Migration From WINC

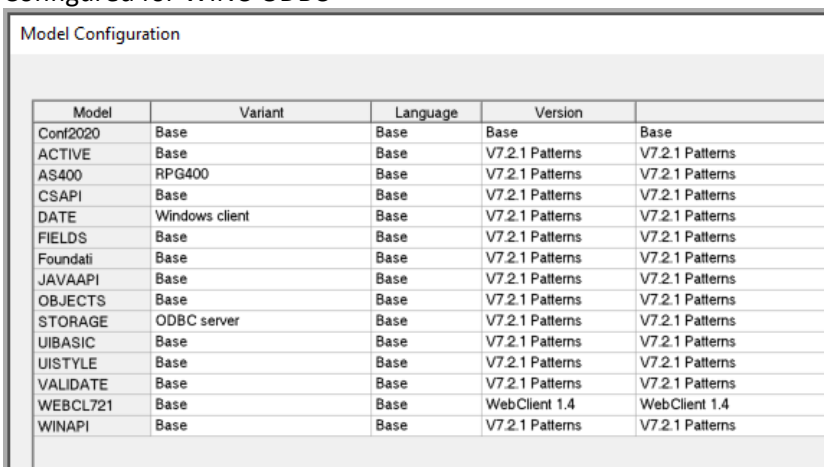
Exercise Goal

Convert WINC-ODBC application to .NET or Java

About Exercise Environment

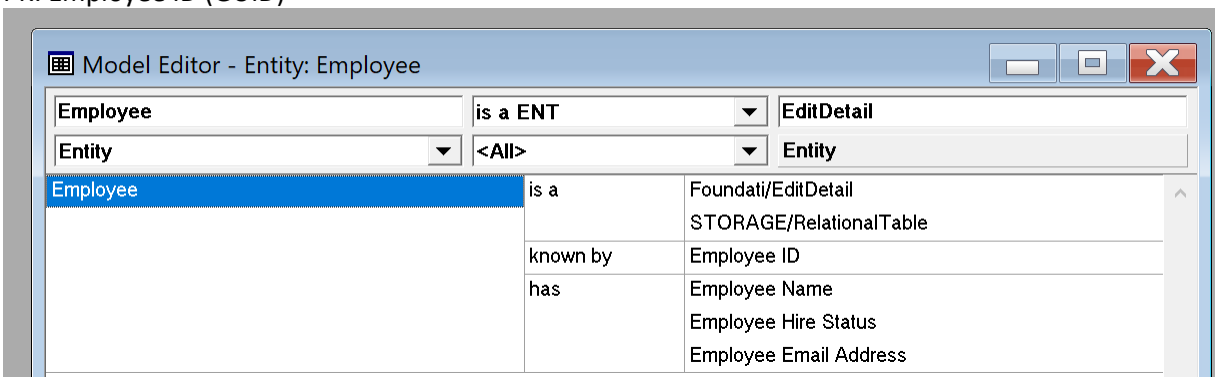
Local Model

C:\Conf2020\Exercise\Local\Conf2020.mdl
Configured for WINC-ODBC



Model	Variant	Language	Version	
Conf2020	Base	Base	Base	Base
ACTIVE	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
AS400	RPG400	Base	V7.2.1 Patterns	V7.2.1 Patterns
CSAPI	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
DATE	Windows client	Base	V7.2.1 Patterns	V7.2.1 Patterns
FIELDS	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
Foundati	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
JAVAAPI	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
OBJECTS	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
STORAGE	ODBC server	Base	V7.2.1 Patterns	V7.2.1 Patterns
UIBASIC	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
UISTYLE	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
VALIDATE	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
WEBCL721	Base	Base	WebClient 1.4	WebClient 1.4
WINAPI	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns

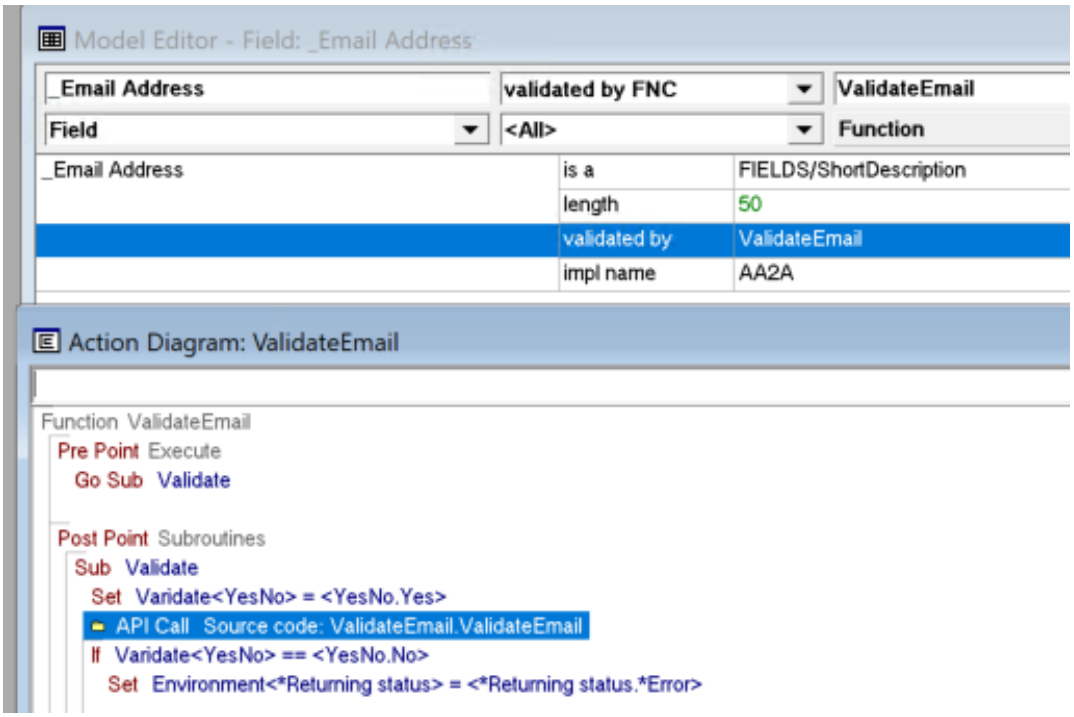
Single Entity Employee
PK: Employee ID (GUID)



Email Validation

The Employee Entity has a Email Address field and using Source code API for Email format validation
The validation is done by checking char by char

