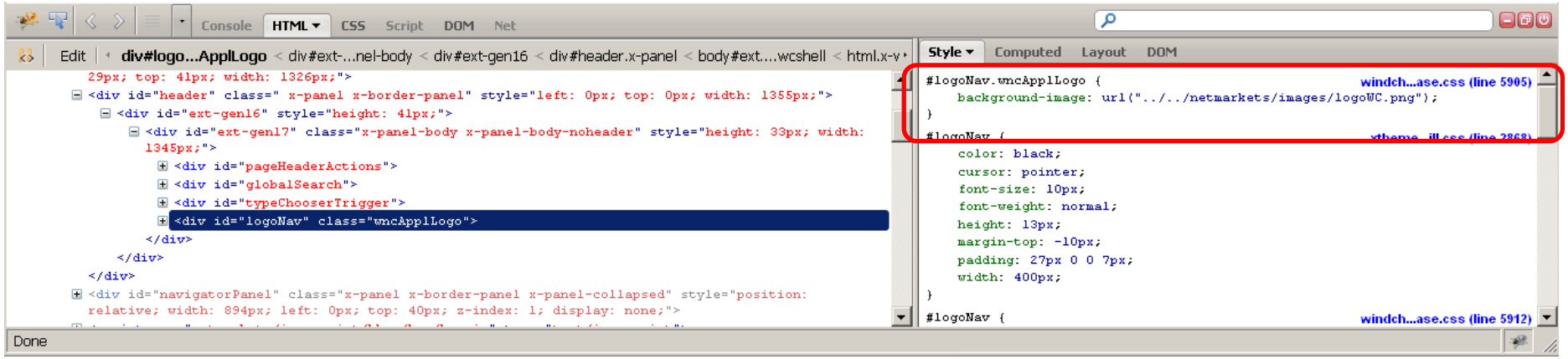


1. Custom Logo

Review and Create Logo image

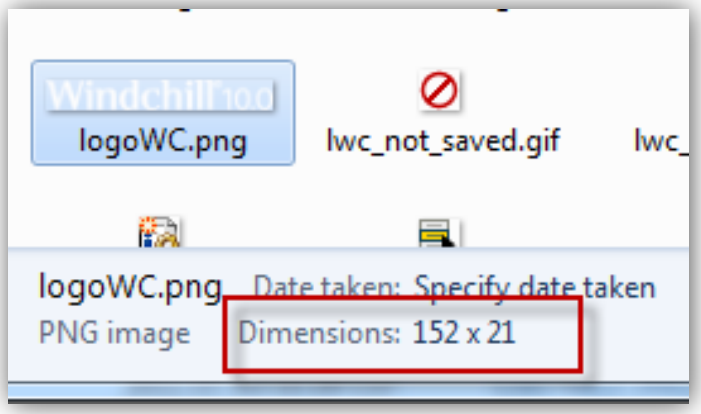
1. We have to check which style sheet is being used and where the icon file is located

2. This logo constitutes white text on a transparent background. As such, it will be difficult to see the logo on the white background of a File Manager window.



3. Check the size of original image on Windchill server.

4. Create custom image (For example, file name is customLogo.png)



1. Custom Logo

Update style sheet

1. Create a new stylesheet called *custom.css and place it in %WT_HOME%\codebase\custom*
2. Create custom.css on new folder
3. Copy following block text to custom.css

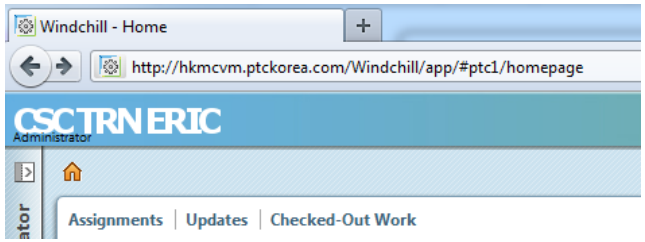
```
#logoNav.wncApplLogo {
   background-image: url("../../custom/customLogo.png ");
}
```

Image path must be based on custom.css location

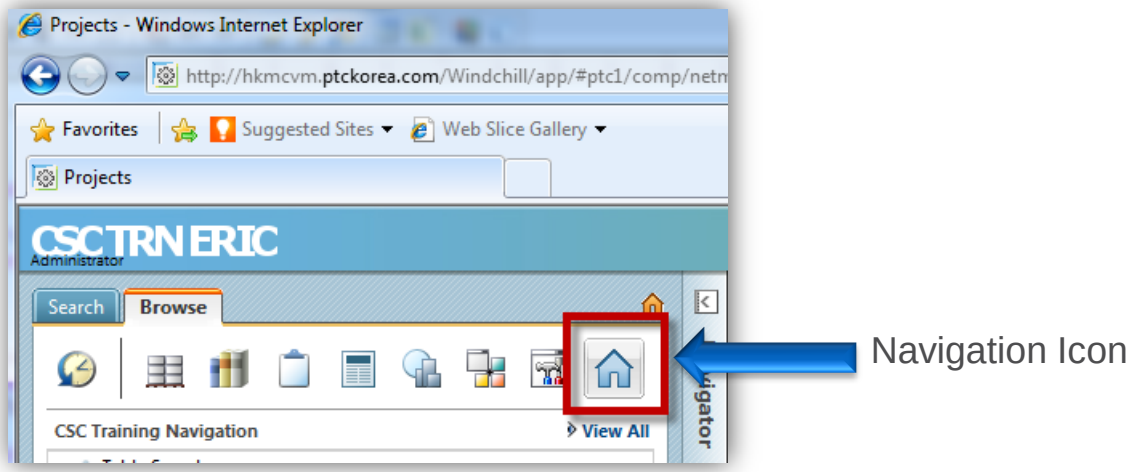
4. Register the *custom.css file by updating codebase\presentation.properties.xconf*

```
1  <?xml version="1.0" encoding="utf-8"?>
2
3  <!DOCTYPE Configuration
4    SYSTEM "xconf.dtd">
5  <Configuration targetFile="codebase/presentation.properties">
6     <Property default="PTC" name="netmarkets.presentation.author"/>
7     <Property default="custom/custom.css" name="netmarkets.presentation.cssFiles"/>
8     <Property default="http://www.ptc.com" name="netmarkets.presentation.website"/>
9  </Configuration>
```

5. Execute xconfmanager & restart all service.



2. Navigation Menu(Icon)



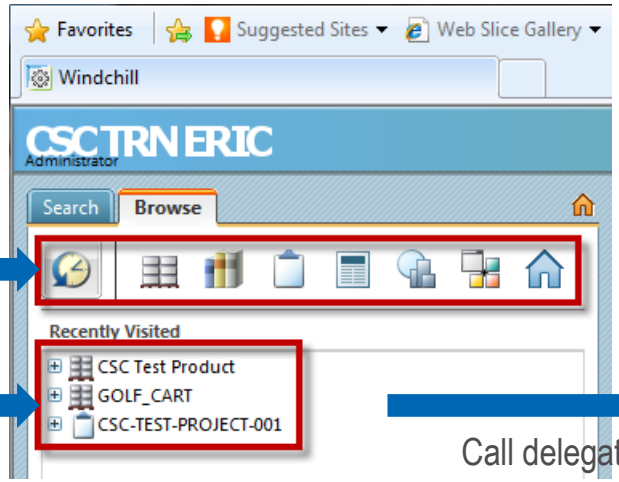
2. Navigation menu

Action.xml and ActionModel.xml

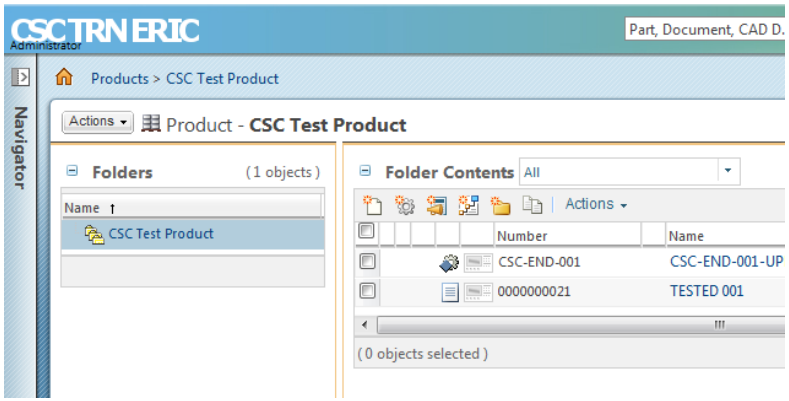
Configuration of Windchill 10 navigation is similar between 9.1.

- All basic Windchill configuration files of action & action model is located in $WT_HOME/codebase/config/actions
- For navigation icon and sub-menu are using xxxAction.xml and xxxActionModel.xml
- ActionModel.xml is including a design of menu lists.
- Action.xml is including for activity of action model menu.

Designed by actionModel



Call delegate when click menu
Register by action



Action Result

2. Navigation menu

Register new XML files

We can change directly original XML, but it is not best way because it can be changed when Windchill updating or patch. So we try to use expanded XML files.

- OOTB has custom-actions.xml and custom-actionModels.xml
- Add new training XML in custom environment

Add following sentence in %WT_HOME%\codebase\config\actions\custom-actions.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE listofactions SYSTEM "actions.dtd">
<listofactions>
        <include href="config/actions/csc/csc-actions.xml"/>
</listofactions>
```

Add following sentence in %WT_HOME%\codebase\config\actions\custom-actionModels.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE actionmodels SYSTEM "actionmodels.dtd">
<actionmodels>
        <include href="config/actions/csc/csc-actionmodels.xml"/>
</actionmodels>
```

In this case, you must create one directory which name is "*csc*" in "*$WT_HOME/codebase/config/actions*"

Create new action

Create registered action file on correct directory. Action and ActionModel files are different format, so you have to watch out for making new file.

Create and Modify %WT_HOME%\codebase\config\actions\csc\csc-actions.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE listofactions SYSTEM "actions.dtd">
3
4  <listofactions>
5       <objecttype name="navigation" class="" resourceBundle="com.ptc.core.ui.navigationRB">
6          <action name = "csc" uicomponent="TAB_ADMIN"/>
7       </objecttype>
8  </listofactions>
```

- For action file, you must add DTD configuration line like line-2.
- Action type which name is “navigation” is already assigned by Windchill, so we just reuse the object type action.
- If you want to add top menu(button), it is enough.

2. Navigation menu

Create new action Model

Create registered action file on correct directory. Action and ActionModel files are different format, so you have to watch out for making new file.

Create and Modify %WT_HOME%\codebase\config\actions\csc\csc-actionmodels.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <actionmodels>
4      <model name="main navigation">
5          <description>Main navigation (tabs)</description>
6          <action name="home"       type="navigation"/>
7          <action name="program"    type="navigation"/>
8          <action name="product"    type="navigation"/>
9          <action name="project"    type="navigation"/>
10         <action name="change"     type="navigation"/>
11         <action name="library"    type="navigation"/>
12         <action name="org"        type="navigation"/>
13         <action name="site"       type="navigation"/>
14         <action name="supplier"   type="navigation"/>
15         <!-- entry for customization tab -->
16         <action name="customization"   type="navigation"/>
17         <action name="csc"    type="navigation"/>
18     </model>
19     <model name="csc navigation">
20         <description>
21             For CSC navigation
22         </description>
23     </model>
24 </actionmodels>
```

It is overriding model. It can be copied from OOTB action models, and we just add our action

Do not copy sample description because each of installation is different , so the elements will be different. Copy your system model block.

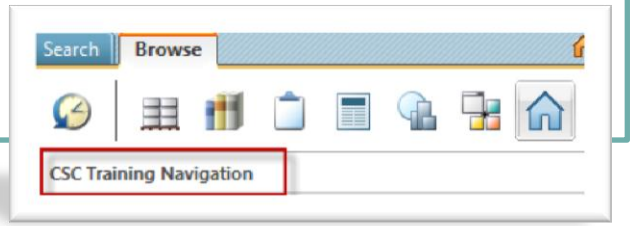
This must be added although sub-menu isn't existed

2. Navigation menu

Add resource bundle

The resource bundles of menu which is displaying on UI are managed the other way comparing original of OOTB resource bundle architecture.

- The resource file is managed on "$WT_HOME/codebase/action_[Locale code].properties
- All of locale character must be changed UTF-8 format except on English
- Also you must update default property file. (action.properties)



Update %WT_HOME%₩codebase₩action_LOCALE.propertie and action.properties

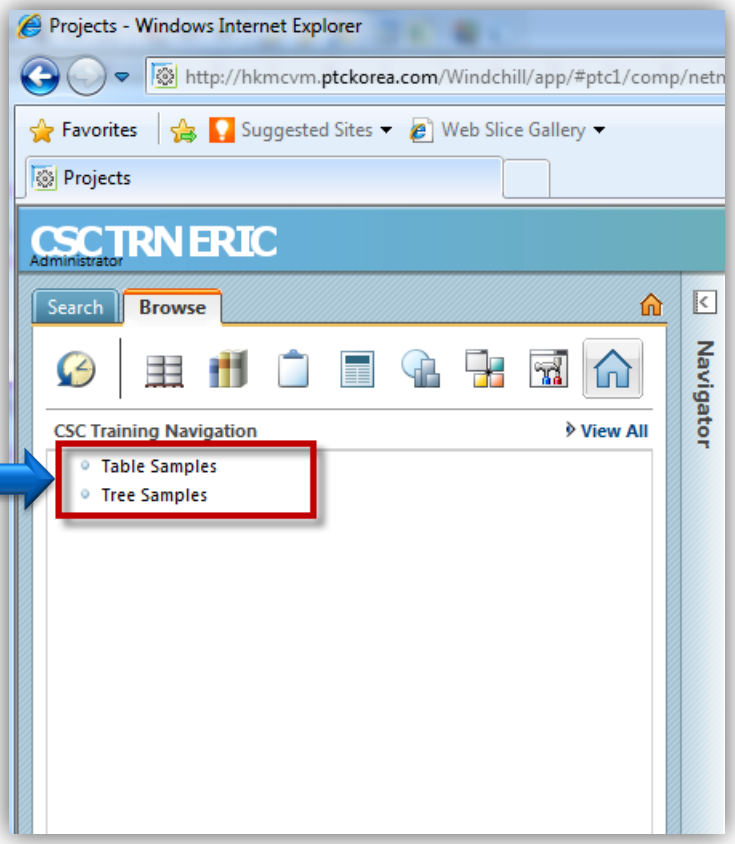
object.search_resulttable_authorizationagreement_file.description=₩ud30c₩uc77c
object.search_resulttable_authorizationagreement_edit.description=₩ud3b8₩uc9d1

navigation.csc.description=CSC

- Finally, you must restart all of Windchill service.
- If you want to adapt your configuration without restart service, use windchill command.: ***windchill com.ptc.netmarkets.util.misc.NmActionServiceHelper***

3. Sub-navigation menu

Sub-Navigation Menu



3. Sub-navigation menu

Modify ActionModels.xml

Sub menu is similar to adding navigation

Designing sub-menu using actions on %WT_HOME%\codebase\config\actions\csc\csc-actionModels.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<actionmodels>
    <model name="main navigation">
        <description>Main navigation (tabs)</description>
        <action name="home"        type="navigation"/>
        <action name="program"     type="navigation"/>
        <action name="product"     type="navigation"/>
        <action name="project"     type="navigation"/>
        <action name="change"      type="navigation"/>
        <action name="library"     type="navigation"/>
        <action name="org"         type="navigation"/>
        <action name="site"        type="navigation"/>
        <!-- <action name="supplier"  type="navigation"/> -->
        <!-- entry for customization tab -->
        <action name="customization"    type="navigation"/>
        <action name="csc"    type="navigation"/>
    </model>
     <model name="csc navigation" defaultActionName="tableSamples" defaultActionType="csc">
          <action name="tableSamples"  type="csc" />
          <action name="treeSamples" type="csc" />
     </model>
</actionmodels>
```

- Sub model name is created by a rule which is combining a name and type values of main navigation's action.
- Action will be generated to find action from action.xml. Type is objecttype name, and name is an element name of objecttype

3. Sub-navigation menu

Modify Actions.xml

Sub menu is similar to adding navigation

Adding designed action for sub-menus on %WT_HOME%\codebase\config\actions\csc\csc-actions.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE listofactions SYSTEM "actions.dtd">
<listofactions>
    <objecttype name="navigation" class="" resour
        <action name = "csc">
        </action>
    </objecttype>
    <objecttype name="csc" class="" resourceBundle="com.ptc.core.ui.navigationRB">
        <action name="tableSamples">
            <component windowType="page" url="comp/netmarkets.project.list.table"/>
        </action>
        <action name="treeSamples">
            <command windowType="page" url="wtcore/jsp/csc/jca/tree/tree.jsp"/>
        </action>
    </objecttype>
</listofactions>`
```

Windchill Basic Rule to find action Path
Windchill Basic Rule to find path: if we do not use "url" on <command/> tag
Basically finding the path which is "$WT_HOME/codebase/netmarkets/jsp"
and then use <objecttype/>'s "name" and Using action name.
finally "$WT_HOME/codebas/netmarkets/jsp/**csc/tableSamples.jsp**"

WindowType attribute on <command/>
"**page**" : Display directly on current page. Meaning is refresh.
"**popup**" : Display new popup page. Meaning is open new popup page.

