CA 2E / CA Plex Worldwide Developers Conference 2023

Web Technology Workshop

Andrew Leggett & Lily Taharudin – CM First

Table of Contents

Workshop Outline

Traditionally the conference workshops provide an opportunity to learn new topics and try out new things that you may be able to apply to your daily jobs.

At CM First we work with our customers to modernize their computer systems using the latest tools and technologies, and these technologies are becoming increasingly web-based. It's likely that your business will have increased exposure to these tools and may already be using some of them. We'll use this workshop as an opportunity to learn more about what they have to offer and try out a few approaches we can use to integrate with our existing CA Plex applications.

In this workshop, we'll be using the following tools and technologies:

- HTML / JavaScript / CSS – website design
- Git – Source control
- VSCode – Modern development environment
- React – JavaScript library for Web applications
- Node.JS – JavaScript server runtime environment
- npm – Node package manager
- REST APIs
- CA Plex / CM WebClient / CM HSync
- chatGPT – AI chatbot

Exercise One – Git, VS Code and HTML

This exercise introduces Git source control combined with the VS Code development environment to create a very simple website.

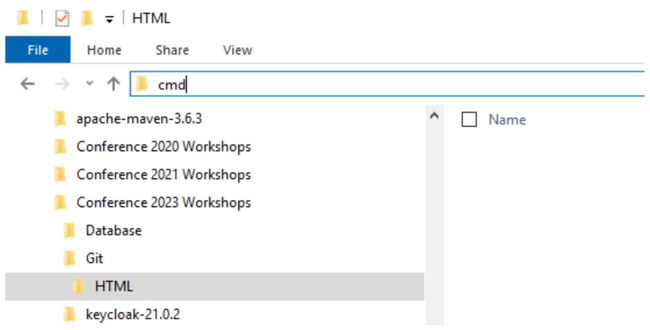
Clone the Repository

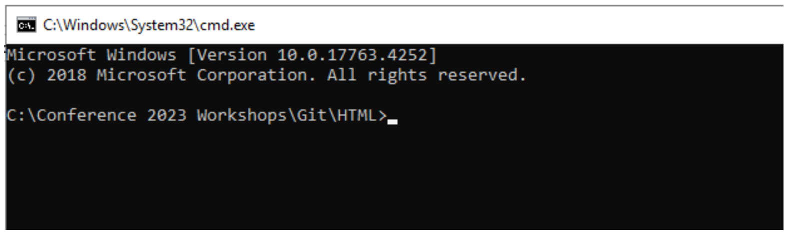
An existing repository has been set up for this workshop exercise at https://github.com/acl-cm1/htmlexercise.git

. We'll use this repository to manage our individual exercise work files.

In File Explorer navigate to the folder C:\Conference 2023 Workshops\Git\HTML.

In the address bar, type 'cmd' then press enter (this is a quick way to open the command window in the current folder)

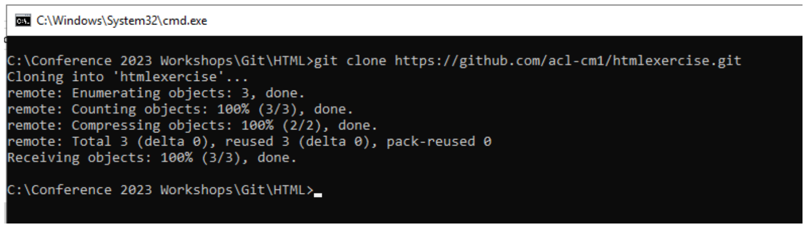




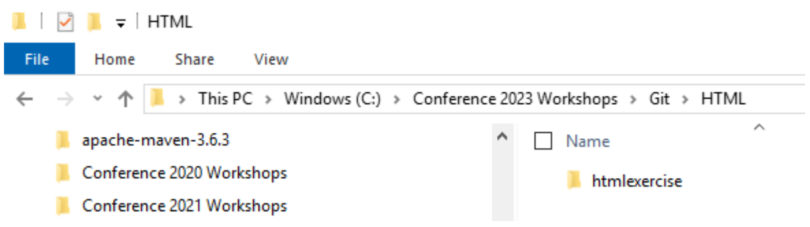
```
Microsoft Windows [Version 10.0.17763.4252]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Conference 2023 Workshops\Git\HTML>_
```

Enter the following command:

git clone https://github.com/acl-cm1/htmlexercise.git



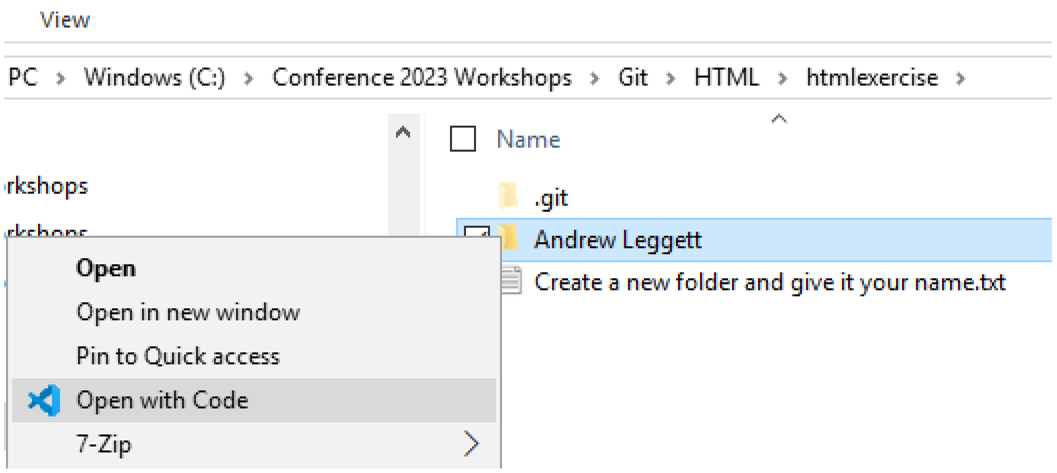
```
C:\Conference 2023 Workshops\Git\HTML>git clone https://github.com/acl-cm1/htmlexercise.git
Cloning into 'htmlexercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

C:\Conference 2023 Workshops\Git\HTML>_
```


HTML
File
Home
Share
View
This PC › Windows (C:) › Conference 2023 Workshops › Git › HTML
apache-maven-3.6.3
Conference 2020 Workshops
Conference 2021 Workshops
Name
htmlexercise

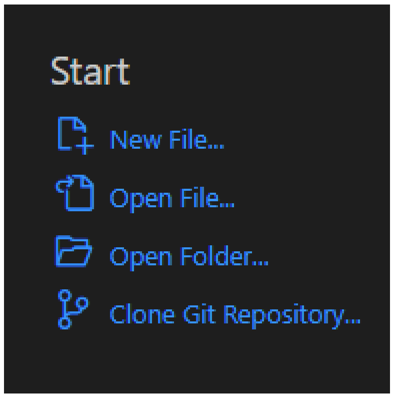
This command makes a copy of the git repository stored at that location. You should now see a folder named 'htmlexercise'. Open that folder and create a new folder and name it after yourself (this will allow everyone to work on the same exercise independently).

Create a simple website

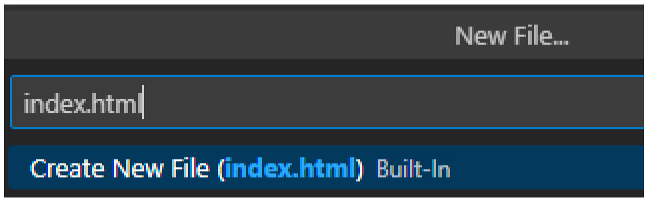
Right-click on your own folder and select 'Open with Code' to open VS Code.


View
PC › Windows (C:) › Conference 2023 Workshops › Git › HTML › htmlexercise ›
rkshops
Name
.git
Andrew Leggett
Create a new folder and give it your name.txt
Open
Open in new window
Pin to Quick access
Open with Code
7-Zip

When VS Code opens, click on 'New File…'



Enter the name 'index.html' and click on 'Create File' in the file dialog.



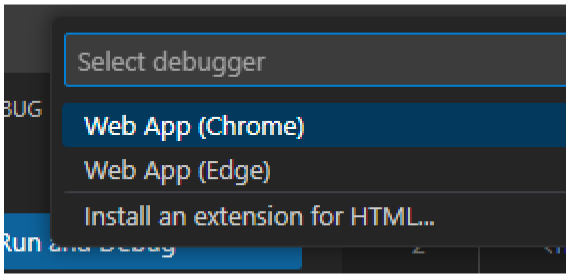
Your index.html file will show in the text exitor.

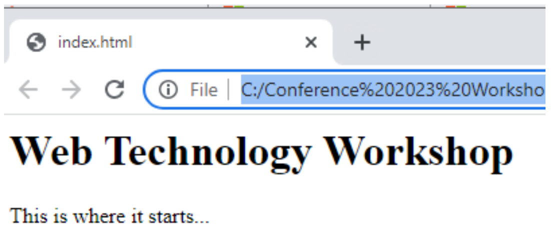
Copy the following HTML code into the editor.

```
<body>
    <h1>Web Technology Workshop</h1>
    <p>This is where it starts...</p>
</body>
```

Click on the icon on the left for the Run & Debug option, then press the blue ‘Run and Debug’ button.

VS Code recognises that you have an HTML file open, so will offer to run it in a web browser. Select ‘Web App (Chrome)’





Web Technology Workshop

This is where it starts...

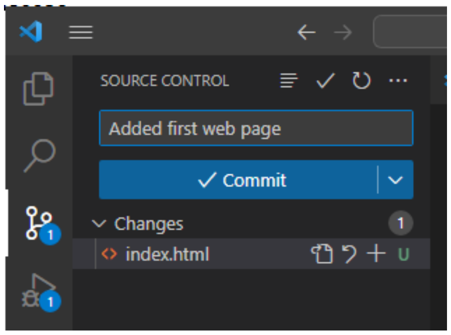
The Chrome browser should open and display your web page. We now have a working site, so we want to commit this to Git so that we can return to it later if our next changes don’t work out.