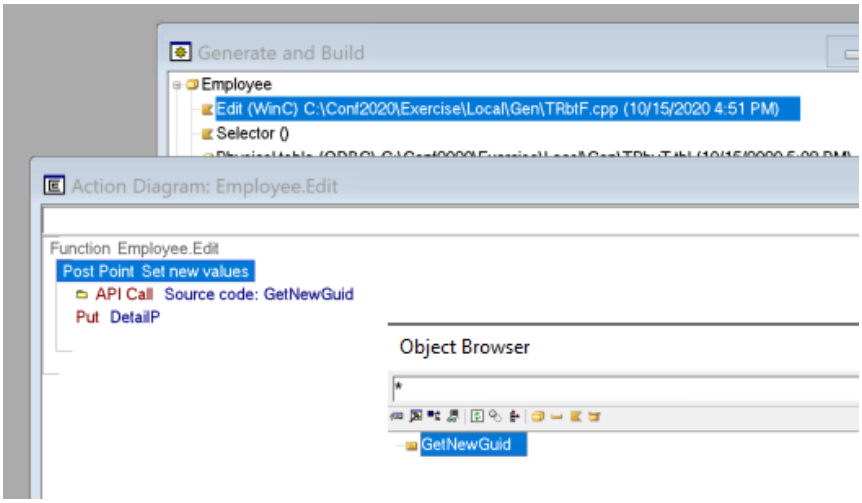
```
{
	CString s = &(1:);
	&(2:) = "Y";

	if (! s.IsEmpty() )
	{
		int i,offset1,offset2,n, hitAtMark = 0;
		bool charflag=true,offsetflag=false;

		n=strlen(s);

		for(i=0;i<n;i++)
		{
			char c = s[i];
			if (!(c >='A' && c <= 'Z' || c >= 'a' && c <= 'z'|| c >= '0' && c <='9' ||c =='.'||c =='-' || c =='+'|| c =='@'|| c =='_'))
			{
				charflag=false;
				break;
			}
		}
		offset1 = -1;

		offset2 = -1;

		for(i=0;i<n;i++)
		{
			if(s[i]=='@')
			{
				hitAtMark++;
				offset1=i;
			}
			if(s[i]=='.')
			{
				offset2=i;
			}
		}

		// '@' mark is not first char and there is a '.' after '@'
		if(offset1 > 0 && offset2 >= 0 && offset1 < offset2)
		{
			offsetflag=true;
		}
		// '@' apear once, all char is valid
		if(hitAtMark == 1 && offsetflag==true && charflag==true)
		{
			// Valid Email
		}
		else
		{
			&(2:) = "N";
		}
	}
	else
	{
		&(2:) = "N";
	}
}
```



GUID generation

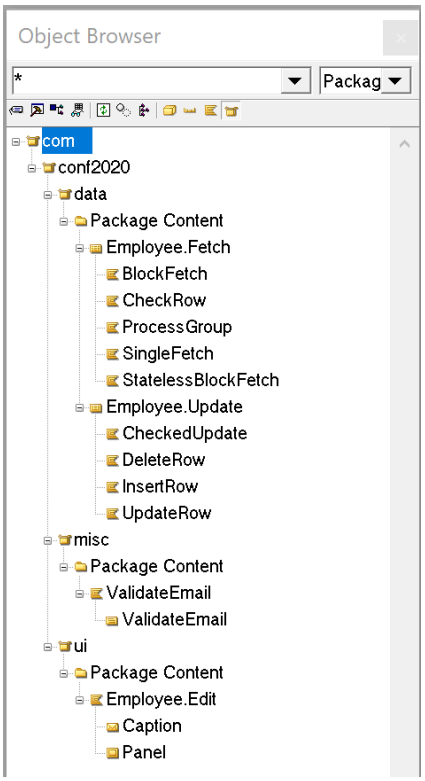
The Employee Entity using GUID as a key and using Source code API to generate New GUID

```
{
        GUID gidReference;
        CoCreateGuid( &gidReference );

        OLECHAR* guidString;
        StringFromCLSID(gidReference, &guidString);
        &(1:)=guidString;
}
```



Package Structure

Please confirm Packages, they are predefined for you
com.conf2020.data
com.conf2020.misc
com.conf2020.ui



Exercise Path

Please Pick .NET or Java (or both)

Both exercise 1 (.NET) and 2 (Java) goal is to convert WinC application to new target platform

Java exercise has extra one to convert Java Client to WebClient (Exercise 3)

Exercise 1 Convert to .NET

1. Change Model Configuration for .NET

First to get the power of CA Plex Model configuration, please change Local model Configuration for .NET

Model Configuration

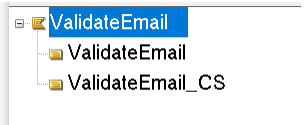
Model	Variant	Language	Version	
Conf2020	Base	Base	Base	Base
ACTIVE	.NET	Base	V7.2.1 Patterns	V7.2.1 Patterns
AS400	RPG400	Base	V7.2.1 Patterns	V7.2.1 Patterns
CSAPI	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
DATE	.NET server	Base	V7.2.1 Patterns	V7.2.1 Patterns
FIELDS	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
Foundati	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
JAVAAPI	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
OBJECTS	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
STORAGE	.NET server	Base	V7.2.1 Patterns	V7.2.1 Patterns
UIBASIC	.NET	Base	V7.2.1 Patterns	V7.2.1 Patterns
UISTYLE	.NET	Base	V7.2.1 Patterns	V7.2.1 Patterns
VALIDATE	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
WEBCL721	Base	Base	WebClient 1.4	WebClient 1.4
WINAPI	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns

2. Email Validation Conversion

Function: Employee.Edit has an employee e-mail field on the panel. It is validated by “Function: ValidateEmail” using “Fld Validated by Fnc” which contains C++ source code do the actual validation in C++. You need to convert it to C#

“Function: ValidateEmail” using C++ Source code to validate email
Please add Meta language Condition check, add new C# Source code and write C# code for email validation

- Add new Source code for C# ‘ValidateEmail_CS’
- Define the same parameter with ‘ValidateEmail’



- Write C# code to validate email
  Google search ‘C# email validation’ one of the answers is end of document

- Add Meta Function call statement in the Initialization

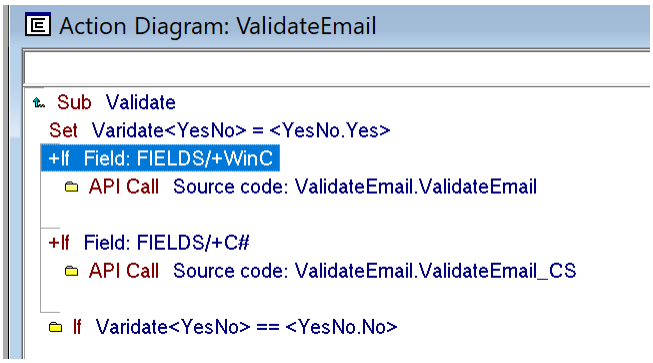
```
Call  UIBASIC/Meta.Options
```

- Add Meta Language Condition in Validate Sub routine

Add +If statement and API Call Statement

For Copy paste to AD

```
+If  Field: FIELDS/+WinC
+If  Field: FIELDS/+C#
```

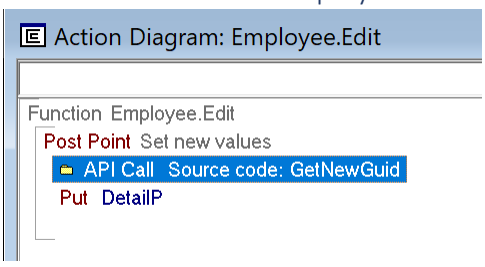


Note:
This is an example of conversion doesn't need to be complicated as C++ and it is within Framework already. The same applies many Data and Time calculations.