- WC10 was added a MVC architecture, so , in the action, you can call MVC builder using <component> tag instead of <command> tag. (MVC will be explained next phase)

```
<action name="tableSamples">
    <component windowType="page" name="netmarkets.project.list.table"/>
</action>
```

3. Sub-navigation menu

Modify action.properties for display

Set display text to action.properties. If you don't set this properties, the menu was not showed text description like navigation icon.

- The resource file is managed on "$WT_HOME/codebase/action_[Locale code].properties
- All of locale character must be changed UTF-8 format except on English
- Also you must update default property file. (action.properties)

Modify to %WT_HOME%\codebase\action_LOCALE.propertie

```
.....
.....
object.search_resulttable_authorizationagreement_file.description=₩ud30c₩uc77c
object.search_resulttable_authorizationagreement_edit.description=₩ud3b8₩uc9d1

############ NEW ADDED
#### TOP MENU
navigation.csc.description=CSC

#### SUB MENU (CSC)
csc.tableSamples.description=Table Samples
csc.treeSamples.description=Tree Samples
```

4. Summary for navigation

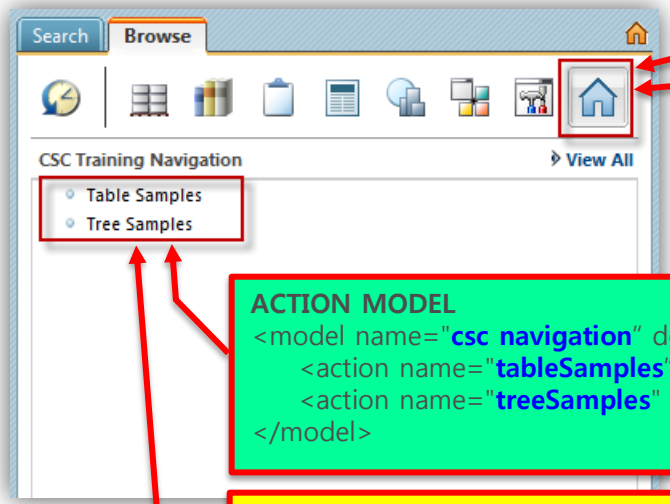
Modify action.properties for display

ACTION

```
<objecttype name="navigation" class="" resourceBundle="com.ptc.core.ui.navigationRB">
    <action name = "csc">
    </action>
</objecttype>
```

ACTION MODEL

```
<model name="main navigation">
      ....
      ....
    <action name="csc"    type="navigation"/>
</model>
```



ACTION MODEL

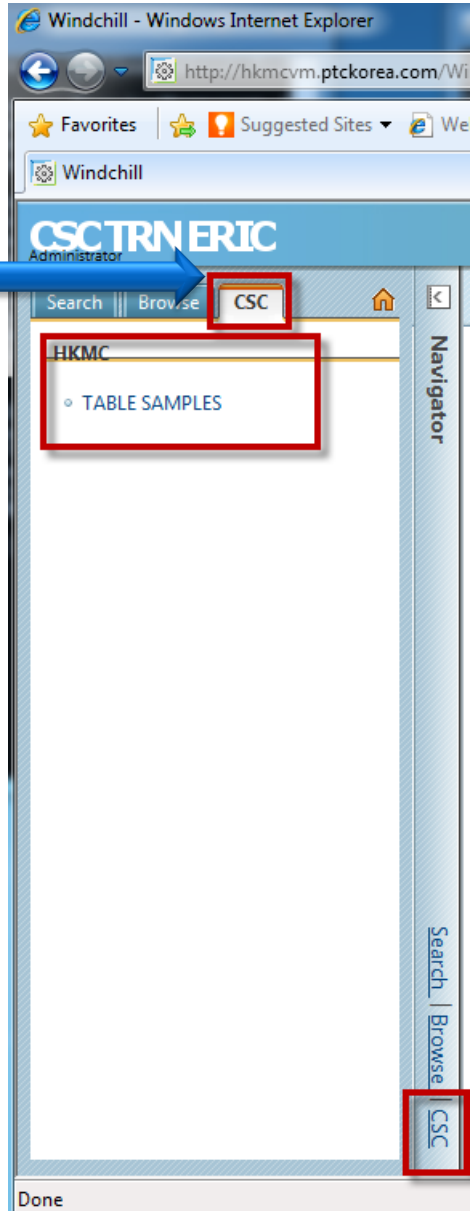
```
<model name="csc navigation" defaultActionName="tableSamples" defaultActionType="csc">
     <action name="tableSamples" type="csc" />
     <action name="treeSamples" type="csc" />
</model>
```

ACTION

```
<objecttype name="csc" class="" resourceBundle="com.ptc.core.ui.navigationRB">
    <action name="tableSamples">
        <command windowType="page" url="wtcore/jsp/csc/jca/table/table.jsp"/>
    </action>
    <action name="treeSamples">
        <command windowType="page" url="wtcore/jsp/csc/jca/tree/tree.jsp"/>
    </action>
</objecttype>
```

4. Create Tab

New Tab Menu



4. Create Tab

Add tab on actionModels.xml

Basically, when we add new tabs, it will work different functionality between OOTB

- Navigation bar action must be included URL for sub navigation lists.
- All of sub navigation will be designed on JSP pages
- Tab description (display) must be used Resource bundle (no use action.properties)

Copy <navigator> block from $WT_HOME/codebase/config/actions/navigation-actionModel.xml to csc-actionModels.xml. And then add new tab model.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <actionmodels>
4       <model name="navigator">
5          <submodel name="search navigation"/>
6          <submodel name="main navigation"/>
7          <submodel name="addtab navigation"/>
8       </model>
```

- Tab must be added on “navigator” block
- For tab, using <submodel> tag

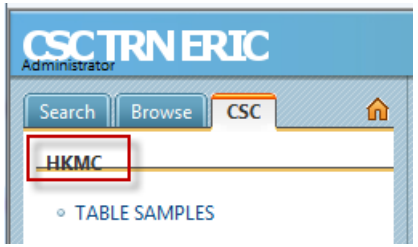
4. Create Tab

Add navigation bar for new tab on actionModels.xml

Add navigation design for new tab. In this example, we will just add one navigation.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <actionmodels>
4       <model name="navigator">
5          <submodel name="search navigation"/>
6          <submodel name="main navigation"/>
7          <submodel name="addtab navigation"/>
8       </model>
9
10      <model name="addtab navigation"
11         id="browseActions" resourceBundle="ext.csc.training.navigation.navigationRB">
12         <description>
13             For CSC TAB
14         </description>
15         <action name="HKMC" type="support"/>
16     </model>
```

- For displaying tab, you must create resource bundle



4. Create Tab

Create resource bundle for Tab

Create new resource bundle using annotation. If you want to use other language, you have to create target languages's resource bundle

```
package ext.csc.training.navigation;

import wt.util.resource.RBComment;
import wt.util.resource.RBEntry;
import wt.util.resource.RBPseudo;
import wt.util.resource.RBUUID;
import wt.util.resource.WTListResourceBundle;

@RBUUID("ext.csc.training.navigation.navigationRB")
public final class navigationRB extends WTListResourceBundle {

   @RBEntry("CSC")
   @RBComment("CSC tab on Navigator.")
   public static final String PRIVATE_CONSTANT_364 = "object.addtab navigation.description";

}
```

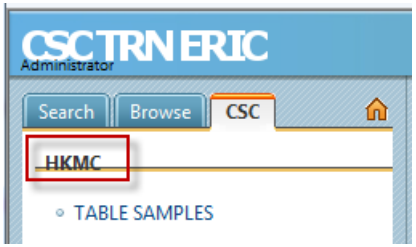
- The constant value must be included tab model name. For example, "object.addtab navigation.description"
- Resource Bundle must be created using Windchill annotation.
- Resource Bundle must be extended from WTListResourceBundle.

4. Create Tab

Add description of navigation bar to action.properties

Add a description of navigation bar on the action.properties. If you want to use other language, you have to create target languages's property file.

```
support.HKMC.tooltip=HKMC
support.HKMC.description=HKMC
```



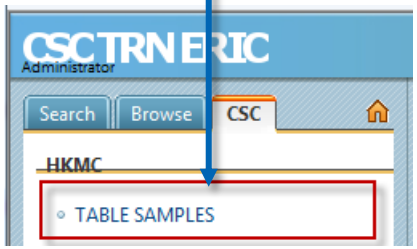
4. Create Tab

Build action for each of action model

For Navigator tab and navigation bar, we define the action. Add following block on the csc-actions.xml

```
<objecttype name="support" resourceBundle="com.ptc.core.ui.navigationRB">
    <action name="HKMC">
       <command windowType="page" url="wtcore/jsp/csc/hkmc/hkmc.jsp"/>
    </action>
 </objecttype>
```

- “addtab navigation” action doesn’t need to add on the action (tab)
- “HKMC” action must be include URL of jsp for sub-navigation body.



4. Create Tab

Build JSP for sub-navigation body

You can make sub-navigation body using JSP which are including general HTML like following.

```
<%@ taglib uri="http://www.ptc.com/windchill/taglib/components" prefix="jca"%>
<jca:tabToHighlight actionName="tableSamples" objectType="csc" />
<%@ include file="/netmarkets/jsp/util/begin.jspf"%>

<p>
<img unselectable="on" class="x-tree-node-icon x-tree-node-inline-icon" src="netmarkets/images/default_leaf.png"><a
href="/Windchill/app/#wtcore/jsp/csc/jca/tree/tree.jsp">TABLE SAMPLES</a>

<%@ include file="/netmarkets/jsp/util/end.jspf"%>
```

- JSP file must be included Windchill begin and end JSP.
- All link must use Windchill servlet path. For example “/Windchill/app/#wtcore/….”

MVC

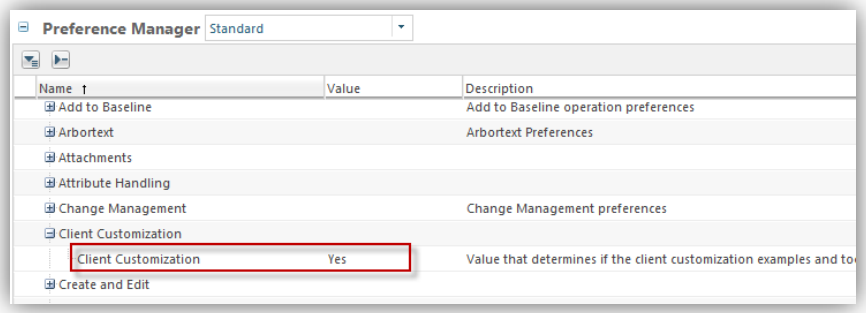
Understanding MVC

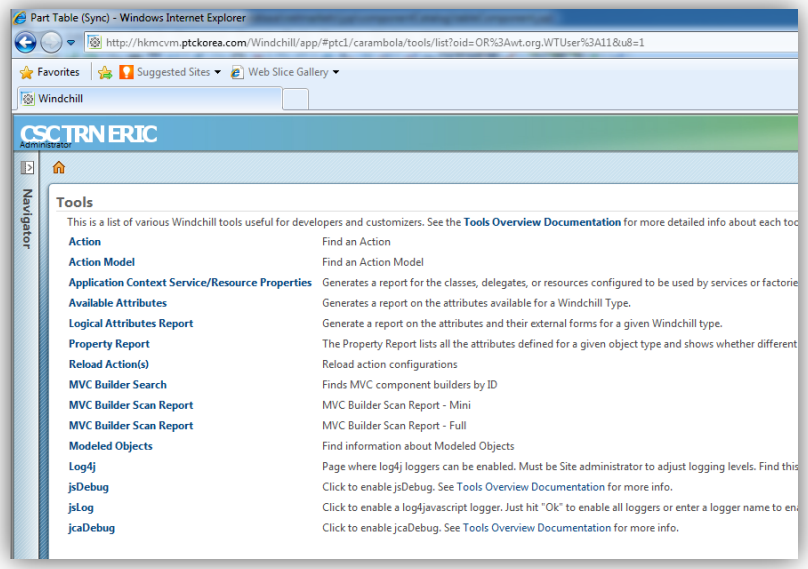
- Windchill 10 is using new architecture for UI.
  - Look attached document.
  - It is just a document to understand MVC architecture using in Windchill 10.