Commit your changes

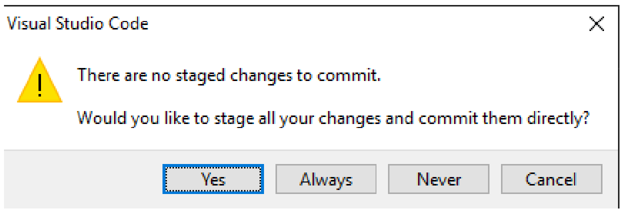
VS Code is integrated with Git, so we can do this without leaving the development environment.

Click on the Source control button on the left. It shows you have 1 change made. You should always enter a meaningful message when committing a change so you can refer back to it later if needed.

Enter “Added first web page” then press Commit.



Added first web page


Visual Studio Code

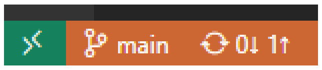
There are no staged changes to commit.

Would you like to stage all your changes and commit them directly?

VS Code will prompt you to ask if you want to 'Stage' your changes before committing. Staging allows you move your changes to an intermediate storage area before committing. This can be useful if you have to manage several changes at a time. We only have a simple change, so click on 'Yes'.

Note that your change is committed, i.e. it has been updated in your local repository (the .git folder on your hard drive), but it hasn't been 'Pushed' to the network repository yet. We don't need to do that yet.

The information box at the bottom left shows that you are on the 'main' branch, and there are no changes to 'Pull' (the down arrow), and one change to 'Push' (the up arrow).



We will synchronize our changes with the network repository later.

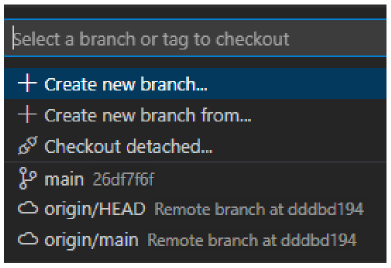
Create a new Branch

Branches allow you to make isolated changes to your project without affecting the main application, e.g. bug fixes, new features, experimentation etc. You can switch back and forth between branches and git will manage the state. Changes on a branch can be merged back to another branch later when the developer is satified that everything is working okay.

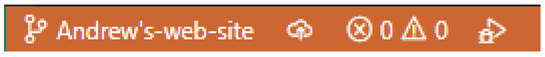
We'll create a new branch to enhance the application without risking what we've already worked on.

Click on the 'main' branch on the bottom left.

At the top of the panel you'll see some options. Enter a branch name including your name (to keep it unique within the group), e.g. "Andrew's web site" and click 'Create new Branch'


Select a branch or tag to checkout
Create new branch...
Create new branch from...
Checkout detached...
main 26df7f6f
origin/HEAD Remote branch at dddbd194
origin/main Remote branch at dddbd194

Note that the branch information shows the current branch information.

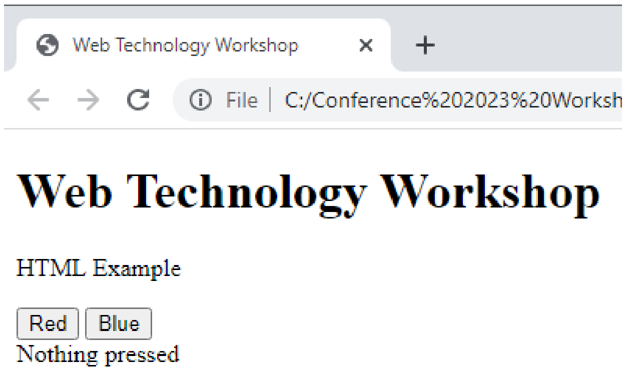


We can continue to work on our website.

In the index.html editor, replace the whole contents with the following HTML.

```
<!DOCTYPE html>
<html>
<head>
    <title>Web Technology Workshop</title>
</head>
<link rel="stylesheet" href="workshop.css">
<script src="workshop.js"></script>

<body>
    <h1>Web Technology Workshop</h1>
    <p>HTML Example</p>
    <button class="redBtn" onclick="buttonClicked(this)">Red</button>
    <button class="blueBtn" onclick="buttonClicked(this)">Blue</button>
    <div id="btnInfo">Nothing pressed</div>
</body>
</html>
```

Save the file by pressing Ctrl+S. You are still in the 'Run and debug' mode, so you can preview your changes. Switch to the Chrome browser and press F5 to refresh the page.