3. GUID generation (Source code: GetNewGuid)

Employee Entity has Employee id as key and using GUID instead of surrogate to keep the record unique.
The Function: Employee.Edit invoking GetNewGUID source code to generate Guid
You need to convert it to C#

- Function: Employee.Edit is calling the source code



Two options

1. The same approach with previous Email Validation exercise. Call Source code from Employee.Edit Function by adding Meta block
2. Promote 'GetNewGuid as Function' by wrapping Source code API and make it reusable

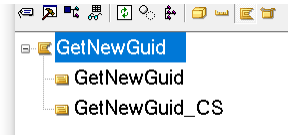
Opt. 1: Keep Source code

- Add one new Source code the ‘Source code: GetNewGuid_CS’ that has the same Parameter with ‘Source code: GetNewGuid’
- Write C# source code to generate New Guid
  The answer is at the end of this document
- Please add ‘Meta Function call’ like previous exercise and add +IF condition to invoke Source Codes

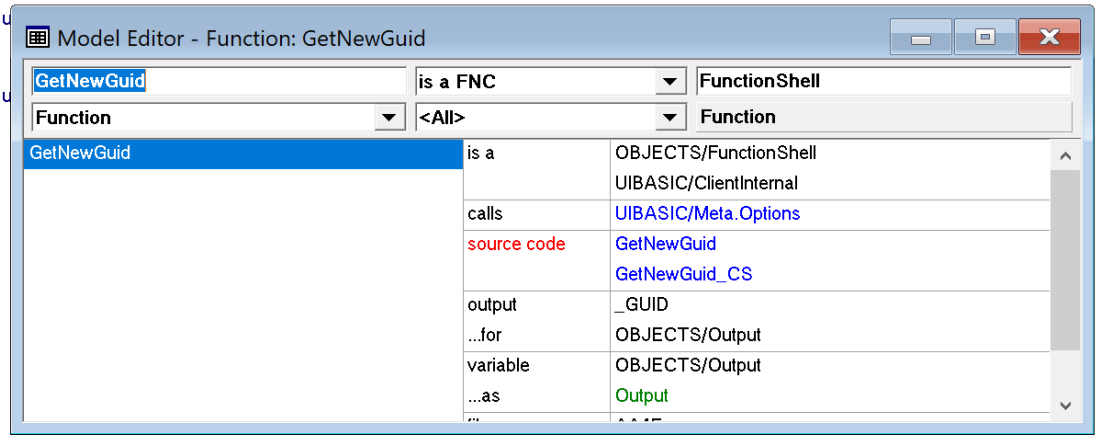
```
+If  Field: FIELDS/+WinC
  API Call  Source code: GetNewGuid.GetNewGuid

+If  Field: FIELDS/+C#
  API Call  Source code: GetNewGuid.GetNewGuid_CS
```

Opt. 2: Promote to Function

- Add one new Source code the ‘Source code: GetNewGuid_CS’ that has the same Parameter with ‘Source code: GetNewGuid’
- Write C# source code to generate New Guid
  The answer is at the end of this document



Add New Function GetNewGuid as bellow

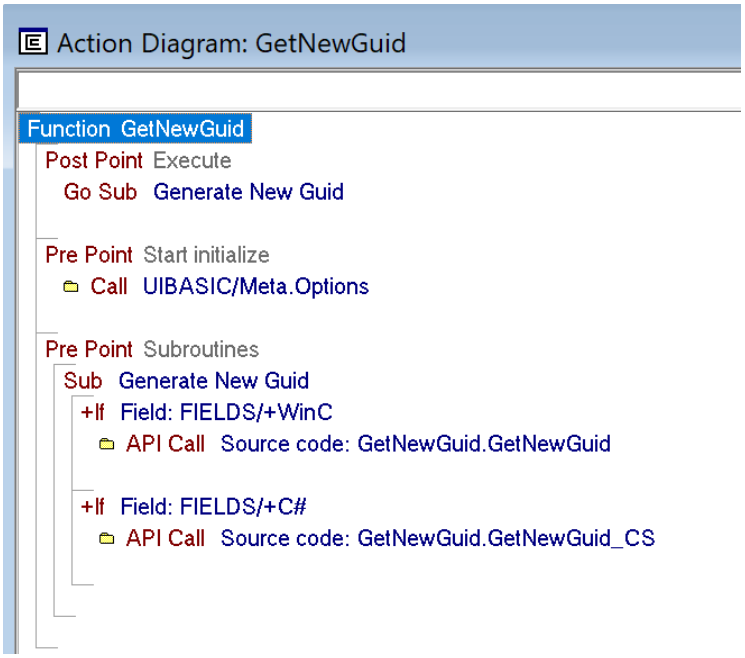


Please add the new Function: GetNewGuid to Package com.conf2020.misc

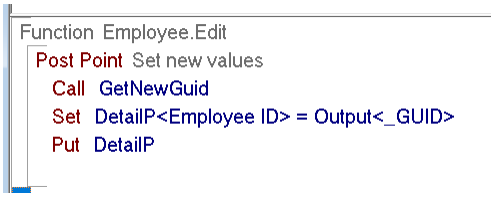
For Copy and Paste

```
FunctionShell
ClientInternal
GetNewGuid
GetNewGuid_CS
```

Add Action Diagram codes



Change Employee.Edit Action Diagram to call the new Function instead of Source code API Call



Note:

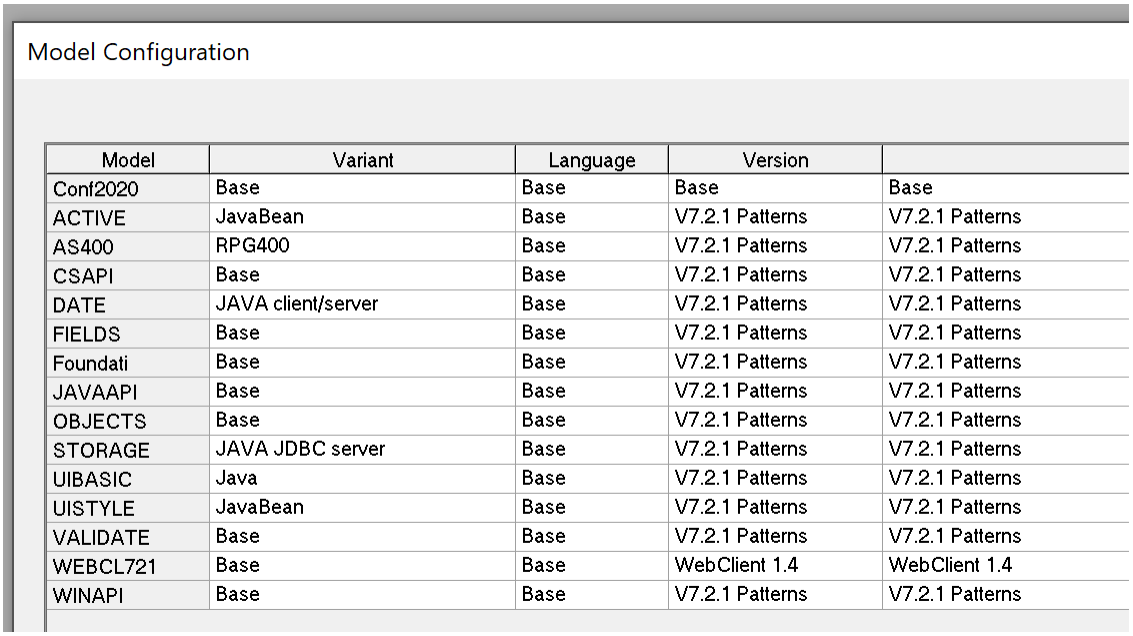
Option2 feels requiring more step. However, if the GetNewGUID is used many functions, Option 2 requires less step to migrate than Option 1.

Exercise 2 Convert to Java

1. Change Model Configuration for Java

First to get the power of CA Plex Model configuration, please change Local model Configuration for Java

Model Configuration



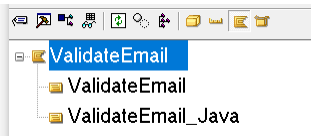
Model	Variant	Language	Version	
Conf2020	Base	Base	Base	Base
ACTIVE	JavaBean	Base	V7.2.1 Patterns	V7.2.1 Patterns
AS400	RPG400	Base	V7.2.1 Patterns	V7.2.1 Patterns
CSAPI	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
DATE	JAVA client/server	Base	V7.2.1 Patterns	V7.2.1 Patterns
FIELDS	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
Foundati	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
JAVAAPI	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
OBJECTS	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
STORAGE	JAVA JDBC server	Base	V7.2.1 Patterns	V7.2.1 Patterns
UIBASIC	Java	Base	V7.2.1 Patterns	V7.2.1 Patterns
UISTYLE	JavaBean	Base	V7.2.1 Patterns	V7.2.1 Patterns
VALIDATE	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns
WEBCL721	Base	Base	WebClient 1.4	WebClient 1.4
WINAPI	Base	Base	V7.2.1 Patterns	V7.2.1 Patterns

2. Email Validation Conversion (Function: ValidateEmail)

Function: Employee.Edit has an employee e-mail field on the panel. It is validated by “Function: ValidateEmail” which contains C++ source code do the actual validation
You need to convert it to Java

“Function: ValidateEmail” using C++ Source code to validate email
Please add Meta language Condition check, add new Java Source code and write Java code for email validation

- Add new Source code for Java ‘ValidateEmail_Java’
- Define the same parameter with ‘ValidateEmail’



- Write Java code to validate email
  Google search ‘C# email validation’ one of the answers is end of document.

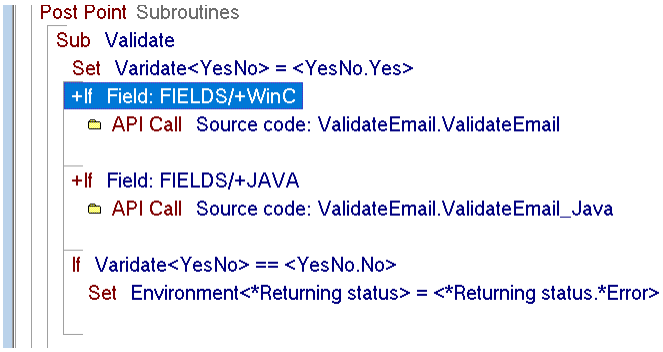
- Add Meta Function call statement in the Initialization

```
Call  UIBASIC/Meta.Options
```

- Add Meta Language Condition in Validate Sub routine
  Add +If statement and API Call Statement

For Copy paste to AD

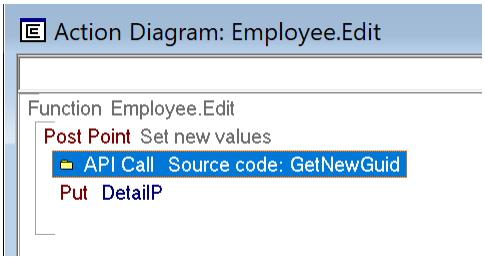
```
+If  Field: FIELDS/+WinC
+If  Field: FIELDS/+Java
```



3. GUID generation (Source code: GetNewGuid)

Employee Entity has Employee id as key and using GUID instead of surrogate to keep the record unique. The Function: Employee.Edit invoking GetNewGUID source code to generate Guid
You need to convert it to Java

- Function: Employee.Edit is calling the source code



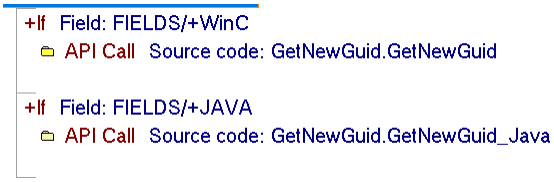
Two options

1. The same approach with previous Email Validation exercise. Call Source code from Employee.Edit Function by adding Meta block
2. Promote 'GetNewGuid as Function' by wrapping Source code API and make it reusable

Opt. 1: Keep Source code

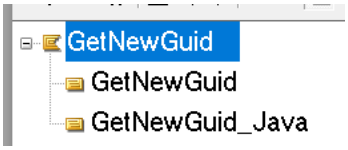
- Add one new Source code the 'Source code: GetNewGuid_Java' that has the same Parameter with 'Source code: GetNewGuid'
- Write Java source code to generate New Guid
  The answer is at the end of this document

- Please add ‘Meta Function call’ like previous exercise and add +IF condition to invoke Source Codes