Microsoft Office
verPoint Presentat

- Simple understanding for implementation
  - For implementation, we just understand how to configure and call MVC function in action or JSP
  - We can use samples and tools for understanding WC10 implementation (Carambola)

- Set Carambola package
  - Open Site>Utilities>Preference Management
  - Change a value to true of Client customization





MVC Configuration

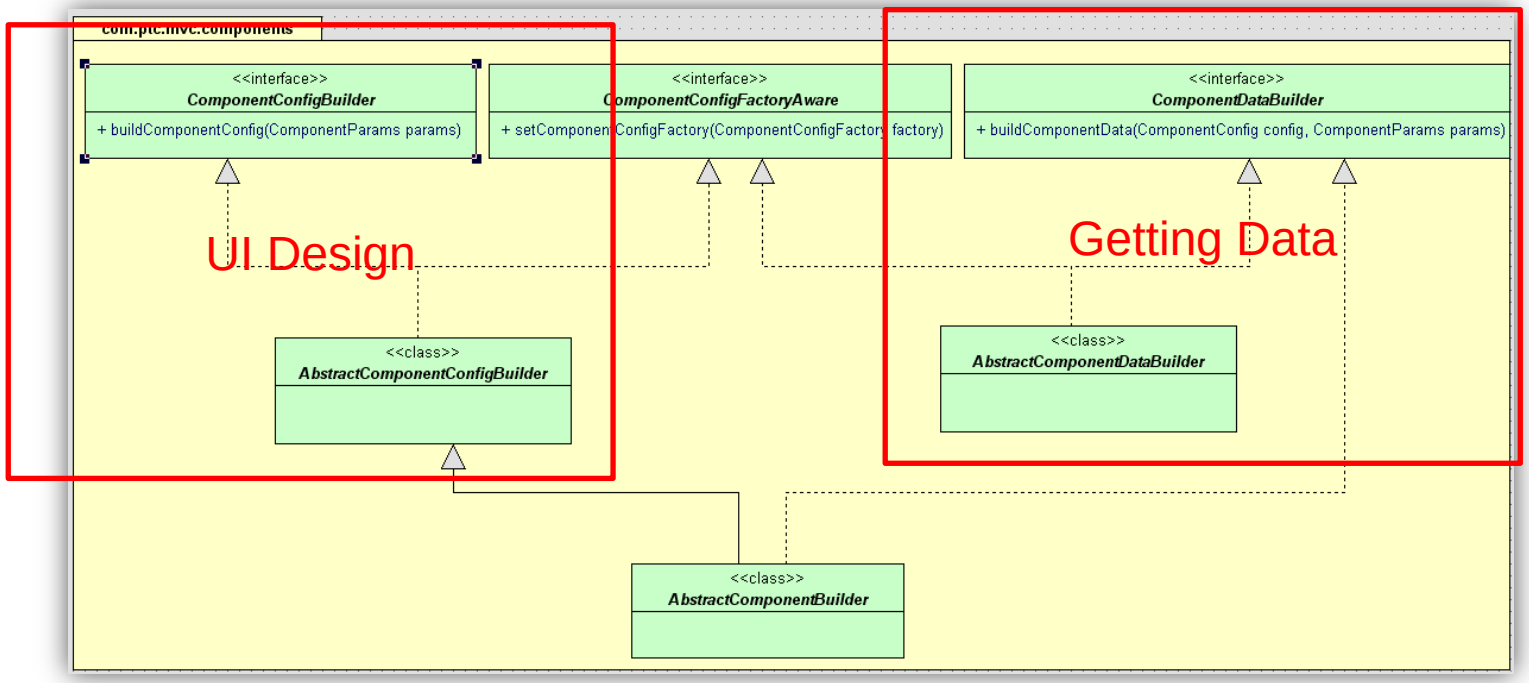
- All of component for MVC will register when starting MethodServer
  - The components list is registered in *$WT_HOME/codebase/WEB-INF/MVCDispatcher-servlet.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    - Application context definition for "MVC" DispatcherServlet.
-->

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

    <import resource="classpath:config/mvc/mvc.xml" />
    <import resource="classpath:config/mvc/jca-mvc.xml" />
    <import resource="classpath:config/mvc/*-configs.xml" />
    <import resource="classpath:config/mvc/custom.xml" />

    <bean id="defaultHandlerMappings"
        class="org.springframework.beans.factory.config.PropertiesFactoryBean">
        <property name="locations">
            <list>
                <value>
                    classpath:/config/mvc/*-urlMappings.properties
                </value>
                <value>classpath:/config/mvc/custom.properties</value>
            </list>
        </property>
    </bean>
```

  - If you want to register own XML you can add in MVCDispatcher-servlet.xml
    - For training, you have to add ***<import resource="classpath:config/mvc/csc-custom.xml"/>***
  - Generally, if you want to register your MVC component, update custom.xml
    - For training, we will use additional XML which name csc-custom.xml
    - Copy and rename from custom.xml to csc-custom.xml
    - add ***<mvc:builder-scan base-package="com.csc.mvc" />***
    - Or directly register component class ***<bean class="com.ptc.windchill.enterprise.product.mvc.builders.ProductListTableBuilder"/>***
    - After that, when starting Methodserver, all components will be loaded below “codebase/com/csc/mvc”.

How to make MVC component

- MVC component has several rules for making
  - All component must be extended OOTB abstraction builder
    - com.ptc.mvc.components.AbstractComponentBuilder
    - The builder is including config and data builder, so generally we use that abstract builder.



How to make MVC component

- MVC component has several rules for making
  - Component must be set component Id for using in action or JSP by annotation
    - Annotate it with ComponentBuilder to specify to which componentId its mapped for.
      - @ComponentBuilder(value = “<componetId>", type = ComponentBuilderType.CONFIG_ONLY)
      - @ComponentBuilder(value = “<componetId>", type = ComponentBuilderType.DATA_ONLY)
      - @ComponentBuilder(value = “{<componetId1>, <componetId2>}”)

```
//Config Builder
@ComponentBuilder(value = "carambola.mvc.table.separateBuilders", type = ComponentBuilderType.CONFIG_ONLY)
public class MvcTableConfigBuilder extends AbstractComponentConfigBuilder {
}

//Data Builder
@ComponentBuilder(value = "carambola.mvc.table.separateBuilders", type = ComponentBuilderType.DATA_ONLY)
public class MvcTableDataBuilder implements ComponentDataBuilder, ComponentConfigFactoryAware {
}

//Config + Data Builder
@ComponentBuilder("carambola.mvc.tree")
public class MvcTreeBuilder extends AbstractConfigurableTableBuilder {
}
```

How to make MVC component

- MVC component has several rules for making
  - Overriding "*buildComponentData*" function for getting data
    - Input parameter:
      - ✓ ComponentConfig
      - ✓ ComponentParams
    - Return
      - ✓ Object
  - Overriding "*buildComponentConfig*" function for design UI
    - Input parameter:
      - ✓ ComponentParams
    - Return
      - ✓ ComponentConfig

How to call MVC component in action or JSP

- Sample of MVC component

```
@ComponentBuilder("com.hkmc.cad.builder.packageHistoryTableBuilder")
public class PackageHistoryTableBuilder extends AbstractComponentBuilder {

    /** The resource. */
    private final ClientMessageSource resource = getMessageSource("com.hkmc.cad.resource.CADModuleRB");

    /*
     * (non-Javadoc)
     *
     * @see
     * com.ptc.mvc.components.ComponentDataBuilder#buildComponentData(com.ptc
     * .mvc.components.ComponentConfig, com.ptc.mvc.components.ComponentParams)
     */
    @Override
    public Object buildComponentData(ComponentConfig paramComponentConfig, ComponentParams paramComponentParams) throws Exception {
        NmHelperBean localNmHelperBean = ((JcaComponentParams) paramComponentParams).getHelperBean();
        localNmHelperBean.getRequest().setAttribute("showContextInfo", "false");
        NmCommandBean localNmCommandBean = localNmHelperBean.getNmCommandBean();
        Workable localWorkable = (Workable) localNmCommandBean.getPrimaryOid().getRefObject();
        QueryResult iterRev = HistoryTablesCommands.currentRevIterHistory(localWorkable);
        return iterRev;
    }

    /*
     * (non-Javadoc)
     *
     * @see
     * com.ptc.mvc.components.ComponentConfigBuilder#buildComponentConfig(com
     * .ptc.mvc.components.ComponentParams)
     */
    @Override
    public ComponentConfig buildComponentConfig(ComponentParams paramComponentParams) throws WTException {
        ComponentConfigFactory localComponentConfigFactory = getComponentConfigFactory();

        JcaTableConfig localJcaTableConfig = (JcaTableConfig) localComponentConfigFactory.newTableConfig();
        localJcaTableConfig.setComponentMode(ComponentMode.VIEW);
        localJcaTableConfig.setLabel(resource.getMessage(CADModuleRB.History));
        localJcaTableConfig.setAutoGenerateRowId(true);
        localJcaTableConfig.setConfigurable(true);

        ColumnConfig localColumnConfig1 = localComponentConfigFactory.newColumnConfig(HKMCConstants.Epm.Attributes.PROJECT_CODE,
```

How to make MVC component

- In buildComponentConfig function, they will use JCA component for UI design.

> JCA Components that implement ComponentConfig interface

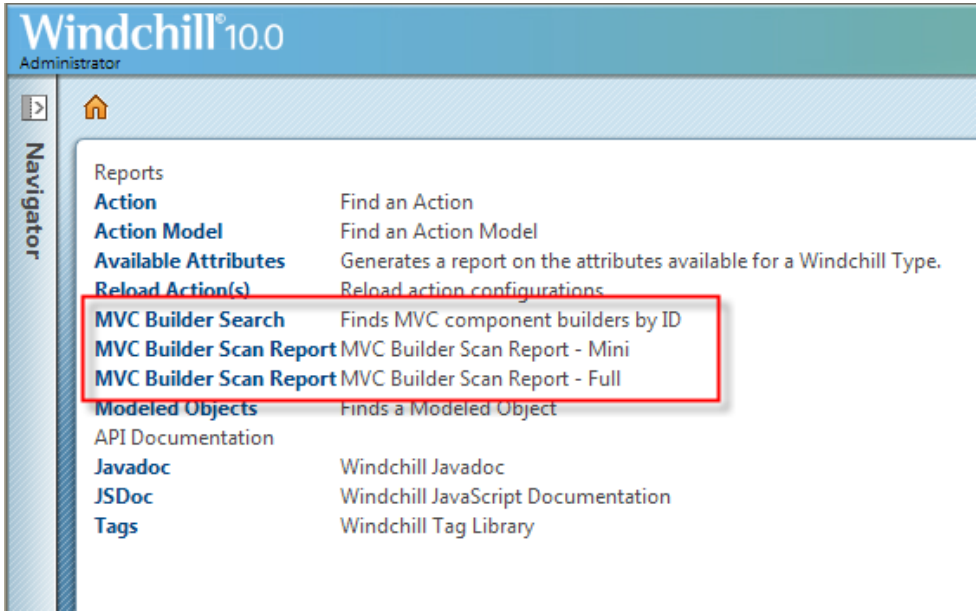
- JcaTableConfig
- JcaAttributesTableConfig
- JcaTreeConfig
- JcaAttributePanelConfig
- JcaColumnConfig
- JcaInfoConfig
- JcaPropertyConfig
- JcaPropertyPanelConfig

```
@ComponentBuilder("com.hkmc.cad.builder.packageHistoryTableBuilder")
public class PackageHistoryTableBuilder extends AbstractComponentBuilder {

@Override
    public ComponentConfig buildComponentConfig(ComponentParams paramComponentParams) throws WTException {
        ComponentConfigFactory localComponentConfigFactory = getComponentConfigFactory();

        JcaTableConfig localJcaTableConfig = (JcaTableConfig) localComponentConfigFactory.newTableConfig();
        return localJcaTableConfig;
    }
}
```

Tools

> Navigate Customization ->Tools

- MVC Builder Search : Find the Builder given a componentId
  - <demo>
- MVC Builder Scan Report : Find a repo rt on mvc-scan
  - <demo>
  - log4j.logger.com.ptc.mvc.scan=INFO should be enabled



> More Resources

- JCA MVC
- Javadoc

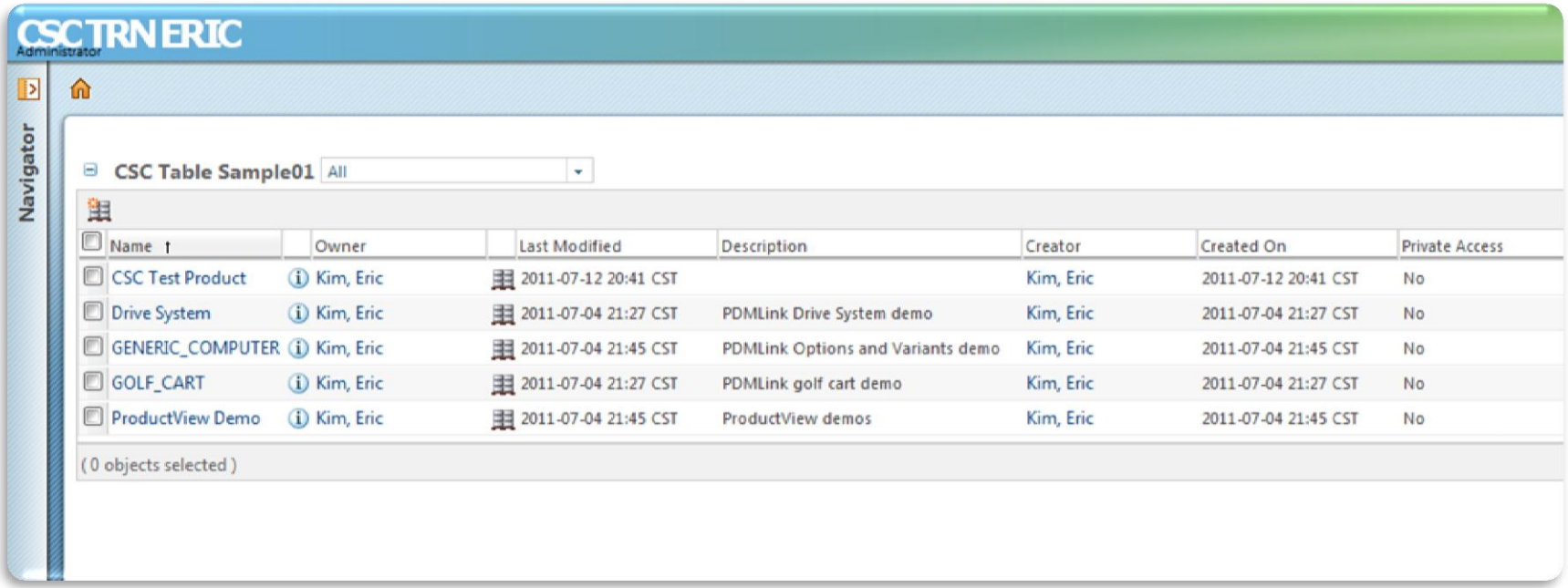
Table

Table Customization

Customization process

We will make MVC table like following process

- Register custom dispatcher
- Register custom MVC component library
- Create MVC table class for search product
- Register MVC ID to action

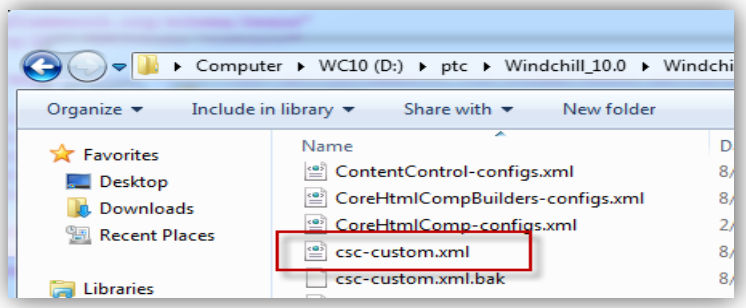


Register MVC builder path

1. Open *$WT_HOME/codebase/WEB-INF/MVCDispatcher-servlet.xml* and add following

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    - Application context definition for "MVC" DispatcherServlet.
-->

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

    <import resource="classpath:config/mvc/mvc.xml" />
    <import resource="classpath:config/mvc/jca-mvc.xml" />
    <import resource="classpath:config/mvc/*-configs.xml" />
    <import resource="classpath:config/mvc/custom.xml" />
    <import resource="classpath:config/mvc/csc-custom.xml" />

    <bean id="defaultHandlerMappings"
        class="org.springframework.beans.factory.config.PropertiesFactoryBean">
        <property name="locations">
            <list>
                <value>
                    classpath:/config/mvc/*-urlMappings.properties
                </value>
                <value>classpath:/config/mvc/custom.properties</value>
            </list>
        </property>
    </bean>


</beans>
```

2. Copy *$WT_HOME/codebase/config/mvc/custom.xml* and rename "*csc-custom.xml*"



Register MVC builder path

3. Open *$WT_HOME/codebase/config/mvc/csc-custom.xml* and add following

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.ptc.com/schema/mvc"
    xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-2.5.xsd
http://www.ptc.com/schema/mvc http://www.ptc.com/schema/mvc/mvc-10.0.xsd">