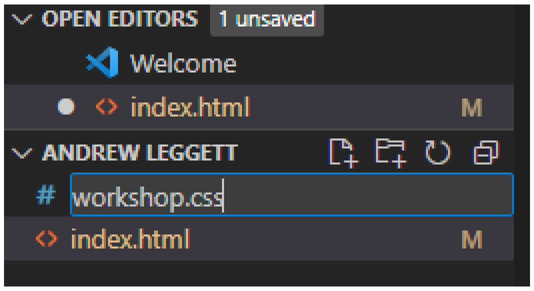
It still looks plain, and clicking a button will show an error in the VS code Debug Console.

```
Uncaught ReferenceError ReferenceError: buttonClicked is not defined at onclick
(c:\Conference 2023 Workshops\Git\HTML\htmlexercise\Andrew
Leggett\index.html:13:58)
```

This is because the code we changed is referencing two files that don't exist yet. We'll create them now.

```
<link rel="stylesheet" href="workshop.css">
<script src="workshop.js"></script>
```

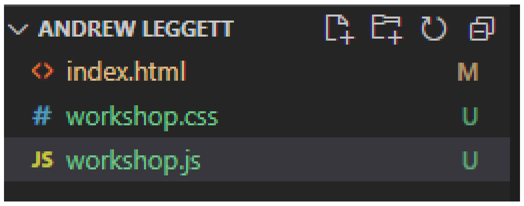
Click on the File Explorer button on the left - , then click on the 'Add File' button . Name the file "workshop.css" and press enter.



Click on the file to open the editor. Paste the following into the workshop.css file.

```
body {
    background: lightblue;
    font-family: Verdana, Arial, Tahoma, Serif;
}

h1 {
    color: green;
}

button {
    color: white;
}

.redBtn {
    background: red;
}

.blueBtn {
    background: blue;
}
```

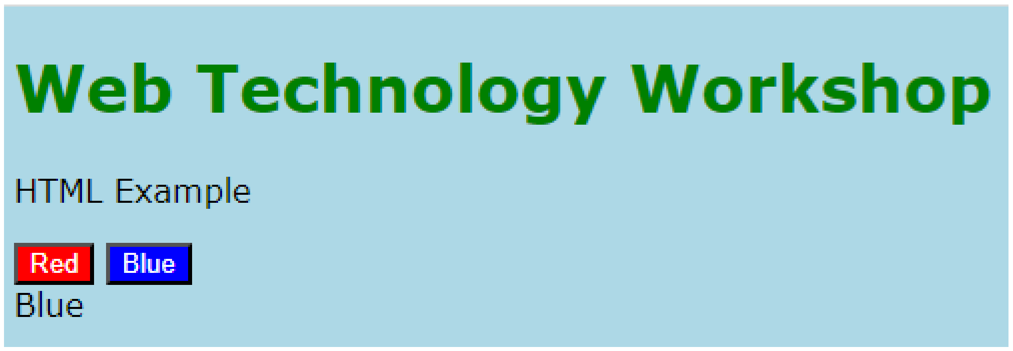
Save the file by pressing Ctrl+S. Now repeat the step above to create the “workshop.js” file.



Paste the following code into the workshop.js file and save it.

```
function buttonClicked(btn) {
    var info = document.getElementById('btnInfo');
    info.textContent = btn.textContent;
}
```

Switch to Chrome and refresh the page. Now the page should display correctly, and clicking the buttons should display the name of the button.

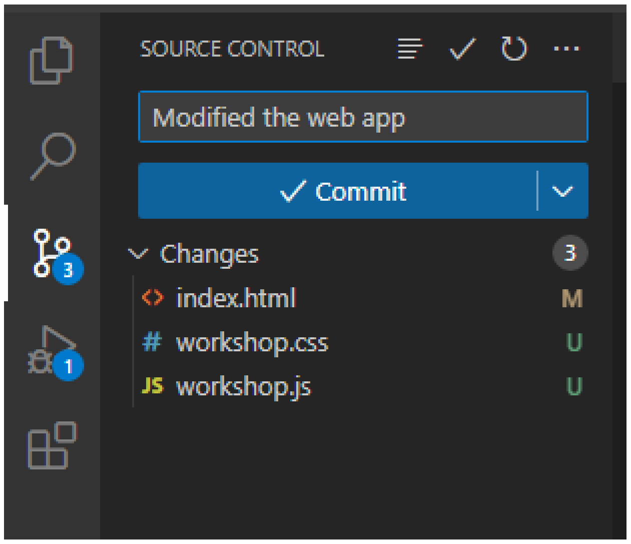

Web Technology Workshop

HTML Example

Red Blue

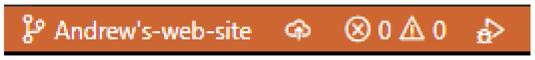
Blue

We can now commit our changes to this branch. Go to the Source control tab on the left.

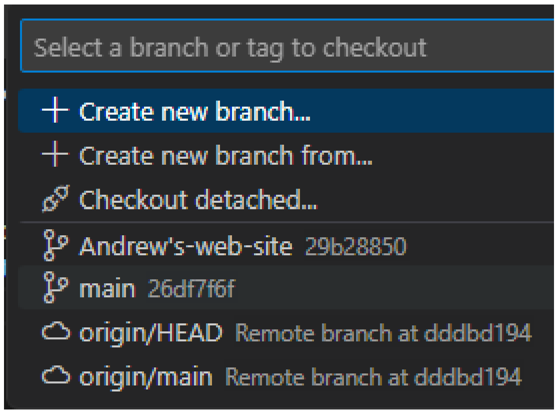


The index.html has a 'M' against it for 'Modified', and the new workshop files have a 'U' against them for 'Unversioned'. Enter a message, e.g. "Modified the web app" and press "Commit". Again click "Yes" on the Staging warning message.