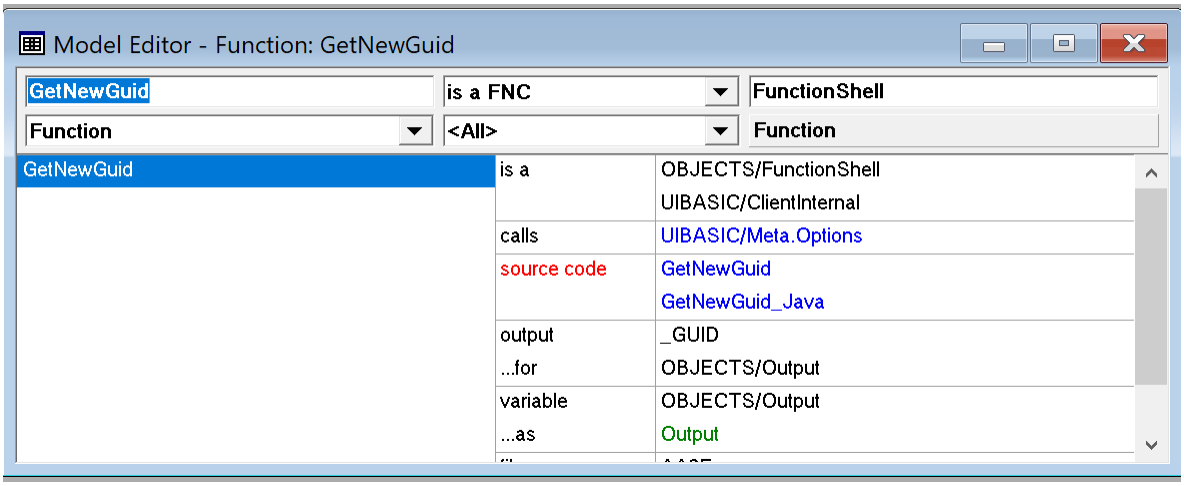


Opt. 2: Promote to Function

- Add one new Source code the ‘Source code: GetNewGuid_CS’ that has the same Parameter with ‘Source code: GetNewGuid’
- Write C# source code to generate New Guid
  The answer is at the end of this document



Add New Function GetNewGuid as bellow

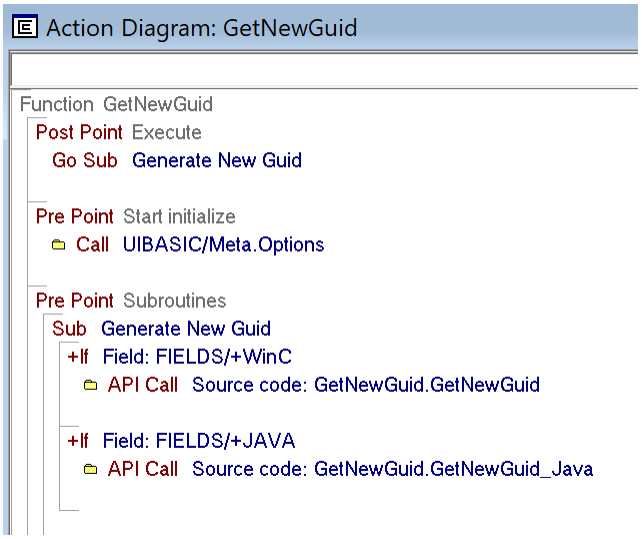


Please add the new Function: GetNewGuid to Package com.conf2020.misc

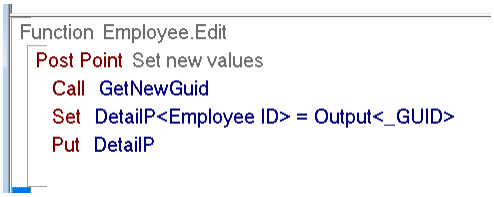
For Copy and Paste

```
FunctionShell
ClientInternal
GetNewGuid
GetNewGuid_Java
```

Add Action Diagram codes



Change Employee.Edit Action Diagram to call the new Function instead of Source code API Call



Note:

Option2 feels requiring more step. However, if the GetNewGUID is used many functions, Option 2 requires less step to migrate than Option 1.

Exercise 3 Convert Java to WebClient

CM WebClient provides the ability to run your Plex functions in a web browser. The web application can behave and perform like a desktop application, plus it can be expanded to take advantage of additional web controls. A great advantage of WebClient is that the web application only needs to be deployed to a web server and it is instantly available for users to access from anywhere in the world.

Before a Plex function can be run with WebClient, it needs to satisfy 3 criteria:

1. The function needs to be able to be generated and built in Java.
2. The function needs to inherit from a function with the name of one of the WebClient root templates, e.g. WebShell.
3. The function needs to include a special source code that the WebClient template generator uses to identify the inheritance path.

1. Prepare the function for WebClient

We have already prepared the application to run on the Java platform, so we have the first criteria satisfied. Let's work on the 2nd & 3rd criteria.

Add the following triple:

Employee.Edit *FNC is a FNC* *~WebShell*

~WebShell is a function in the WebClient group model which has an implementation name of "WebShell", which matches one of the root templates for WebClient. We'll discuss templates further down.

Open up the action diagram for your Employee.Edit function, and open the Post Point Description collection point. Here you can see the source code objects inherited from ~WebShell that informs WebClient of the function's inheritance path. Again, we'll discuss this later.

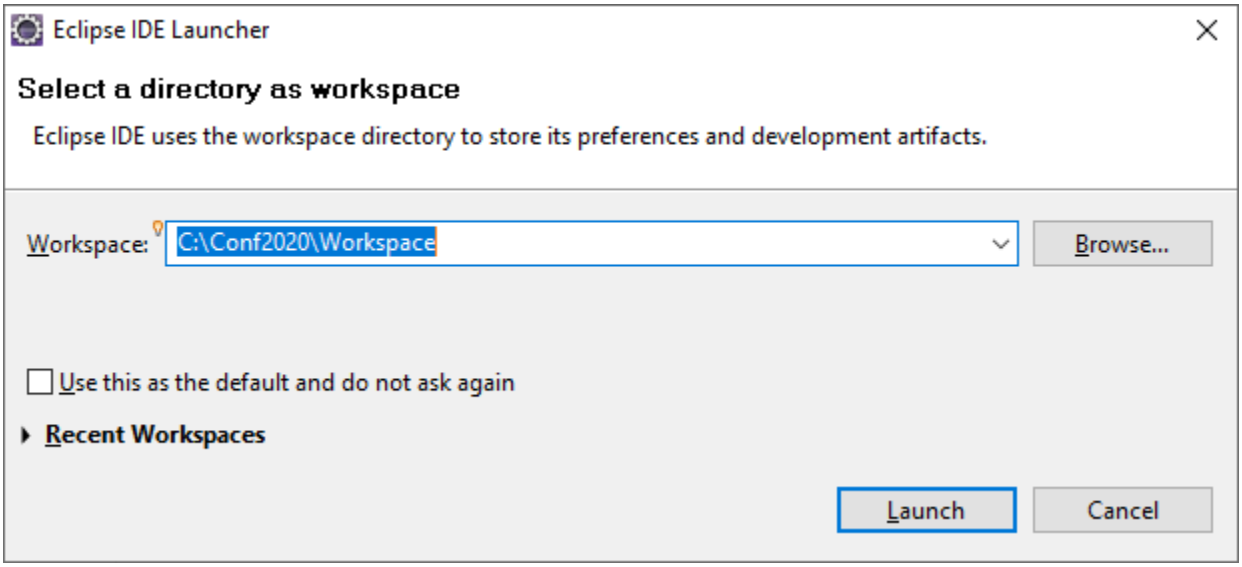
Your function has now been prepared to work with WebClient. There are a few more steps needed to deploy the function to the web.