    <!-- Configurations in this file override all other configurations -->
    <mvc:builder-scan base-package="com.csc.mvc" />

</beans>
```

Create MVC Component class

4. Download sample Java from OpenGrok which is list to Product list.

http://ah-opengrok/xref/x-20-M010/wcEnterprise/ProductLibrary/src/com/ptc/windchill/enterprise/product/mvc/builders/ProductListTableBuilder.java

CSCSampleTable01.java

5. Change following about downloaded class

1. Change Class name and File name (CSCSampleTable01)
2. Change package (com.csc.mvc)
3. Change the Component annotation (csc.mvc.sampleTable01)
4. Change Table ID (csc.mvc.sampleTable01)
5. Change Label (table.setLabel("CSC Table Sample01");)

```
/* bcwti
package com.csc.mvc;

import wt.util.WTException;

/**
 * MVC builder class for Product List table
 */
@ComponentBuilder("csc.mvc.sampleTable01")
public class CSCSampleTable01 extends AbstractConfigurableTableBuilder  {
    private final ClientMessageSource messageSource = getMessageSource("com.ptc.windchill.enterprise.product.ProductClie


    /*
     * (non-Javadoc)
     * @see com.ptc.mvc.components.ComponentDataBuilder#buildComponentData(com.ptc.mvc.components.ComponentConfig, com.p
     */
    @Override
    public Object buildComponentData(ComponentConfig config, ComponentParams params) throws Exception {

        String tableId="csc.mvc.sampleTable01";
        return ProductListCommand.getProducts(tableId);
    }

    @Override
    public ComponentConfig buildComponentConfig(ComponentParams params) throws WTException {
        String helpContext ="ProductsHelp";
        ComponentConfigFactory factory = getComponentConfigFactory();
        TableConfig table = factory.newTableConfig();
        table.setId("csc.mvc.sampleTable01");
        table.setConfigurable(true);
        table.setType("wt.pdmlink.PDMLinkProduct");
```

Create MVC Component class

6. Add new action model and action

csc-actionmodels.xml

```
<model name="csc navigation" defaultActionName="treeSamples" defaultActionType="csc">
        <description>
            For CSC navigation
        </description>
        <action name="tableSample01" type="csc" />
        <action name="tableSamples" type="csc" />
        <action name="treeSamples" type="csc" />
</model>
```

csc-actions.xml

```
<objecttype name="csc" resourceBundle="com.ptc.core.ui.navigationRB">
      <action name="tableSample01">
            <component windowType="page" name="csc.mvc.sampleTable01"/>
      </action>
```

7. Update action property for display description

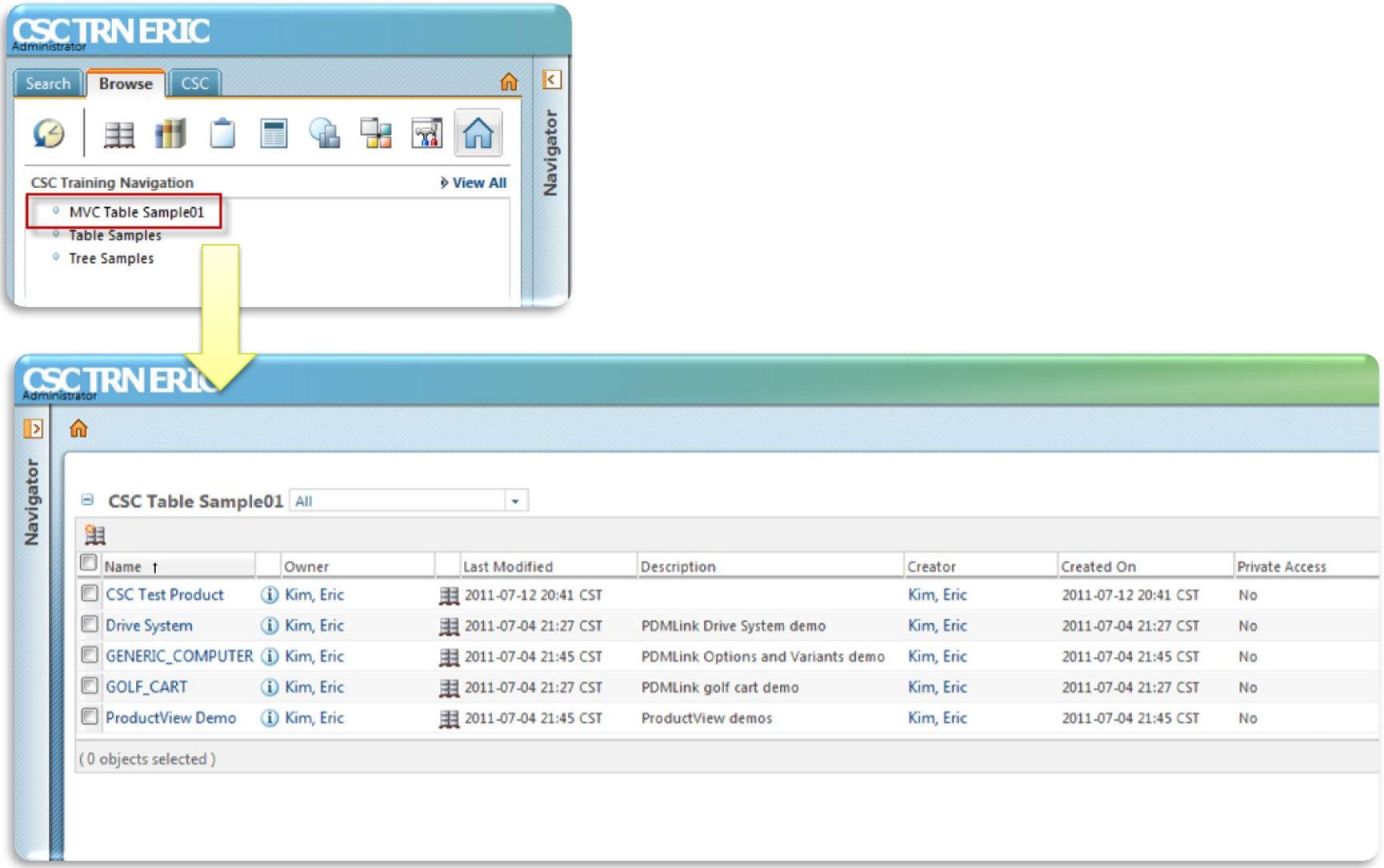
actions.properties and action_${locale}.properties

```
csc.tableSample01.description=MVC Table Sample01
```

8. Restart All service

2. Create MVC Component

Result



Search and Result Table

1. Create Populator Tag library

Populator tag and Wrapper tag

What is wrapper tag?

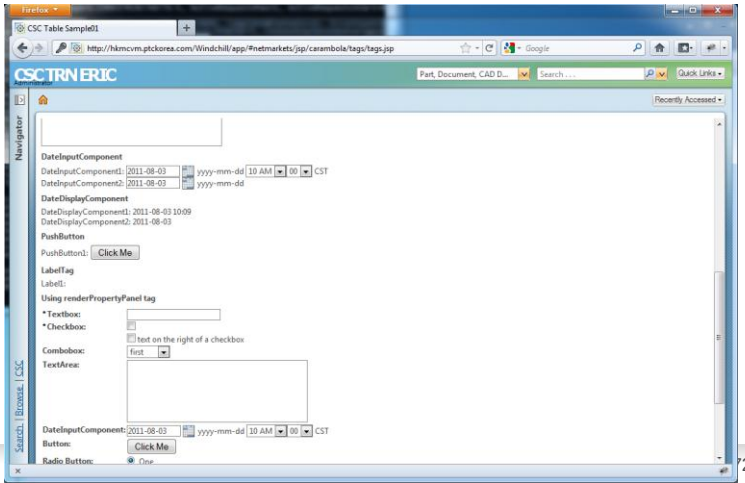
- Wrapper tag can help to make simple UI design on JSP source code
- If using wrapper tag, JSP source code doesn't need to involve getting data source code.
- Wrapper tag is OOTB tag library for just UI modeling.

What is Populator tag?

- Wrapper tag can not get data, so it needs populator tag for getting data.
- Polulator tag is not OOTB tag, so it must be created by developer. It is custom tag.

Sample page of wrapper tag

- http://localhost/Windchill/app/#netmarkets/jsp/carambola/tags/tags.jsp

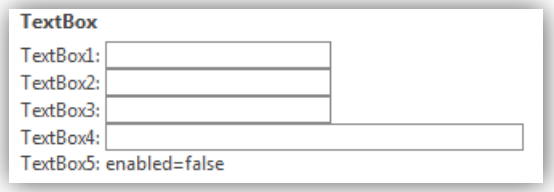


1. Create Populator Tag library

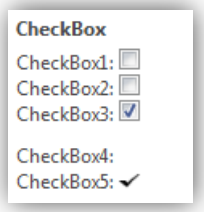
Tag type of Wrapper tag

The lists of wrapper tag

- Textbox
  - <w:textBox name="textbox1" onblur="foo"/>
  - <w:textBox name="textbox2" onblur="foo" required="true"/>
  - <w:textBox name="textbox3" onblur="foo" styleClass="boo bar"/>
  - <w:textBox name="textbox4" size="40" maxlength="50"/>
  - <w:textBox name="textbox5" hidden="true" enabled="false" value="enabled=false"/>



- CheckBox
  - <w:checkBox name="checkbox1" onblur="foo" renderExtra="whereDoesThisGo"/>
  - <w:checkBox name="checkbox2" onblur="foo" required="true" submitAsNew="true"/>
  - <w:checkBox name="checkbox3" onblur="foo" styleClass="boo bar" checked="true"/>
  - <w:checkBox name="checkbox4" enabled="false" />
  - <w:checkBox name="checkbox4" enabled="false" checked="true"/>



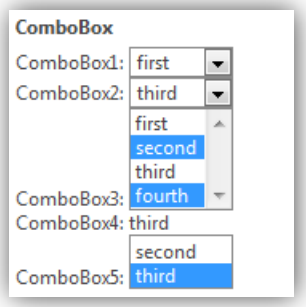
1. Create Populator Tag library

Tag type of Wrapper tag

The lists of wrapper tag

- ComboBox
  - <w:comboBox name="combobox" onselect="lauch()" internalValues="${ivals}" displayValues="${dvals}" />
  - <w:comboBox name="combobox2" onmousedown="doFoo()" required="true" internalValues="${ivals}" displayValues="${dvals}" selectedValues="${svals}"/>
  - <w:comboBox name="combobox3" onmousemove="doBar()" styleClass="boo bar" multiSelect="true" internalValues="${ivals}" displayValues="${dvals}" selectedValues="${multivals}"/>
  - <w:comboBox name="combobox4" enabled="false" internalValues="${ivals}" displayValues="${dvals}" selectedValues="${svals}"/>
  - <w:comboBox name="combobox5" size="2" internalValues="${ivals}" displayValues="${dvals}" selectedValues="${svals}"/>

  - ComboBox needs populate value, for example, ${dvals}, ${svals} will set on populate tag process



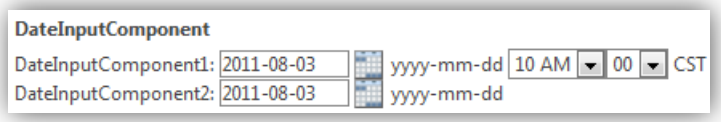
1. Create Populator Tag library

Tag type of Wrapper tag

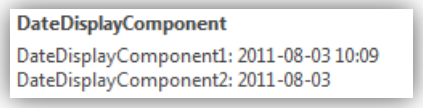
The lists of wrapper tag

- TextArea
  - <w:textArea id="message" name="message" value="" cols="39" rows="5" required="false"/>

- DateInputComponent
  - <w:dateInputComponent name="startdate" required="true" dateValueType="DATE_TIME"/>
  - <w:dateInputComponent name="startdate" required="true" dateValueType="DATE_ONLY"/>



- DateDisplayComponent
  - <w:dateDisplayComponent name="datetime" dateDisplayFormat="${standardDateTime}" dateValue="${date}"/>
  - <w:dateDisplayComponent name="datetime2" dateDisplayFormat="${standardDate}" dateValue="${date}"/>

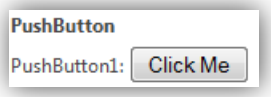


1. Create Populator Tag library

Tag type of Wrapper tag

The lists of wrapper tag

- PushButton
  - <w:button name="button1" value="Click Me" typeSubmit="false" toolTipText="Click Me" onclick="alert('clicked')"/>



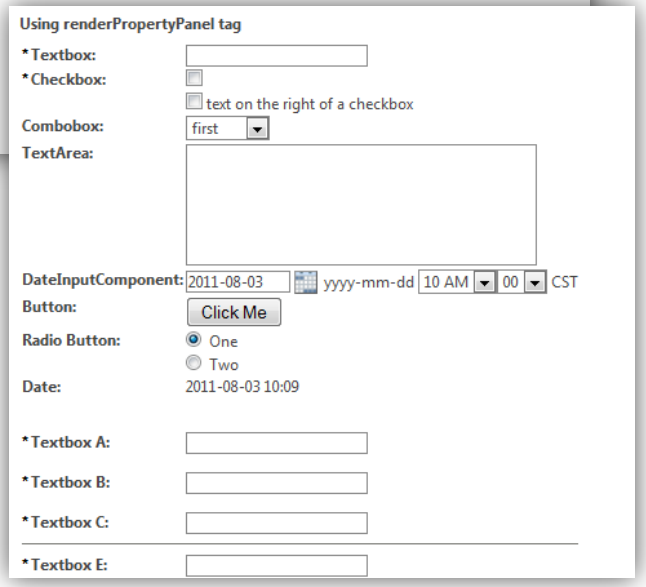
- Label
  - <w:label value="${labelCc}" styleClass="boo bar"/>

1. Create Populator Tag library

Using RenderPropertyPanel JCA Tag

When you are using RenderPropertyPanel JCA tag, by this action, all wrapper tag will stand in line.