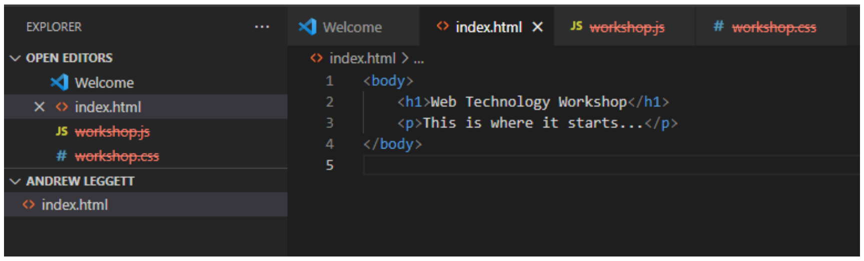
After your changes are committed, you can switch back to the original "main" branch.

Click on your branch name at the bottom of the screen 

At the top of the screen, click on the branch that just says "main"

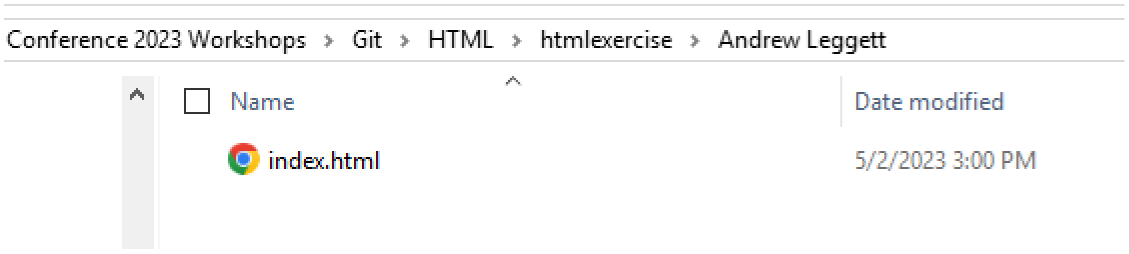

Select a branch or tag to checkout
Create new branch...
Create new branch from...
Checkout detached...
Andrew's-web-site 29b28850
main 26df7f6f
origin/HEAD Remote branch at dddbd194
origin/main Remote branch at dddbd194

The workshop files have lines through the names, because they didn't exist in the main branch.



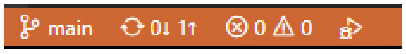
```
<body>
    <h1>Web Technology Workshop</h1>
    <p>This is where it starts...</p>
</body>
```

In File Explorer, they are no longer there.

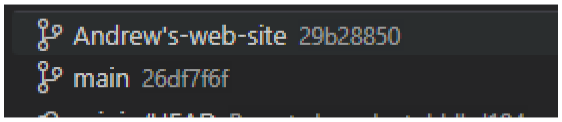

Conference 2023 Workshops > Git > HTML > htmlexercise > Andrew Leggett
Name
Date modified

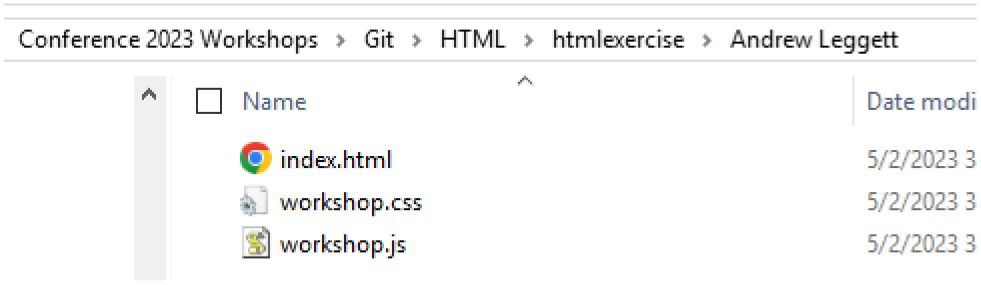
Name	Date modified
index.html	5/2/2023 3:00 PM

Don't worry, they have been saved. We can switch back to your branch to check.

Click on 'main' again. 

Now select the branch you created, and the files return just as you left them.




Conference 2023 Workshops > Git > HTML > htmlexercise > Andrew Leggett
Name
Date modi

Name	Date modi
index.html	5/2/2023 3
workshop.css	5/2/2023 3
workshop.js	5/2/2023 3

Exercise Two – Node Server

In this exercise we'll demonstrate how to create a simple web server using only node.js

Follow the instructions in this document - C:\Conference 2023 Workshops\node\01-simple-server\starter\Instructions.docx

Exercise Three – React Primer

This exercise demonstrates a very simple React application, and introduces NPM.

Follow the instructions in this document - C:\Conference 2023 Workshops\react\01-simple-react-app\starter\Instructions.docx

Exercise Four – React Menu Application

This exercise uses the react-router library for client-side routing.