2. Configure your WebClient application in Eclipse.

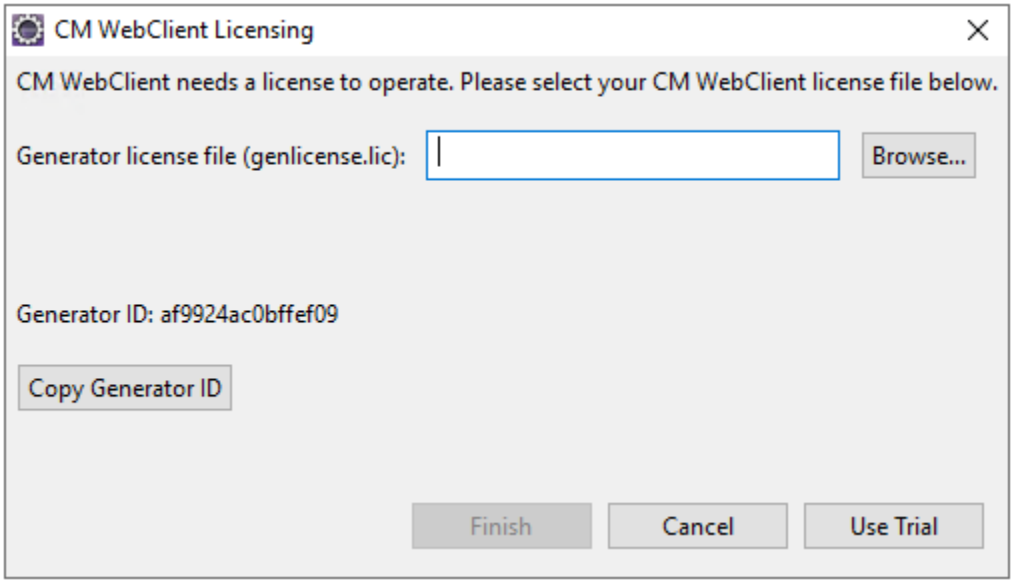
We use the Eclipse IDE to build, publish and debug WebClient applications.

Start Eclipse by clicking on the desktop icon.

Make sure the Workspace is set to C:\Conf2020\Workspace, and click on “Launch”.



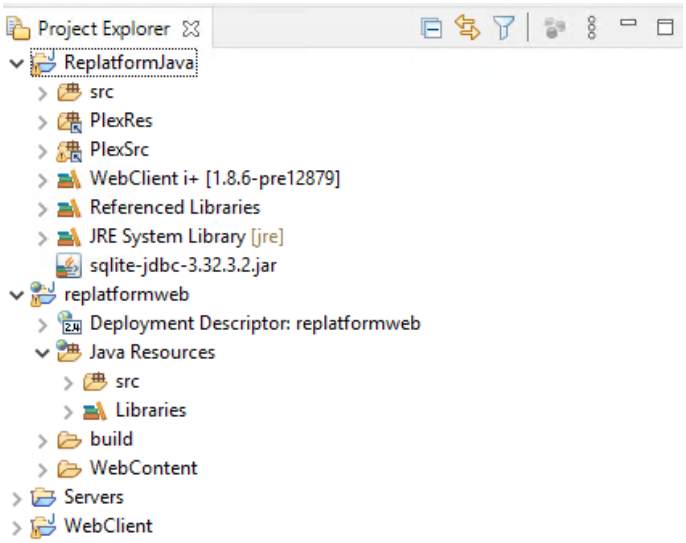
Once the workspace has loaded, you may see a WebClient License dialog. Click the “Use Trial” button.



On the left is the Project explorer. There are 3 main projects

- ReplatformJava – The Java project where the Plex-generated Java is published to.

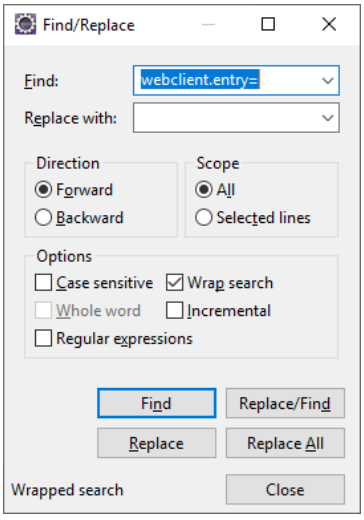
- replatformweb – The web project that contains everything to be published to the web server.
- WebClient – The WebClient product, containing templates, the template generator, and runtime files required to run the Plex application.



To run our function in the browser, we first need to tell WebClient which function it needs to run first. We can do this by configuring the WebClient property file.

Expand the *replatformweb* project, then expand *Java Resources* and *src*. You will see the WebClient.properties file. Double-click on this file to open it up in a text editor.

The WebClient.properties file has all the settings needed to run the application. The property *webclient.entry* is required to tell WebClient the entry point for your application. Press Ctrl+F to bring up the Find dialog, and enter “webclient.entry=” in the search field and press “Find”.



The property is there but it has been commented out (the # at the start of the line is a comment).

```
137
138 #webclient.entry=MyPackage.MyFunction
139
140 #webclient.entry.timeout=MyPackage.MyTimeout
141
142
```

Remove the # and set the value to “com.conf2020.ui.TRbtF”. With Java functions, we need to include the package name as well as the function’s implementation name.

```
137
138 webclient.entry=com.conf2020.ui.TRbtF
139
140 #webclient.entry.timeout=MyPackage.MyTimeout
141
```

Press Ctrl+S to save the properties file.

3. Generate and Build the function

Switch back to Plex, drag the Employee.Edit function to the Gen & Build window. Generate the function – there is no need to build here as we will build the function in Eclipse.

Go back to Eclipse. Select the ReplatformJava project, right-click and select ‘Refresh’. This will let Eclipse find the latest files you generated and apply the build process. Eclipse has been configured already to use the WebClient template builder.

You should see the WebClient template builder running in the Console tab near the bottom of the screen.

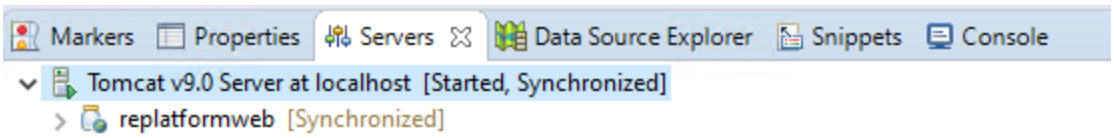
```
>>> Starting WebClient Build for ReplatformJava

> Using CM WebClient v1.8.6-pre12879
> Licensed to CMFirst Technologies
> Valid until Wed Dec 30 00:00:00 CST 2020

Generating Web templates.
....

>>> WebClient Build complete.
    1 panels processed. 0 panels not processed.
```

Your function is now ready to publish.

4. Publish the Web Application

There is “Servers” tab near the bottom of the screen with an entry for Tomcat. This is the server we will use to publish the web application to.