```
<jca:renderPropertyPanel>
    <jca:addHeader text="Adding a bunch of different gui components..." />
    <w:textBox propertyLabel="Textbox" name="textboxA" onblur="foo" required="true"/>
    <w:checkBox propertyLabel="Checkbox" name="checkboxA" onblur="foo" required="true" submitAsNew="true"/>
    <w:checkBox label="text on the right of a checkbox" name="checkboxB" onblur="foo" required="false" submitAsNew="true" renderLabel="true" renderLabelOnRight="true" />
    <w:comboBox propertyLabel="Combobox" name="comboboxA" onselect="lauch()" internalValues="${ivals}"  displayValues="${dvals}" />
    <w:textArea propertyLabel="TextArea" id="messageA" name="messageA" value="" cols="39" rows="5" required="false"/>
    <w:dateInputComponent propertyLabel="DateInputComponent" name="startdateA" required="true" dateValueType="DATE_TIME"/>
    <w:button propertyLabel="Button" name="button1" value="Click Me" typeSubmit="false" toolTipText="Click Me" onclick="alert('clicked')"/>
    <w:radioButton propertyLabel="Radio Button" label="One" value="123" name="radio" checked="true" onclick="doFoo()"/>
    <w:radioButton label="Two" value="456" name="radio" onclick="doFoo()"/>
    <w:dateDisplayComponent propertyLabel="Date" name="datetime"  dateDisplayFormat="${standardDateTime}" dateValue="${date}"/>
    <jca:addSeparator/>
    <jca:addSeparator/>
    <jca:addHeader text="Using separators..." />
    <w:textBox propertyLabel="Textbox A" name="textboxA" onblur="foo" required="true"/>
    <jca:addSeparator/>
    <w:textBox propertyLabel="Textbox B" name="textboxB" onblur="foo" required="true"/>
    <jca:addSeparator style="blank"/>
    <w:textBox propertyLabel="Textbox C" name="textboxC" onblur="foo" required="true"/>
    <jca:addSeparator style="line"/>
    <w:textBox propertyLabel="Textbox E" name="textboxE" onblur="foo" required="true"/>
</jca:renderPropertyPanel>
```



1. Create Populator Tag library

Create custom tag for populator

1. Create new file on $WT_HOME/codebase/WEB-INF/tlds, and modify

csc.tld

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

<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
    version="2.0">

    <description>
                  Search  TLD for  CSC Training
    </description>
    <display-name>CSC Tag Library</display-name>
    <tlib-version>1.1</tlib-version>
    <short-name>csc</short-name>
    <uri>http://www.ptc.com/windchill/taglib/csc</uri>

    <tag>
        <description>
            Search data tag for Document
        </description>
        <name>searchPopulator</name>
        <tag-class>com.csc.common.tags.SearchDataPopulatorTag</tag-class>
        <body-content>empty</body-content>
        <attribute>
            <description>
               Name of data populator
            </description>
            <name>name</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
        </attribute>

    </tag>
</taglib>
```

1. Create Populator Tag library

PTC®

Create custom tag for populator

2. Create populator interface class

SearchDataPopulator.class

```
package com.csc.common.tags;

import javax.servlet.jsp.JspContext;

public interface SearchDataPopulator {
    public static final String VERSION = "$Id: $";

    public void populateSearchCriteria(JspContext jspContext);
}
```

1. Create Populator Tag library

PTC®

Create custom tag for populator

3. Create populator tag class

SearchDataPopulatorTag.class

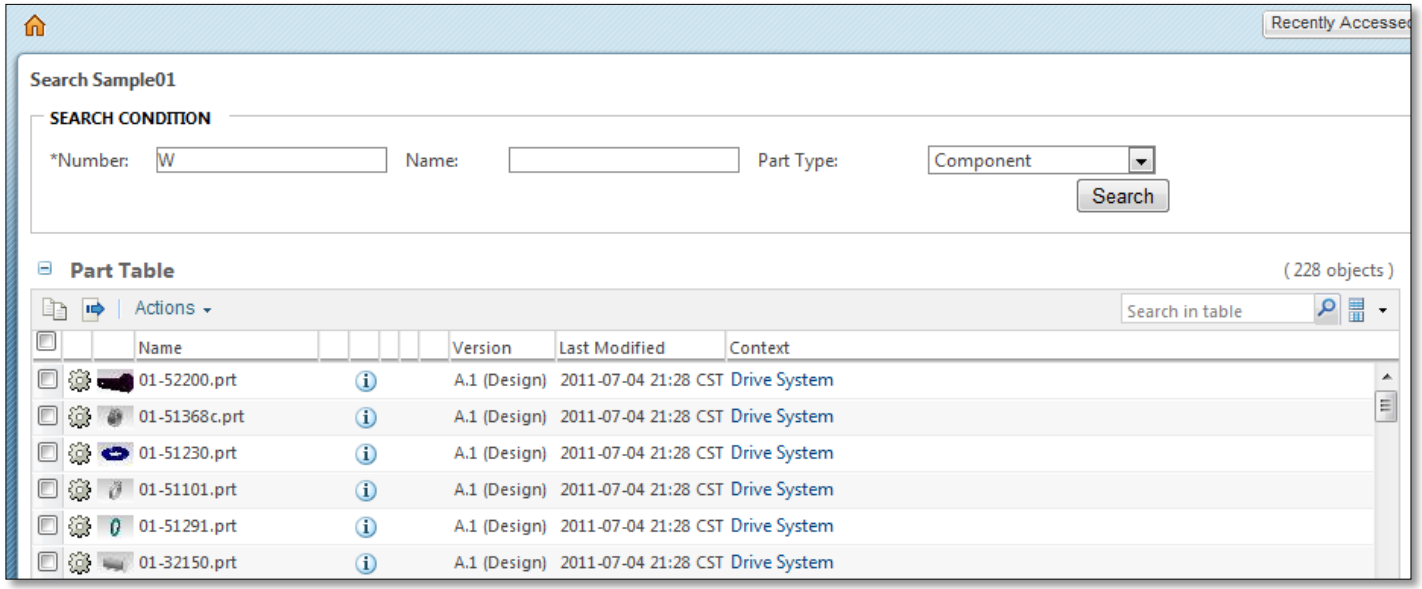
```
package com.csc.common.tags;

import java.io.IOException;

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.SimpleTagSupport;
import com.csc.common.tags.SearchDataPopulator;

public class SearchDataPopulatorTag extends SimpleTagSupport {
public static final String VERSION = "$Id: $";

    /**
     * Search Data populator classname
     */
    private String name;

    @Override
    public void doTag() throws JspException, IOException {
        SearchDataPopulator populator = null;
        try {
            final Class<?> clas = Class.forName(name);
            populator = (SearchDataPopulator) clas.newInstance();

        } catch (final ClassNotFoundException e) {
            e.printStackTrace();
        } catch (InstantiationException e) {
            e.printStackTrace();
        } catch (final IllegalAccessException e) {
            e.printStackTrace();
        }

        if (populator != null) {
            populator.populateSearchCriteria(getJspContext());
        } else {
            throw new JspException("Unable to initialize search populator:\t" + name);
        }
    }

    public String getName() {
        return name;
```

2. Create search and table

Scenario for search and table

Before we understand how to make search and table, we must have one scenario like following.

- Search target will be all WTPart object.
- Search criteria needs to be number, name and Part Type.
- Number field must be required field
- Part Type must be designed ComboBox
- Result table must be show like Part list.
  - Number, Name, Small thumbnail, etc..



2. Create search and table

Build JSP for criteria

1. Create JSP and set basic environment

sampleSearch01.jsp

```
<%@ taglib prefix="jca" uri="http://www.ptc.com/windchill/taglib/components"%>
<%@ include file="/netmarkets/jsp/util/begin.jspf"%>
<%@ taglib prefix="mvc" uri="http://www.ptc.com/windchill/taglib/jcaMvc"%>
<%@ taglib prefix="wrap" uri="http://www.ptc.com/windchill/taglib/wrappers"%>
<%@ taglib prefix="csc" uri="http://www.ptc.com/windchill/taglib/csc"%>
<%@ taglib tagdir="/WEB-INF/tags" prefix="wctags"%>

<%@ include file="/netmarkets/jsp/util/end.jspf"%>
```

2. Add populator tag for get criteria data set

sampleSearch01.jsp

```
<%@ taglib prefix="jca" uri="http://www.ptc.com/windchill/taglib/components"%>
<%@ include file="/netmarkets/jsp/util/begin.jspf"%>
<%@ taglib prefix="mvc" uri="http://www.ptc.com/windchill/taglib/jcaMvc"%>
<%@ taglib prefix="wrap" uri="http://www.ptc.com/windchill/taglib/wrappers"%>
<%@ taglib prefix="csc" uri="http://www.ptc.com/windchill/taglib/csc"%>
<%@ taglib tagdir="/WEB-INF/tags" prefix="wctags"%>

<csc:searchPopulator name="com.csc.search.SampleSearchDataPopulator01" />

<%@ include file="/netmarkets/jsp/util/end.jspf"%>
```

2. Create search and table

Build JSP for criteria

3. Create populator class for sample criteria

SampleSearchDataPopulator01.class

```
package com.csc.search;

import java.util.ArrayList;
import java.util.List;
import javax.servlet.jsp.JspContext;

import com.csc.common.tags.SearchDataPopulator;

public class SampleSearchDataPopulator01 implements SearchDataPopulator {
    public static final String VERSION = "$Id: $";

    @Override
    public void populateSearchCriteria(JspContext jspContext) {
        // TODO Auto-generated method stub
        try {
            // Set all of key in here
            //Sample01 - Part Type Population
            List<String> partTypeKey = new ArrayList<String>();
            List<String> partTypeDisplay = new ArrayList<String>();
            partTypeKey.add("");
            partTypeDisplay.add("");

            PartType[] partSet = PartType.getPartTypeSet();

            for( int i=0; i < partSet.length; i++) {
                partTypeKey.add(partSet[i].getStringValue().substring(partSet[i].getStringValue().lastIndexOf(".")+1));
                partTypeDisplay.add(partSet[i].getDisplay());
            }

            jspContext.setAttribute("partType_internal_vals", partTypeKey);
            jspContext.setAttribute("partType_display_vals", partTypeDisplay);
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

2. Create search and table

Build JSP for criteria

4. Build criteria UI in JSP page

Add following source in “sampleSearch01.jsp”


```
<b>Search Sample01</b>
<fieldset class="x-fieldset x-form-label-left" id="Visualization_and_Attributes" style="width: 1024px;">
    <legend>SEARCH CONDITION</legend>
        <table>
            <tr>
                <td scope="row" width="50" class="tableColumnHeaderfont" align="left">*Number:</td>
                <td class="tabledatafont" align="left"> 
                    <wrap:textBox name="number" id="number" maxlength="30" size="10"
                            nblur="this.value=this.value.toUpperCase()" />
                </td>

                <td scope="row" width="50" class="tableColumnHeaderfont" align="left">Name:</td>
                <td class="tabledatafont" align="left"> 
                    <wrap:textBox name="name" id="name" maxlength="100" size="20"/>
                </td>

                <td scope="row" width="100" class="tableColumnHeaderfont" >Part Type:</td>
                <td class="tabledatafont" align="left">
                    <wrap:comboBox id="partType" name="partType" multiSelect="false" size="1"
                        displayValues="${partType_display_vals}"
                        internalValues="${partType_internal_vals}" />
                </td>
            </tr>

            <tr>
                <td colspan="6" scope="row" class="tableColumnHeaderfont" align="right">
                    <wrap:button name="search” value='Search’ onclick="submitDocSearch();" />
                </td>
            </tr>
        </table>
</fieldset>
```

2. Create search and table

Build JSP for criteria

searchSample01.jsp

5. Add MVC table and javascript function in JSP page

Add following source in “sampleSearch01.jsp”

```
<mvc:tableContainer compId="csc.mvc.sampleSearch01" height="500" />

<%@ include file="/netmarkets/jsp/util/end.jspf"%>

<script>

            function submitDocSearch(){
                        if (document.getElementById('number').value==""){
                                    JCAAlert("Number must be inputted");
                        } else {
                                    submitParameters();
                        }
            }

            function submitParameters() {
                        var number = document.getElementById('number').value;
                        var name = document.getElementById('name').value;
                        var partType = document.getElementById('partType').value;

                        var params = {
                                                number : number,
                                                name : name,
                                                partType : partType

                        };

                        PTC.jca.table.Utils.reload('csc.mvc.sampleSearch01', params, true);
            }

</script>
```

MVC table

Recall MVC table using parameters

Build MVC Table

6. Create MVC Class

Before creating MVC class, you must register MVC directory. If you do not want to register directory, you can directly register new MVC class.

```
1  /* bcwti
12 package com.csc.mvc;
13
14 import static com.ptc.core.components.descriptor.DescriptorConstants.ColumnIdentifiers.CHANGE_STATUS_FAMILY;
46
47 /**
48  * MVC builder class for Product List table
49  */
50 @ComponentBuilder("csc.mvc.sampleSearch01")
51 public class CSCSampleSearch01 extends AbstractComponentBuilder  {
52
53     private static final String RESOURCE = "com.ptc.carambola.carambolaResource";
54     /*
55      * (non-Javadoc)
56      * @see com.ptc.mvc.components.ComponentDataBuilder#buildComponentData(com.ptc.mvc.components.ComponentConi
57      */
58     @Override
59     public Object buildComponentData(ComponentConfig config, ComponentParams params) throws Exception {
60         String number = (String) params.getParameter("number");
61         String name = (String) params.getParameter("name");
62         String partType = (String) params.getParameter("partType");
63
64         if ( number != null && number.length() > 0 )  {
65             number = number.trim();
66         } else {
```

```
97
98     @Override
99     public ComponentConfig buildComponentConfig(ComponentParams params) throws WTException {
100        ComponentConfigFactory factory = getComponentConfigFactory();
101        ClientMessageSource messageSource = getMessageSource(RESOURCE);
102
103        TableConfig table = factory.newTableConfig();
104        table.setLabel(messageSource.getMessage("PART_TABLE_LABEL"));
```

1. Set package which is registered in mvc.xml
2. Create annotation for MVC component ID
3. Set Java Class name which extended “AbstractComponentBuilder”
4. Override “buildComponentData” which function will be prgorammed to get data by inputted parameters.
5. Override “buildComponentConfig” fucinton which will work to design UI.

CSCSampleSearch01.java

Other Table feature

1. JCA MVC Table Features – New & Customizable

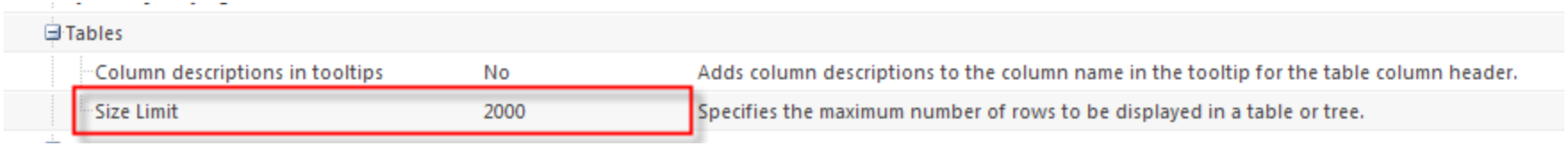
PTC

> Paging

- Paging concept removed in X20
- DataSource enabled table : client will load till the “size limit”.
- Non DataSource table : client will load all the data.

> Table Size Limit

- Preference driven (Visible in all Organization and Site Utilities Preference Managers)
- Value range : 1 to 100,000
- OOTB value is 2,000



- DataBuilder to extend AbstractJcaDataSourceComponentDataBuilder
  - Override getComponentLimit(ComponentConfig config,ComponentParams params)

2. JCA MVC Table Features – Sorting

PTC

> Sorting

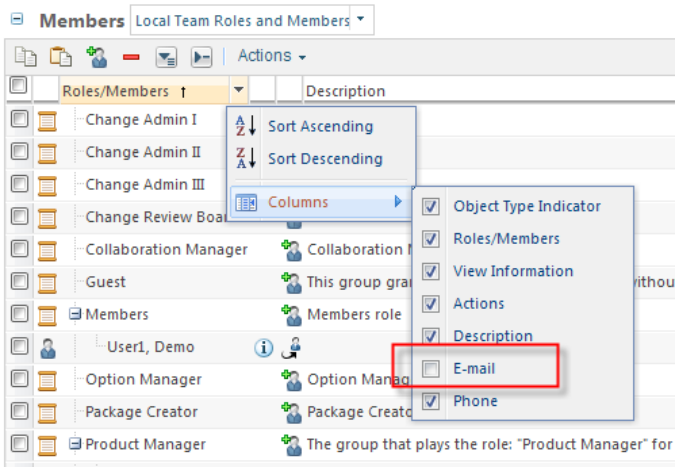
- Non DataSource & DataSource.SYNCHRONOUS table
  - Table view sort criteria is applied in server.
  - User sort applied in data that is available in the client only.
- DataSource.ASYNCHRONOUS table
  - By default Table view sort criteria is applied in client.
  - DataBuilder can send sorted data from the server and inform the Infrastructure
    - ComponentResultProcessor. setPresorted(boolean preSorted)
- How to specify client sort function for a column?
  - ColumnConfig.setCompareJsFunction(String compareJsFunction)