Follow the instructions in this document - C:\Conference 2023 Workshops\react\02-react-router-demo\starter\Instructions.docx

Exercise Five – CM WebClient application

Prerequisite

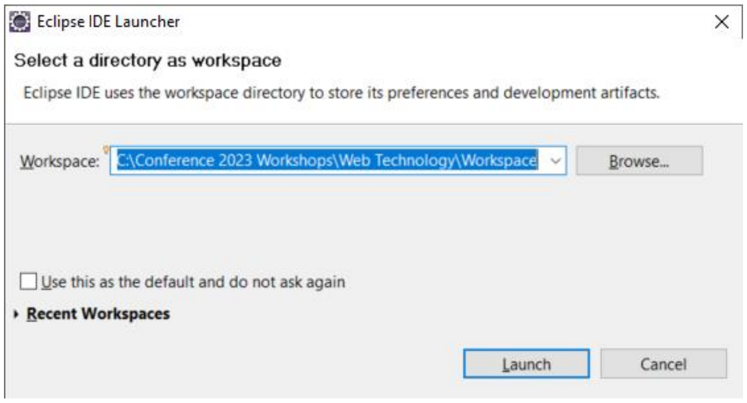
- Restore the *Web Technology-wcStart.7z* snapshot using the instructions here –> Restoring Snapshots
- (Optional) To skip the exercise, restore the *Web Technology-wcEnd.7z* snapshot using the instructions here –> Restoring Snapshots and skip to section.

Test the Java Client application

The application is a very simple store application. There are editing screens for Products and Customers, and the ability to create orders for the products, which are types of robots.

Open Eclipse by clicking on the icon in the Taskbar -

You will be prompted to select the Workspace location. Make sure this is set to *C:\Conference 2023 Workshops\Web Technology\Workspace*



Eclipse IDE Launcher

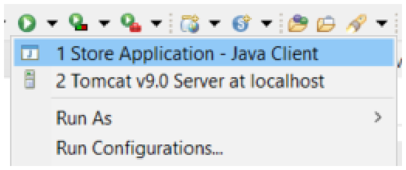
Select a directory as workspace

Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

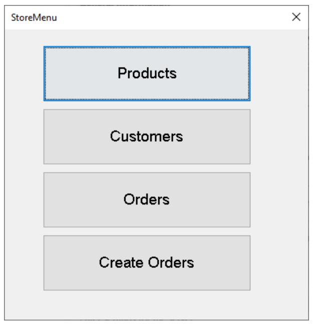
Workspace: C:\Conference 2023 Workshops\Web Technology\Workspace

Click the “Launch” button.

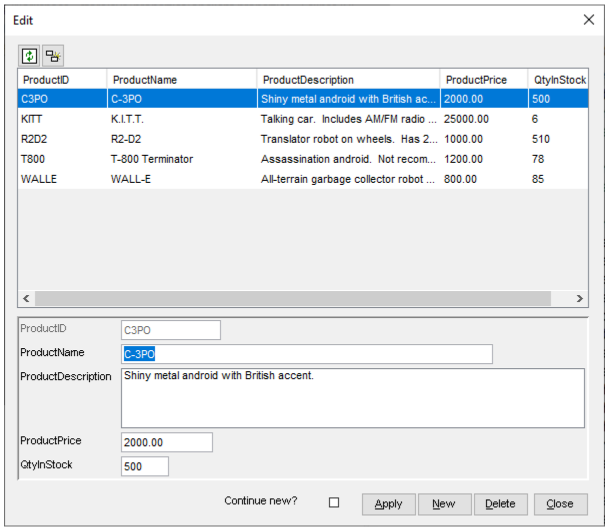
On the Eclipse toolbar, click on the play icon, then select “Store Application – Java Client”


1 Store Application - Java Client
2 Tomcat v9.0 Server at localhost
Run As
Run Configurations...

The application will launch.



Click on the Products button and add a couple of Product records.


Edit

ProductID	ProductName	ProductDescription	ProductPrice	QtyInStock
C3PO	C-3PO	Shiny metal android with British ac...	2000.00	500
KITT	K.I.T.T.	Talking car. Includes AM/FM radio ...	25000.00	6
R2D2	R2-D2	Translator robot on wheels. Has 2...	1000.00	510
T800	T-800 Terminator	Assassination android. Not recom...	1200.00	78
WALLE	WALL-E	All-terrain garbage collector robot ...	800.00	85

ProductID	C3PO
ProductName	C-3PO
ProductDescription	Shiny metal android with British accent.
ProductPrice	2000.00
QtyInStock	500

Continue new? ☐

Set up WebClient inheritance

We want to be able to run this same application in a browser, so we can use CM WebClient to do this.

If you're not familiar with CM WebClient, it's a tool that takes Plex generated Java functions and generates the appropriate HTML and JavaScript to run in the browser as if it was a desktop application. It can do a lot more beyond this to create fully featured web applications, but we'll keep it simple so that we can make it work with our React application.