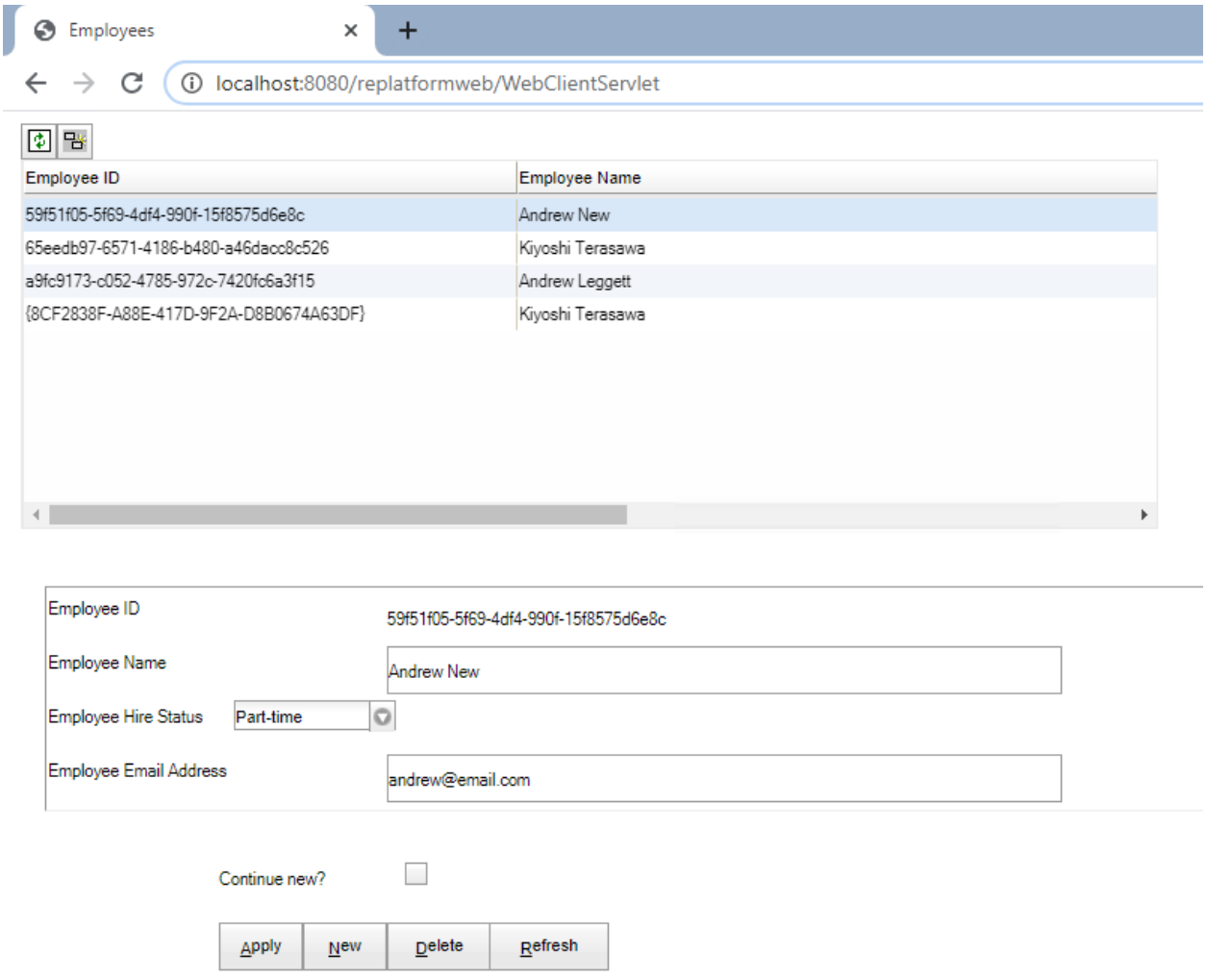
Right-click on the Tomcat server and select “Start”. After a few seconds, the status of the server should change to *[Started, Synchronized]*.



Your web application is ready to run.

Go to your desktop and click on the Google Chrome icon.

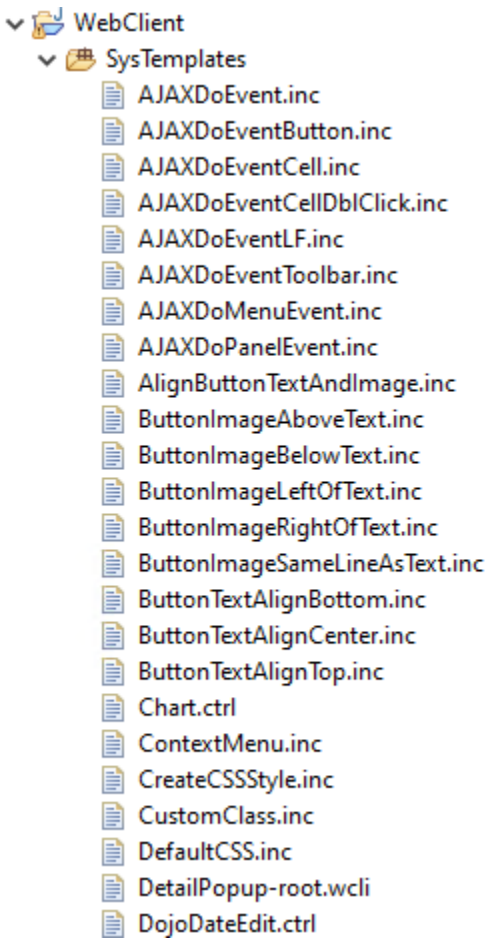
In the address bar of the browser, enter http://localhost:8080/replatformweb/WebClientServlet



Everything looks good, but let's test the application.

Change the email address to “test” and press Apply. We are expecting an error message as the email address is invalid, but nothing is displayed.

The reason for this is that we need to specify a page template for to display messages. Let’s take a look at templates. In Eclipse, expand the WebClient project, then expand the SysTemplate folder.



In general, there are 3 types of templates:

- Root templates – These are identified as having a *-root.wcli* suffix, and there are two in the SysTemplates folder, DetailPopup-root.wcli and WebShell-root.wcli. Root templates form the base for the other templates to build on, similar to FunctionShell in Plex. A function needs to inherit from at least one of these Root templates, or more exactly, a function needs to inherit from a function with an implementation name that matches one of these root templates, i.e. “WebShell”, or “DetailPopup”. WebShell is used for entry functions, or functions that can take up the whole page in a browser, while DetailPopup will display the function in a Dialog box in the foreground of the calling function. A DetailPopup always needs to be called from another function, so it can’t be used as an entry function.
- Page templates – Contain HTML and JavaScript that can be added on top of a Root template. Page templates are identified as having a -page.wcli suffix. A function can inherit from many Page templates. Again, WebClient uses the implementation name of functions in the inheritance path to find matching page templates.