```
//Define the column
ColumnConfig col = factory.newColumnConfig("Quantity", columnLabel, true);
col.setNeed("quantityWithUnits");
col.setDataUtilityId("part.report.quantity");
//set the compareJsFunction
col.setCompareJsFunction("prc.quantityComparator");

// javascript function available in the view
quantityComparator : function(cellA, cellB, rowA, rowB) {
    var comparableA = jsca.getComparable(cellA);
    var comparableB = jsca.getComparable(cellB);
    ----
},
```

3. JCA MVC Table Features – Columns

> Hidden Column

- In X12, user cannot make it visible in the client; whe reas in X20 its possible.
- Hidden columns are not displayed in the table OOT B. They can be made visible from the menu availabl e in column header.
- ColumnConfig.setHidden(boolean hidden)

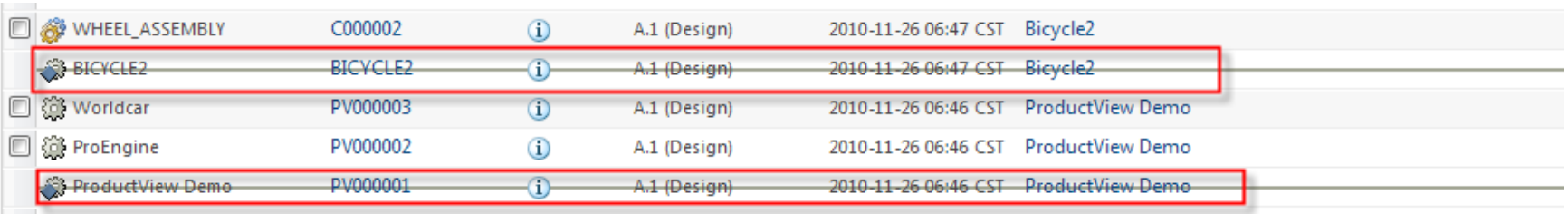


> Data Store Column

- Data Store columns cannot be displayed in the table
- They are available in the client (store).
- Mainly used to send some custom values for the ro w.
- ColumnConfig.setDataStoreOnly(boolean forDataSt oreOnly)

4. JCA MVC Table Features – Strike through

> Strike through rows

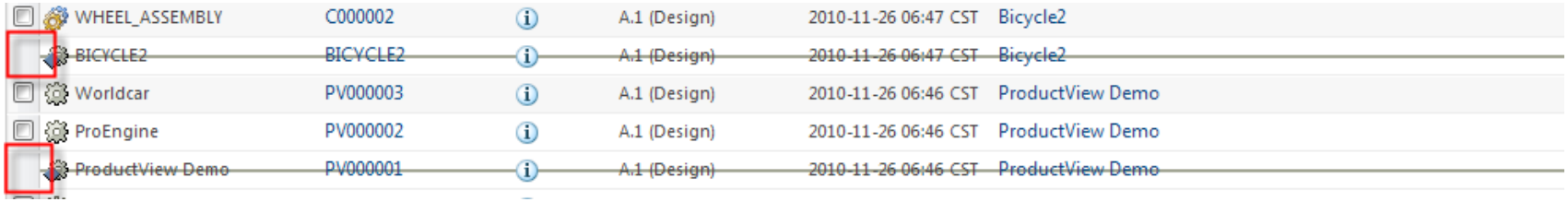


– JcaTableConfig.setStrikeThroughColumn(ColumnConfig column)

```
// strikeThrough column handling : "strikeThroughRow" should be the ID to which the default DataUtility is
// mapped to
ColumnConfig strikeThroughColumn = factory.newColumnConfig("strikeThroughRow", false);
strikeThroughColumn.setDataStoreOnly(true);
// map it to endItem attribute
strikeThroughColumn.setNeed("endItem");
// add the column
table.addComponent(strikeThroughColumn);
//mark the column as StrikeThroughColumn
table.setStrikeThroughColumn(strikeThroughColumn);
```

5. JCA MVC Table Features – Non Selectable

> Non-selectable rows

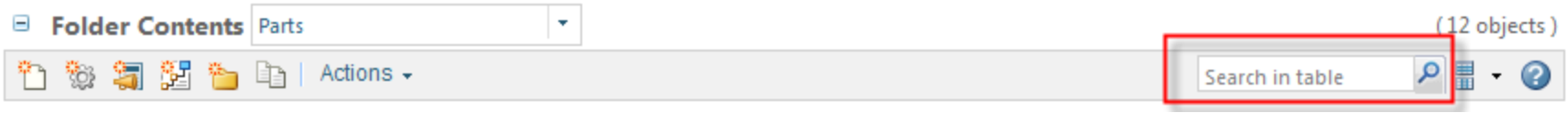


– JcaTableConfig.setNonSelectableColumn(ColumnConfig column)

```
// nonSelectable column handling : "nonSelectableColumn" should be the ID to which the default DataUtility
// is mapped to
ColumnConfig nonSelectableColumn = factory.newColumnConfig
                                   (DescriptorConstants.ColumnIdentifiers.NON_SELECTABLE_COLUMN, false);
nonSelectableColumn.setDataStoreOnly(true);
// map it to endItem attribute
nonSelectableColumn.setNeed("endItem");
// add the column
table.addComponent(nonSelectableColumn);
//mark the column as NonSelectableColumn
table.setNonSelectableColumn(nonSelectableColumn);
```

6. JCA MVC Table Features – Find in Table

> Find in Table / Search in Table

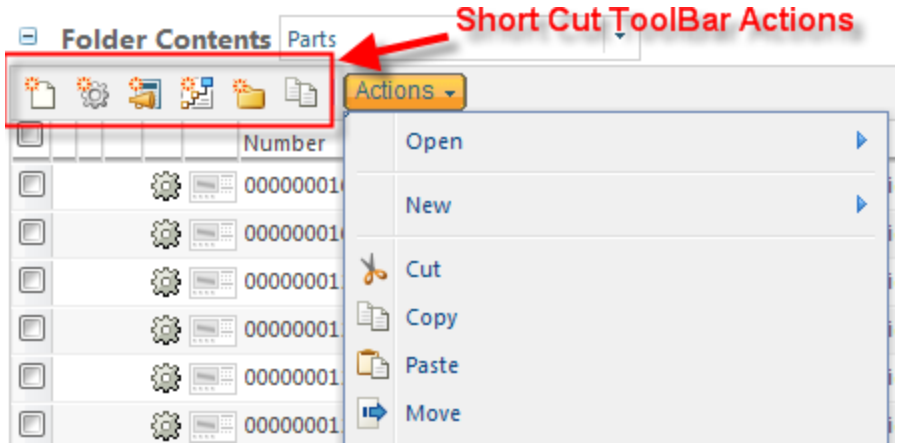


- Configured to disable or find in client or find in client & server
- JcaTableConfig.setFindInTableMode(FindInTableMode findInTableMode)
- Default is FindInTableMode.CLIENT_AND_SERVER

7. JCA MVC Table Features – ToolBar Actions

> Table ToolBar Actions

```
<model name="folderbrowser_toolbar_actions">
   <submodel name="folderbrowser_toolbar_open_submenu"/>
   <action name="separator" type="separator"/>
   <submodel name="folderbrowser_toolbar_new_submenu"/>
   <action name="separator" type="separator"/>
   //following actions are not shortcut
   <action name="list_cut" type="object"/>
   <action name="pasteAsCopy" type="saveas"/>
   ----
</model>
```




```
<model name="folderbrowser_toolbar_new_submenu" resourceBundle="com.ptc.win
dchill.enterprise.folder.FolderActionResource">
   //mark the following actions as shortcut
   <action name="create" type="document" shortcut="true"/>
   <action name="createPartWizard" type="part" shortcut="true"/>
   ----
</model>
```

8. JCA MVC Table Client Side Events

PTC®

> Some major client events fired during table render/data processing

Event name	Target object	Event scenario
dataChanged	Ext.data.store	Fired when a data chunk is received at client.
dataSourceComplete	Ext.data.store	Fired when the last data chunk is recei ved at client.
refresh	Ext.grid.GridView	Fired when the grid view refreshes to di splay the additional row data.
render	Ext.grid.GridPanel	Fired when Ext-JS finishes the grid ren dering.
renderTableStart	Ext.grid.GridPanel	Fired when the infrastructure starts ren dering of the table.
renderTableEnd	Ext.grid.GridPanel	Fired when the infrastructure ends the r endering of the table.
DSLoadingInterupted	Ext.data.store	Fired when table has partial data.

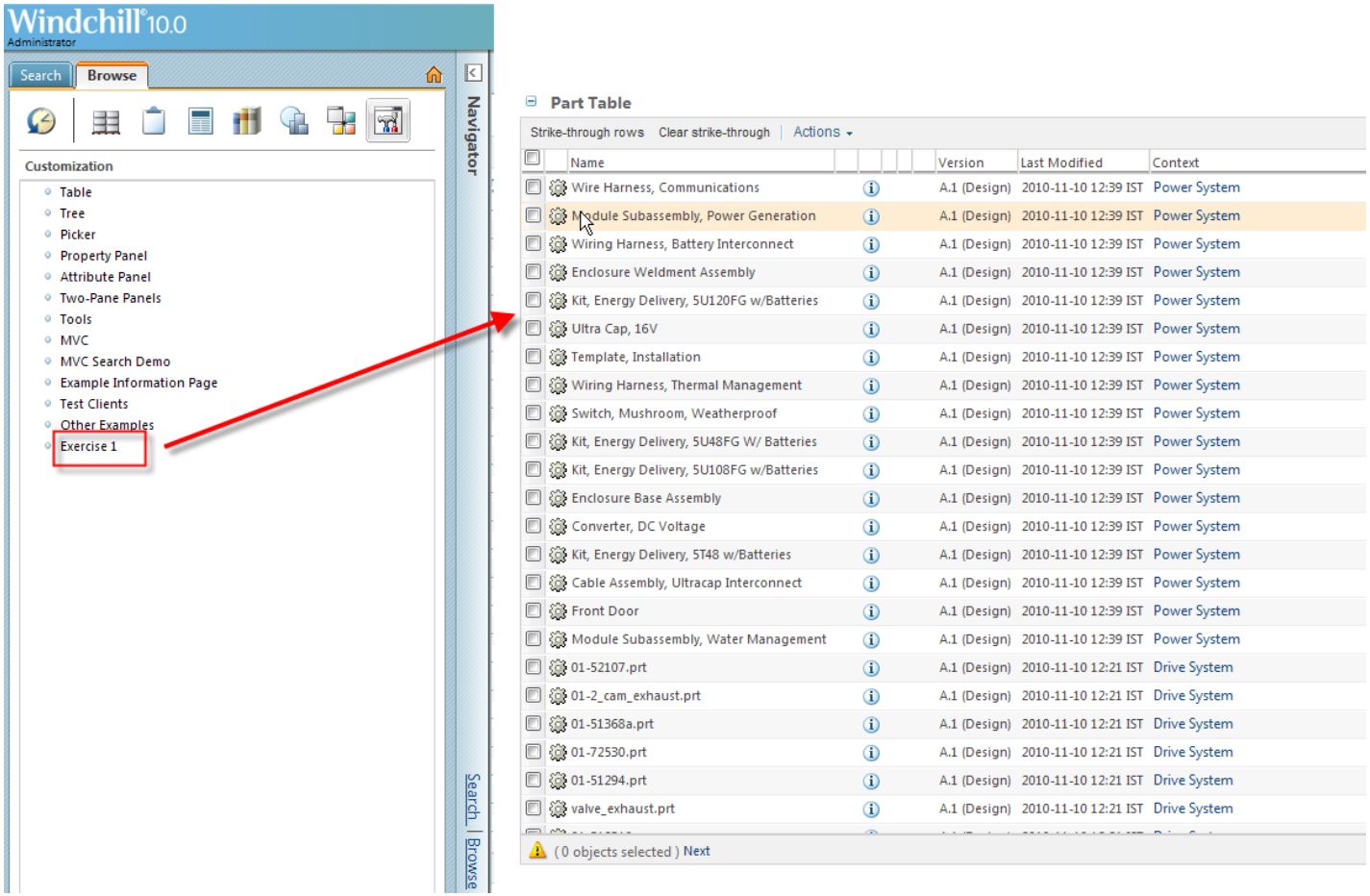
9. Registering listeners

> In your view jsp

```
<%-- register the listeners --%>
<script type="text/javascript">
   PTC.onReady(function() {
      Ext.ComponentMgr.onAvailable(<table id>, function(){

         //register on renderTableEnd
         this.on("renderTableEnd", <function>);

         //register on dataSourceComplete
          this.getStore().on("dataSourceComplete", <function>);
         });
      });
   });
</script>

<%-- render the table --%>
```

Lab session – Exercise 1

> Create a page with one MVC JCA Table which contains the list of parts



Exercise 1 : Builders

PTC

> Configure an action that generate a MVC url, that builds a table with componentId “exercise1”
  - Define the new action in WT_HOME/codebase/config/actions/custom-actions.xml
  - Find the definition of “Customization” action model
  - Override it in WT_HOME/codebase/config/actions/custom-actionModels.xml
  - Load the updated action configurations.

> Create a Table builder
  - Base package com.ptc.mvc.builders
  - Extend AbstractComponentBuilder
  - Annotate it with @ComponentBuilder
  - Provide the ComponentConfig
  - Provide the Data
  - Compile the class and restart the MethodServer

Exercise 1 : Code Snippets

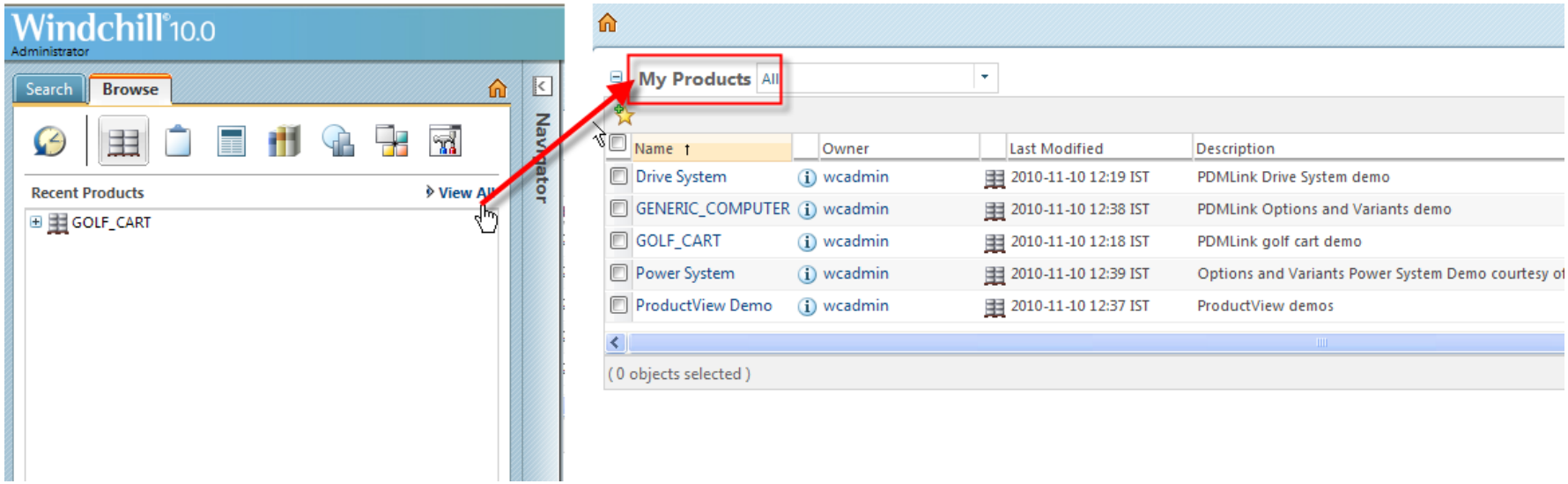
```
private static final String RESOURCE = "com.ptc.carambola.carambolaResource";