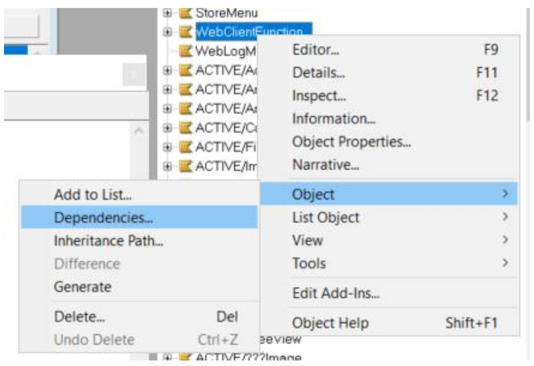
Open Plex by clicking on the taskbar icon at the bottom of the screen

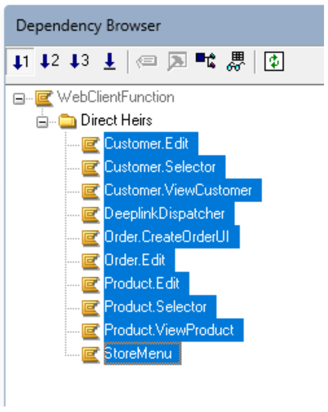
Open the file *C:\Conference 2023 Workshops\Web Technology\Plex\store.mdl*

Add the following triples to the UI functions below:

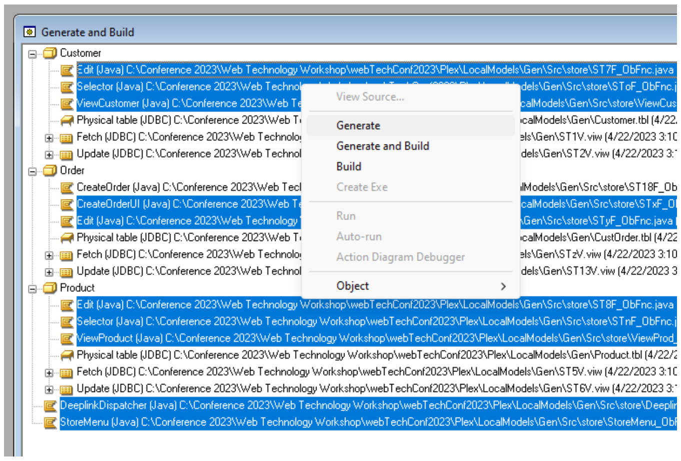
Customer.Edit	FNC is a FNC	WebClientFunction
Customer.Selector	FNC is a FNC	WebClientFunction
Customer.ViewCustomer	FNC is a FNC	WebClientFunction
Order.CreateOrderUI	FNC is a FNC	WebClientFunction
Order.Edit	FNC is a FNC	WebClientFunction
Product.Edit	FNC is a FNC	WebClientFunction
Product.Selector	FNC is a FNC	WebClientFunction
Product.ViewProduct	FNC is a FNC	WebClientFunction
StoreMenu	FNC is a FNC	WebClientFunction

Generate all dependencies of WebClientFunction. You can find them all by right-clicking on the 'WebClientFunction' and selecting Object -> Dependencies…





Drag them all to the Generate & Build window. Select 'Generate' only. We will build them in Eclipse.