- Control Templates – These templates have a *.ctrl* suffix. A control template provides the functionality for individual controls on your panel. For example, the WebGrid.ctrl template provides the ability to display and edit data, select rows, re-order columns etc. Control templates can be overridden, so a grid on a panel can be displayed as a Pie Chart, or points on a map.

Coming back to the messages, WebClient provides different page templates for how you want your application messages to be handled. We will use the *WebLogMessages-page.wcli* template, this template treats Dialog and Log messages the same, showing multiple messages at a time in a pop-up dialog.

Applying this page template is easy, just inherit from a function which has the implementation name "WebLogMessages".

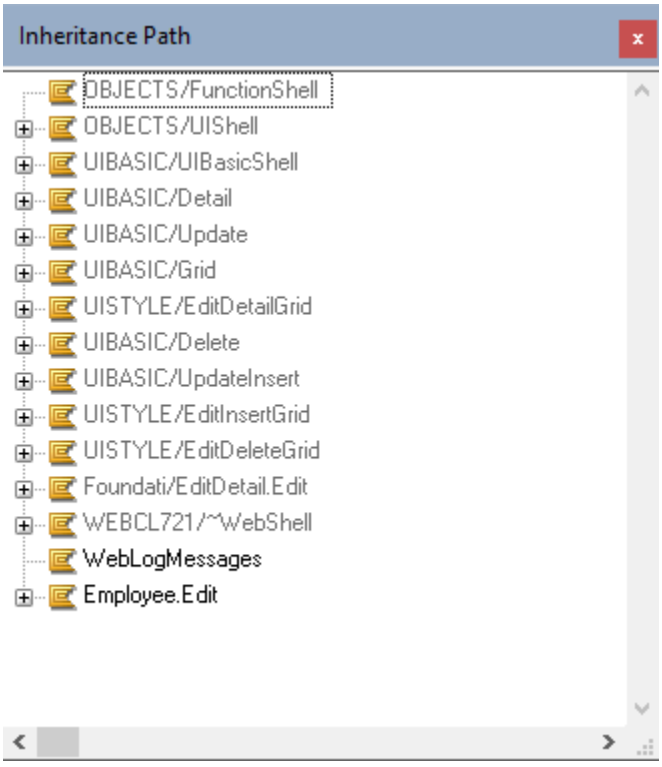
5. Provide Messaging

The WebLogMessages function doesn't exist in our model, so we'll create it here.

Add the following triples:

WebLogMessages	FNC impl name NME	WebLogMessages
Employee.Edit	FNC is a FNC	WebLogMessages

Check your inheritance path, make sure that the WebLogMessages is below ~WebShell. WebClient will ignore any page templates above the root template.



It's important to note that this inheritance does not affect the behavior of the native Java application. You will still be able to run this function as a Java Swing application.

6. Try out your changes

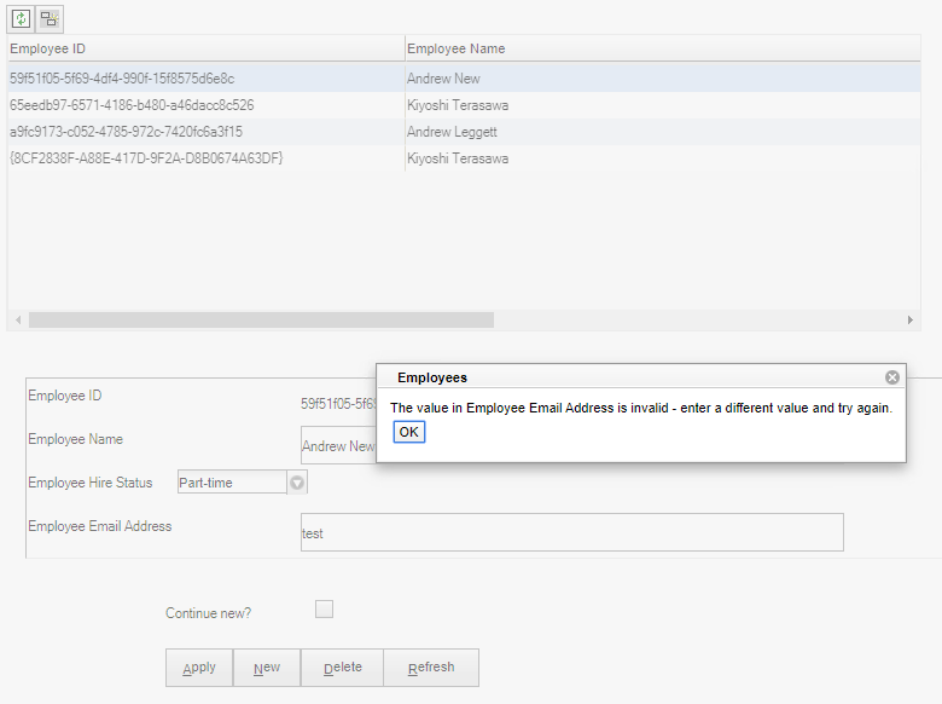
In Plex, generate your function Employee.Edit function.

Switch to Eclipse, right-click on the ReplatformJava project and select Refresh (or press F5). Eclipse will build your latest generated function and you will see the progress of the WebClient template generator in the Console tab.

```
|
>>> Starting WebClient Build for ReplatformJava

> Using CM WebClient v1.8.6-pre12879
> Licensed to CMFirst Technologies
> Valid until Wed Dec 30 00:00:00 CST 2020

Generating Web templates.
....

>>> WebClient Build complete.
    1 panels processed. 0 panels not processed.
```

Switch to the server tab, right-click on the Tomcat server and select Start, or Restart. Once the status has changed to [Started, Synchronized], open Chrome and visit http://localhost:8080/replatformweb/WebClientServlet

Now if you change the email address to 'test' and press Apply you will see the error message.



Congratulations, you have successfully created a WebClient application.

Source code Sample Answers

C# code for email validation

You can code like C++ to read char by char. But, .NET framework provides MailAddress class and it gives exception if you give a wrong email.

```
using System.Net.Mail;
{
        try
  {
    MailAddress m = new MailAddress(&(1:).Value);

    &(2:).fromString("Y");
  }
  catch (FormatException)
  {
    &(2:).fromString("N");
  }
}
```

C# code for GUID generation

Simply use System.Guid class

```
{
        Guid g = Guid.NewGuid();
        &(1:).fromString(g.ToString());
}
```

Java code for email validation

Using REGEX (Regular Expression)

top-15-commonly-used-regex

```
{
        String regex = "^[\\w-_\\.+]*[\\w-_\\.]\\@([\\w]+\\.)+[\\w]+[\\w]$";
  if(&(1:).getValue().matches(regex))
        {
                &(2:).fromString("Y");
        }
        else
        {
                &(2:).fromString("N");
        }
```

Java code for GUID generation

Simply use java.util.UUID class

```
{
        &(1:).fromString(java.util.UUID.randomUUID().toString());
}
```