    @Override
    public ComponentConfig buildComponentConfig(ComponentParams params) throws WTException {

        ComponentConfigFactory factory = getComponentConfigFactory();
        ClientMessageSource messageSource = getMessageSource(RESOURCE);

        TableConfig table = factory.newTableConfig();
        table.setLabel(messageSource.getMessage("PART_TABLE_LABEL"));

        table.setSelectable(true);
        // set the actionModel that comes in the TableToolBar
        table.setActionModel("mvc_tables_toolbar");

        // add columns
        table.addComponent(factory.newColumnConfig(ICON, true));
        table.addComponent(factory.newColumnConfig(NAME, true));
        table.addComponent(factory.newColumnConfig(FORMAT_ICON, false));

        table.addComponent(factory.newColumnConfig(ORG_ID, false));
        table.addComponent(factory.newColumnConfig(INFO_ACTION, false));
        ColumnConfig nmActionsCol = factory.newColumnConfig(NM_ACTIONS, false);
        // specify the actionModel for the action column
        ((JcaColumnConfig) nmActionsCol) .setActionModel("CustEx_table_row_actions");
        table.addComponent(nmActionsCol);

        return table;
}
```

Exercise 1 : Code Snippets

```
@Override
    public QueryResult buildComponentData(ComponentConfig config, ComponentParams params) throws WTException {
        QuerySpec qs = new QuerySpec(WTPart.class);
        qs.setQueryLimit(10000);
        return PersistenceHelper.manager.find((StatementSpec) qs);
    }
```

Lab session – Exercise 2

> Override the OOTB builder for Product List table to show its label as “My Products”



Exercise 2 : Builders

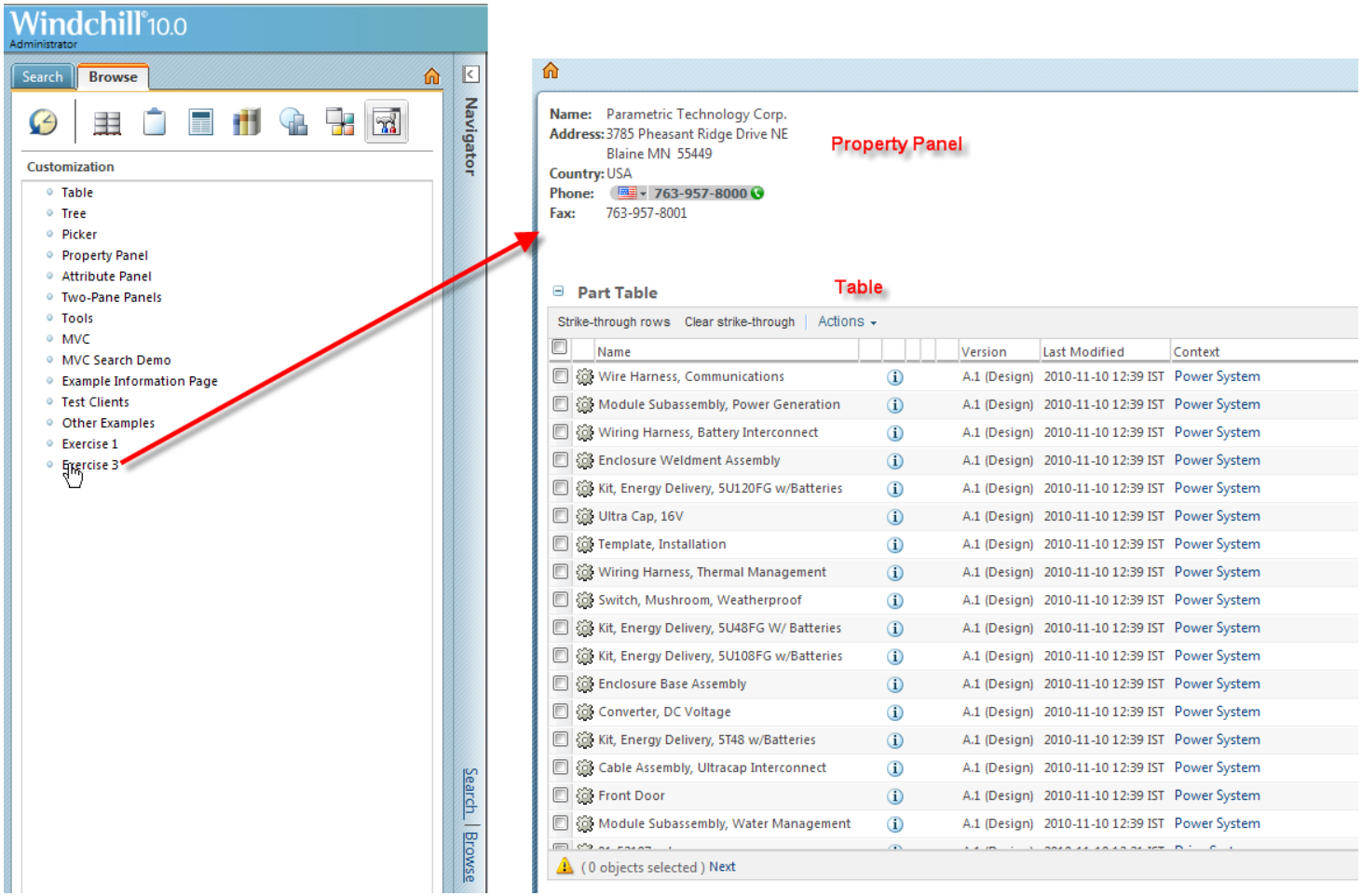
> Find the builder for the current Product List Table

> Override the OOTB Builder

- Base package com.ptc.mvc.builders
- Annotate it with @OverrideComponentBuilder
- Compile the class and restart the MethodServer

Lab session – Exercise 3

> Create a page with multiple MVC JCA Components (Property Panel & Table)



Exercise 3 : Builders

PTC

> Configure an action that generate a MVC url, that builds a table with componentId “exercise3”
  - Define the new action in WT_HOME/codebase/config/actions/custom-actions.xml
  - Find the definition of “Customization” action model
  - Override it in WT_HOME/codebase/config/actions/custom-actionModels.xml
  - Load the updated action configurations.

> Create the builder
  - Base package com.ptc.mvc.builders
  - Extend AbstractComponentBuilder
  - Annotate it with @ComponentBuilder
  - Provide the MultiComponentConfig
    - Add Property Panel
    - Add the Table created in exercise1
    - Set the custom view (/custom/exercise3.jsp)
  - Provide the Data for Property Panel
  - Compile the class and restart the MethodServer

Exercise 3 : Code Snippets

```
@Override
public ComponentConfig buildComponentConfig(ComponentParams params) throws WTException {

    ComponentConfigFactory factory = getComponentConfigFactory();
    PropertyPanelConfig propertyPanelConfig = factory.newPropertyPanelConfig();
    propertyPanelConfig.setId("exercise3PropertyPanel");

    // add properties
    PropertyConfig nameConfig = factory.newPropertyConfig();
    nameConfig.setId("name");
    propertyPanelConfig.addComponent(nameConfig);

    PropertyConfig address1Config = factory.newPropertyConfig();
    address1Config.setId("address1");
    address1Config.setLabel("Address");
    propertyPanelConfig.addComponent(address1Config);

    MultiComponentConfig mconfig = new MultiComponentConfig();
    // add the PropertyConfig
    mconfig.addComponent(propertyPanelConfig);
    // add a nested component
    mconfig.addNestedComponent("exercise1");

    // set the view
    mconfig.setView("/custom/exercise3.jsp");

    return mconfig;
}
```

Exercise 3 : Code Snippets

```
@Override
  public Object buildComponentData(ComponentConfig config, ComponentParams params) throws WTException {

      // data for the PropertyConfig
      Map<String, String> address = new HashMap<String, String>();
      address.put("name", "Parametric Technology Corp.");
      address.put("address1", "3785 Pheasant Ridge Drive NE");
      address.put("address2", "Blaine MN  55449");
      address.put("country", "USA");
      address.put("phone", "763-957-8000");
      address.put("fax", "763-957-8001");
      return address;
  }
```

Exercise 3 : view

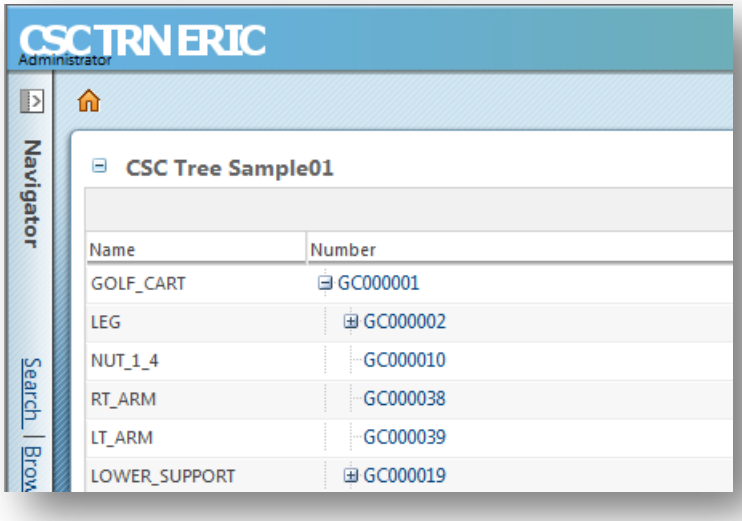
```
// content of lab/exercise3.jsp
 <%@ taglib uri="http://www.ptc.com/windchill/taglib/components" prefix="jca"%>
<%@ taglib uri="http://www.ptc.com/windchill/taglib/jcaMvc" prefix="mvc"%>

<%@include file="/netmarkets/jsp/util/begin_comp.jspf"%>

<%-- render the property panel --%>
<jca:renderPropertyPanel componentDef="${exercise3PropertyPanel}" />

<%-- register the listeners --%>
<script type="text/javascript">
   PTC.onReady(function() {
      Ext.ComponentMgr.onAvailable('exercise1', function(){

         //register on renderTableEnd
         this.on("renderTableEnd", function (){
             alert("Data after renderTableEnd : " + this.getStore().getCount());
         });

         //register on dataSourceComplete
          this.getStore().on("dataSourceComplete", function (){
            alert("Data after dataSourceComplete : " + this.getCount());
         });
      });
   });
</script>

<%-- render the table --%>
<mvc:table compId="exercise1"/>

<%@ include file="/netmarkets/jsp/util/end_comp.jspf"%>
```

Tree

Customization process

Tree customization needs two ways implementation.

- Create Tree Builder same like Table builder
- Table builder needs to implement some interface for tree synchronization
- Create Tree Handler – For making roots and child nodes
- Optional : Register custom MVC library (if you already set it does not need.)



Advanced implementation “AbstractComponentBuilder”

PTC

Component architecture.

Creating component builder is most similar of table component. Additionally, tree component needs Tree handler class. Following is explanation of the comparing and architecture for class.

Extend AbstractComponentBuiler
- Overriding buindComponentData
  - ✓ Get data set
- Overriding buindComponentConfig
  - ✓ UI design using JcaTableConfig

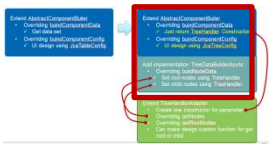
Extend AbstractComponentBuiler
- Overriding buindComponentData
  - ✓ Just return TreeHandler Construction
- Overriding buindComponentConfig
  - ✓ UI design using JcaTreeConfig

Add implementation TreeDataBuilderAsync
- Overriding buildNodeData
  - ✓ Set root nodes using TreeHandler
  - ✓ Set child nodes using TreeHandler

Extend TreeHandlerAdapter
- Create one constructor for parameter
- Overriding getNodes
- Overriding getRootNodes
- Can make design custom function for get root or child

1. Create MVC component for tree table

Build MVC Tree Table



1. Create MVC Class

Before creating MVC class, you must register MVC directory. If you do not want to register directory, you can directly register new MVC class.

```
1  package com.csc.mvc;                                                    (1)
2
3  import java.util.ArrayList;
22
23 @ComponentBuilder("csc.mvc.sampleTree01")                               (2)
24 public class CSCSampleTree01 extends AbstractComponentBuilder implements TreeDataBuilderAsync {   (3) (4)
25     CSCSampleTreeHandler01 sampleHandler;
26
27     @Override                                                           (5)
28     public Object buildComponentData(ComponentConfig arg0, ComponentParams arg1) throws Exception {
29         // TODO Auto-generated method stub
30         return new CSCSampleTreeHandler01();                           (6)
31     }
32
33     @Override
34     public ComponentConfig buildComponentConfig(ComponentParams arg0) throws WTException {   (7)
35         // TODO Auto-generated method stub
36         //Create TreeConfig
37         ComponentConfigFactory factory = getComponentConfigFactory();
38         JcaTreeConfig tree = (JcaTreeConfig) factory.newTreeConfig();
39
40         // Need to set DataSOurceModes explicitely to DataSourceMode.ASYNCHRONOUS
41         tree.setDataSourceMode(DataSourceMode.ASYNCHRONOUS);
```

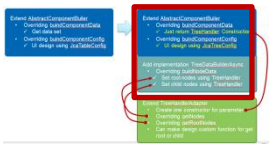
Using **JcaTreeConfig** for tree UI design

1. Set package which is registered in mvc.xml
2. Create annotation for MVC component ID
3. Set Java Class name which extended “AbstractComponentBuilder”
4. **Implement “TreeDataBuilderAsync” interface**
5. Override “buildComponentData” which function will be prgorammed to get data by inputted parameters.
6. **Just return TreeHandler constructor function**
7. Override “buildComponentConfig” fucinton which will work to design UI.

1. Create MVC component for tree table

Build MVC Tree Table

CSCSampleTree01.java



1. Create MVC Class

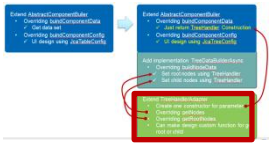
Before creating MVC class, you must register MVC directory. If you do not want to register directory, you can directly register new MVC class.

```
55      @Override                                                                    1
56      public void buildNodeData(Object node, ComponentResultProcessor resultProcessor) throws Exception {
57          if (node == TreeNode.RootNode) {// special case for root nodes
58              sampleHandler = new CSCSampleTreeHandler01(resultProcessor.getParams());   2
59              resultProcessor.addElements(sampleHandler.getRootNodes()); // should fetch rootNodes and to
60          } else {// case for any other node
61              List nodeList = new ArrayList();
62              nodeList.add(node);
63              Map<Object, List> map = sampleHandler.getNodes(nodeList);
64              Set keySet = map.keySet();
65              for (Object key : keySet) {
66                  resultProcessor.addElements(map.get(key));
67              }
68          }
69      }
```

1. **Overriding "buindNodeData"**
   → ComponentResultProcessor parameter can get opener's parameters.
2. **Constructor Custom tree handler**
   → This sample's constructor needs to opener parameters, so sample tree handler class has a constructor for initializing and getting parameters)
3. **Other source line is standard source code for get node data, so you just copy sample source codes**

1. Create MVC component for tree table

Build MVC Tree Table



2. Create Tree Handler

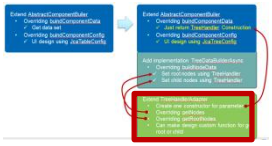
```
1  package com.csc.mvc;
2
3  import java.util.ArrayList;
27
28 public class CSCSampleTreeHandler01 extends TreeHandlerAdapter {   1
29     private NmCommandBean cb = null;
30     String number = null;
31     String name = null;
32     String partType = null;
33
34     public CSCSampleTreeHandler01() {   2
35
36     }
37
38     /****
39     public CSCSampleTreeHandler01(ComponentParams params) throws WTException {
40         cb = ((JcaComponentParams) params).getHelperBean().getNmCommandBean();
41     }
42     *****/
43
44     public CSCSampleTreeHandler01(ComponentParams params) throws WTException {   3
45         //String number = (String) params.getParameter("number");
46         //String name = (String) params.getParameter("name");
47         //String partType = (String) params.getParameter("partType");
48         number = "GC000001";
49     }
```

It is other sample when the parameter is “NmCommandBean”

1. Create custom TreeHandler extended from “TreeHandlerAdapter”
2. Create default constructor (Empty constructor)
3. Create custom constructor (Sample is for get parameter from opener)

1. Create MVC component for tree table

Build MVC Tree Table



2. Create Tree Handler

```
51    @Override
52    public Map<Object, List> getNodes(List parents) throws WTException {   (1)
53        // TODO Auto-generated method stub
54        WTPartStandardConfigSpec standardConfigSpec = WTPartStandardConfigSpec.newWTPartSta
55        WTPartConfigSpec configSpec = WTPartConfigSpec.newWTPartConfigSpec(standardConfigSp
56
57        Map<Object, List> result = new HashMap<Object,List>();
58
59        Persistable[][][] all_children = WTPartHelper.service.getUsesWTParts(new WTArrayLis
60
61        for ( ListIterator i = parents.listIterator(); i.hasNext(); ) {
62            WTPart parent = (WTPart)i.next();
63
64            Persistable[][] oneParentNode = all_children[i.previousIndex()];
65            if ( oneParentNode == null ) {
66                continue;
67            } else {
68                List children = new ArrayList(oneParentNode.length);
69                for( int j=0; j < oneParentNode.length; j++ ) {
70                    children.add(oneParentNode[j][1]);
71                }
72                result.put(parent, children);
73            }
74        }
75        return result;
76    }
77
78    @Override
79    public List<Object> getRootNodes() throws WTException {   (2)
80        // TODO Auto-generated method stub
81        List<Object> resultList = new ArrayList<Object>();
82
83        if ( number != null && number.length() > 0 )  {
84            number = number.trim();
85        } else {
86            return resultList;
```

1. Overriding for get child nodes
2. Overriding for get root nodes

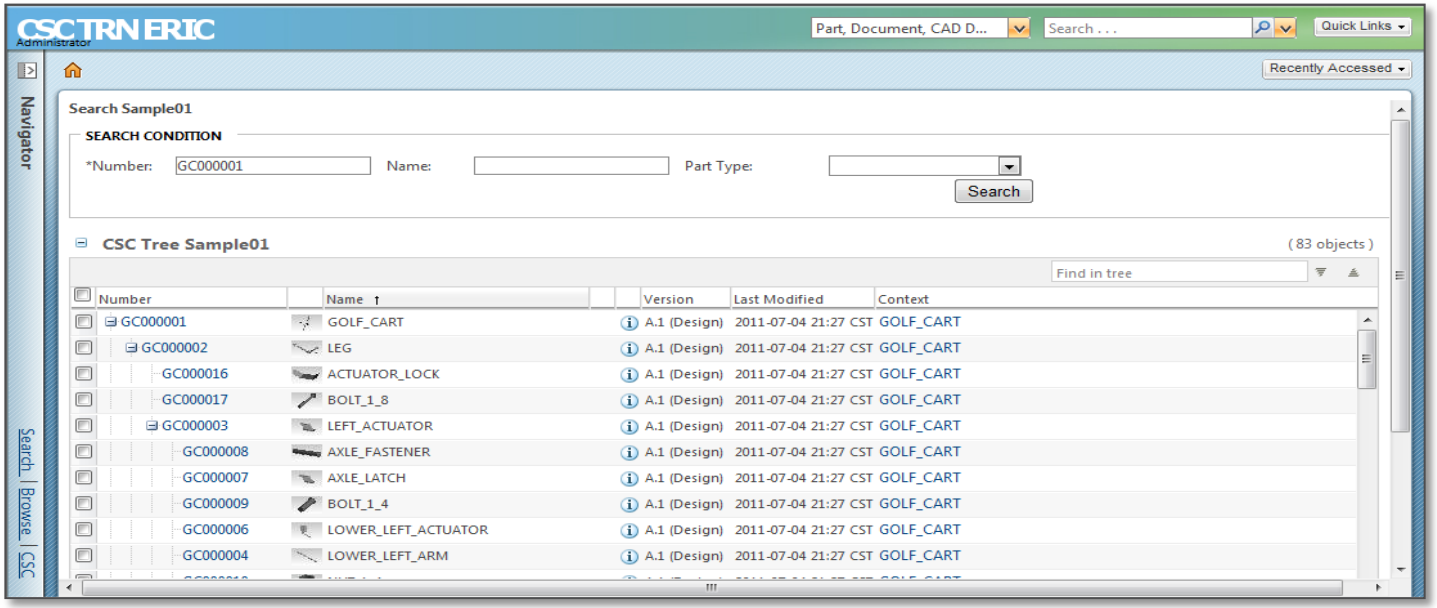
CSCSampleTree Handler01.java

2. Exercise: Search and Tree result

Scenario for search and Tree Table

Before we understand how to make search and tree table, we must have one scenario like following.

- Search target will be all WTPart object.
- Search criteria needs to be number, name and Part Type.
- Number field must be required field
- Part Type must be designed ComboBox
- Result tree table must be show like Part list.
  - Number, Name, Small thumbnail, etc..



2. Create search and table

Build JSP for criteria

1. Create JSP and set basic environment

sampleSearch01_tree.jsp

```
<%@ taglib prefix="jca" uri="http://www.ptc.com/windchill/taglib/components"%>
<%@ include file="/netmarkets/jsp/util/begin.jspf"%>
<%@ taglib prefix="mvc" uri="http://www.ptc.com/windchill/taglib/jcaMvc"%>
<%@ taglib prefix="wrap" uri="http://www.ptc.com/windchill/taglib/wrappers"%>
<%@ taglib prefix="csc" uri="http://www.ptc.com/windchill/taglib/csc"%>
<%@ taglib tagdir="/WEB-INF/tags" prefix="wctags"%>

<%@ include file="/netmarkets/jsp/util/end.jspf"%>
```

2. Add populator tag for get criteria data set

sampleSearch01_tree.jsp

```
<%@ taglib prefix="jca" uri="http://www.ptc.com/windchill/taglib/components"%>
<%@ include file="/netmarkets/jsp/util/begin.jspf"%>
<%@ taglib prefix="mvc" uri="http://www.ptc.com/windchill/taglib/jcaMvc"%>
<%@ taglib prefix="wrap" uri="http://www.ptc.com/windchill/taglib/wrappers"%>
<%@ taglib prefix="csc" uri="http://www.ptc.com/windchill/taglib/csc"%>
<%@ taglib tagdir="/WEB-INF/tags" prefix="wctags"%>

<csc:searchPopulator name="com.csc.search.SampleSearchDataPopulator01" />

<%@ include file="/netmarkets/jsp/util/end.jspf"%>
```

2. Create search and table

Build JSP for criteria

3. Create populator class for sample criteria

SampleSearchDataPopulator01.class

```
package com.csc.search;

import java.util.ArrayList;
import java.util.List;
import javax.servlet.jsp.JspContext;

import com.csc.common.tags.SearchDataPopulator;

public class SampleSearchDataPopulator01 implements SearchDataPopulator {
    public static final String VERSION = "$Id: $";

    @Override
    public void populateSearchCriteria(JspContext jspContext) {
        // TODO Auto-generated method stub
        try {
            // Set all of key in here
            //Sample01 - Part Type Population
            List<String> partTypeKey = new ArrayList<String>();
            List<String> partTypeDisplay = new ArrayList<String>();
            partTypeKey.add("");
            partTypeDisplay.add("");

            PartType[] partSet = PartType.getPartTypeSet();

            for( int i=0; i < partSet.length; i++) {
                partTypeKey.add(partSet[i].getStringValue().substring(partSet[i].getStringValue().lastIndexOf(".")+1));
                partTypeDisplay.add(partSet[i].getDisplay());
            }

            jspContext.setAttribute("partType_internal_vals", partTypeKey);
            jspContext.setAttribute("partType_display_vals", partTypeDisplay);
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

2. Create search and table

Build JSP for criteria

4. Build criteria UI in JSP page

Add following source in “sampleSearch01_tree.jsp”


```
<b>Search Sample01</b>
<fieldset class="x-fieldset x-form-label-left" id="Visualization_and_Attributes" style="width: 1024px;">
    <legend>SEARCH CONDITION</legend>
        <table>
            <tr>
                <td scope="row" width="50" class="tableColumnHeaderfont" align="left">*Number:</td>
                <td class="tabledatafont" align="left"> 
                    <wrap:textBox name="number" id="number" maxlength="30" size="10"
                            nblur="this.value=this.value.toUpperCase()" />
                </td>

                <td scope="row" width="50" class="tableColumnHeaderfont" align="left">Name:</td>
                <td class="tabledatafont" align="left"> 
                    <wrap:textBox name="name" id="name" maxlength="100" size="20"/>
                </td>

                <td scope="row" width="100" class="tableColumnHeaderfont" >Part Type:</td>
                <td class="tabledatafont" align="left">
                    <wrap:comboBox id="partType" name="partType" multiSelect="false" size="1"
                        displayValues="${partType_display_vals}"
                        internalValues="${partType_internal_vals}" />
                </td>
            </tr>

            <tr>
                <td colspan="6" scope="row" class="tableColumnHeaderfont" align="right">
                    <wrap:button name="search” value='Search’ onclick="submitDocSearch();" />
                </td>
            </tr>
        </table>
</fieldset>
```

2. Create search and table

Build JSP for criteria

searchSample01_tree.jsp

5. Add MVC tree table and javascript function in JSP page

Add following source in “sampleSearch01_tree.jsp”

```
<mvc:tableContainer compId="csc.mvc.sampleTree01" height="500" />

<%@ include file="/netmarkets/jsp/util/end.jspf"%>

<script>

            function submitDocSearch(){
                        if (document.getElementById('number').value==""){
                                    JCAAlert("Number must be inputted");
                        } else {
                                    submitParameters();
                        }
            }

            function submitParameters() {
                        var number = document.getElementById('number').value;
                        var name = document.getElementById('name').value;
                        var partType = document.getElementById('partType').value;

                        var params = {
                                                number : number,
                                                name : name,
                                                partType : partType

                        };

                        PTC.jca.table.Utils.reload('csc.mvc.sampleTree01', params, true);
            }

</script>
```

Call MVC tree table

Recall MVC tree table using parameters

PTC

Build MVC Tree Table

5. Create MVC Class

Before creating MVC class, you must register MVC directory. If you do not want to register directory, you can directly register new MVC class.

```
1  package com.csc.mvc;                                                          1
2
3  import static com.ptc.core.components.descriptor.DescriptorConstants.ColumnIdentifiers.CONTAINER_NAM
32
33 @ComponentBuilder("csc.mvc.sampleTree01")                                     2
34 public class CSCSampleTree01 extends AbstractComponentBuilder implements TreeDataBuilderAsync {
35     CSCSampleTreeHandler01 sampleHandler;                                     3   4
36
37     @Override
38     public Object buildComponentData(ComponentConfig arg0, ComponentParams arg1) throws Exception {
39         // TODO Auto-generated method stub                                    5
40         return new CSCSampleTreeHandler01();                                  6
41     }
42
43     @Override                                                                 7
44     public ComponentConfig buildComponentConfig(ComponentParams arg0) throws WTException {
45         // TODO Auto-generated method stub
46         //Create TreeConfig
47         ComponentConfigFactory factory = getComponentConfigFactory();
48         JcaTreeConfig tree = (JcaTreeConfig) factory.newTreeConfig();
49
```

Using **JcaTreeConfig** for tree UI design

1. Set package which is registered in mvc.xml
2. Create annotation for MVC component ID
3. Set Java Class and extend “AbstractComponentBuilder”
4. Implement “TreeDataBuilderAsync” interface
5. Override “buildComponentData” which function will be prgorammed to get data by inputted parameters.
6. Set return for TreeHandler constructor
7. Override “buildComponentConfig” fucinton which will work to design UI.

PTC

Build MVC Tree Table

5. Create MVC Class

Before creating MVC class, you must register MVC directory. If you do not want to register directory, you can directly register new MVC class.

```
33 @ComponentBuilder("csc.mvc.sampleTree01")
34 public class CSCSampleTree01 extends AbstractComponentBuilder implements TreeDataBuilderAsync {
35     CSCSampleTreeHandler01 sampleHandler;
36
38     public Object buildComponentData(ComponentConfig arg0, ComponentParams arg1) throws Exception {
42
44     public ComponentConfig buildComponentConfig(ComponentParams arg0) throws WTException {
83
84     @Override
85     public void buildNodeData(Object node, ComponentResultProcessor resultProcessor) throws Exception {
86         if (node == TreeNode.RootNode) {// special case for root nodes
87             sampleHandler = new CSCSampleTreeHandler01(resultProcessor.getParams());
88             resultProcessor.addElements(sampleHandler.getRootNodes()); // should fetch rootNodes and to
89         } else {// case for any other node
90             List nodeList = new ArrayList();
91             nodeList.add(node);
92             Map<Object, List> map = sampleHandler.getNodes(nodeList);
93             Set keySet = map.keySet();
94             for (Object key : keySet) {
95                 resultProcessor.addElements(map.get(key));
96             }
97         }
98     }
```

1

CSCSampleTre
e01.java

1. Overriding “buildNodeData” function.

→You just copy sample source code because the source code block is the most general implementation for tree.

→For getting Root and Child, call to TreeHandler.

2. TreeHandler is custom handler, so you have to create your custom handler.

PTC

Build MVC Tree Table

6. Create Tree Handler Class

Handler class is custom class for get Root Node and Child nodes.

```
28 public class CSCSampleTreeHandler01 extends TreeHandlerAdapter {
29     private NmCommandBean cb = null;
30     String number = null;
31     String name = null;
32     String partType = null;
33
34     public CSCSampleTreeHandler01() {
35
36     }
37
39     public CSCSampleTreeHandler01(ComponentParams params) throws WTException {
43
44     public CSCSampleTreeHandler01(ComponentParams params) throws WTException
45         number = (String) params.getParameter("number");
46         name = (String) params.getParameter("name");
47         partType = (String) params.getParameter("partType");
48         //number = "GC000001";
49     }
50
52     public Map<Object, List> getNodes(List parents) throws WTException {
77
78     @Override
79     public List<Object> getRootNodes() throws WTException {
80         // TODO Auto-generated method stub
81         List<Object> resultList = new ArrayList<Object>();
```

1. **Create custom TreeHandler.**

   →extended from “TreeHandlerAdapter”

   →If you have parameter, you set global parameter for keeping parameter values.

2. **Create Default constructor**
3. **Create custom constructor**

   →If you have parameters, use this constructor

4. **Overriding “getNodes” function**

   →Find child nodes using “parents” input

   →return must be sent Map<parent, children>

5. **Overring “getRootNodes” function**

   →Find root nodes

CSCSampleTreeHandler01.java

3. Detail of JcaTreeConfig

> Will be update soon