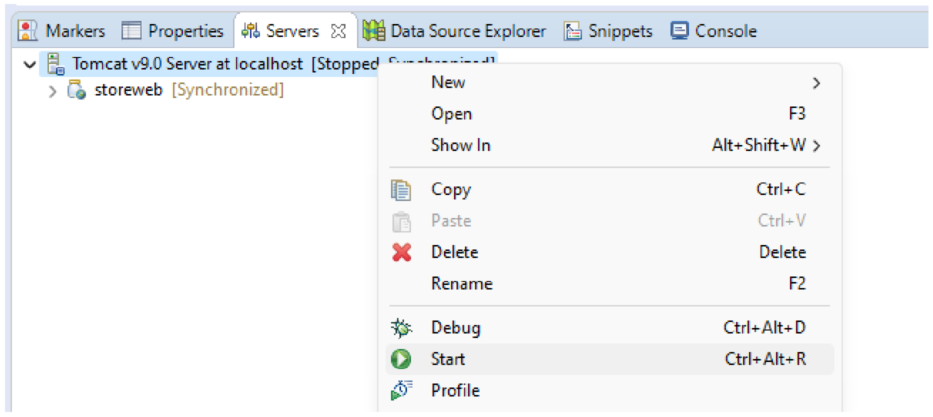
```
Customer
    Edit (Java) C:\Conference 2023\Web Technology Workshop\webTechConf2023\Plex\LocalModels\Gen\Src\store\ST7F_ObFnc.java
    Selector (Java) C:\Conference 2023\Web Technol                    dels\Gen\Src\store\SToF_ObFnc.j
    ViewCustomer (Java) C:\Conference 2023\Web Te                     calModels\Gen\Src\store\ViewCus
    Physical table (JDBC) C:\Conference 2023\Web T                    calModels\Gen\Customer.tbl (4/22.
    Fetch (JDBC) C:\Conference 2023\Web Technolo                      els\Gen\ST1V.viw (4/22/2023 3:10
    Update (JDBC) C:\Conference 2023\Web Technol                      dels\Gen\ST2V.viw (4/22/2023 3:
Order
    CreateOrder (Java) C:\Conference 2023\Web Tec                     lModels\Gen\Src\store\ST18F_Ob
    CreateOrderUI (Java) C:\Conference 2023\Web Te                    calModels\Gen\Src\store\STxF_O
    Edit (Java) C:\Conference 2023\Web Technology                     Gen\Src\store\STyF_ObFnc.java
    Physical table (JDBC) C:\Conference 2023\Web T                    calModels\Gen\CustOrder.tbl (4/22
    Fetch (JDBC) C:\Conference 2023\Web Technolo                      els\Gen\STzV.viw (4/22/2023 3:10
    Update (JDBC) C:\Conference 2023\Web Technol                      dels\Gen\ST13V.viw (4/22/2023 3
Product
    Edit (Java) C:\Conference 2023\Web Technology Workshop\webTechConf2023\Plex\LocalModels\Gen\Src\store\ST8F_ObFnc.java
    Selector (Java) C:\Conference 2023\Web Technology Workshop\webTechConf2023\Plex\LocalModels\Gen\Src\store\STnF_ObFnc.j
    ViewProduct (Java) C:\Conference 2023\Web Technology Workshop\webTechConf2023\Plex\LocalModels\Gen\Src\store\ViewProd_
    Physical table (JDBC) C:\Conference 2023\Web Technology Workshop\webTechConf2023\Plex\LocalModels\Gen\Product.tbl (4/22/2
    Fetch (JDBC) C:\Conference 2023\Web Technology Workshop\webTechConf2023\Plex\LocalModels\Gen\ST5V.viw (4/22/2023 3:10
    Update (JDBC) C:\Conference 2023\Web Technology Workshop\webTechConf2023\Plex\LocalModels\Gen\ST6V.viw (4/22/2023 3:
DeeplinkDispatcher (Java) C:\Conference 2023\Web Technology Workshop\webTechConf2023\Plex\LocalModels\Gen\Src\store\Deeplin
StoreMenu (Java) C:\Conference 2023\Web Technology Workshop\webTechConf2023\Plex\LocalModels\Gen\Src\store\StoreMenu_ObF
```

If Eclipse is not already open, open it following the steps in Test the Java Client application

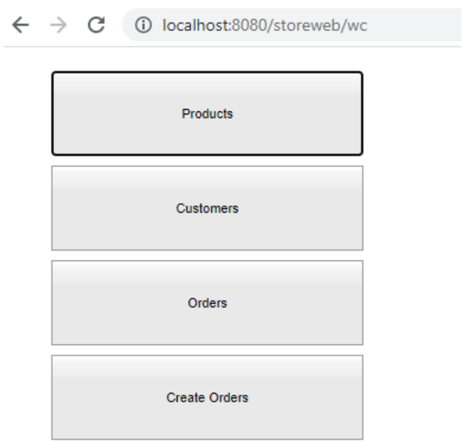
Select the 'storeJava' project and press F5 to refresh the workspace to include the files you have just generated. The console should display a message something like this:

```
>>> Starting WebClient Build for storeJava

> Using CM WebClient v1.8.8-pre13284
> Licensed to CMFirst Technologies
> Valid until Mon Jan 01 00:00:00 CST 2024

Generating Web templates.
....

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

Near the bottom of the screen, right-click on the Servers tab and select Start.



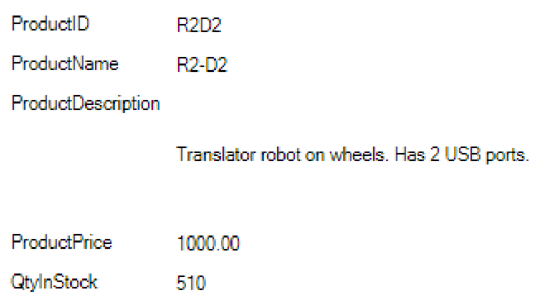
In your browser navigate to URL - http://localhost:8080/storeweb/wc



The workspace also includes a "Deep Link Dispatcher" this is an extra utility which allows you to call a particular function directly by passing in the function's implementation name and input parameters. We will use these links in a future exercise.

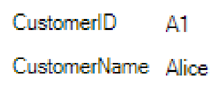
Try the links below.

http://localhost:8080/storeweb/deeplink/store.ViewProd,Input.ProdID=R2D2



ProductID	R2D2
ProductName	R2-D2
ProductDescription	Translator robot on wheels. Has 2 USB ports.
ProductPrice	1000.00
QtyInStock	510

http://localhost:8080/storeweb/deeplink/store.ViewCust,Input.CustID=A1



CustomerID	A1
CustomerName	Alice

If you created a new product in the previous step, you can access that by adding its key value to the URL, e.g. if you created a product with Product ID ‘AAA’, you can access it directly using the URL.

http://localhost:8080/storeweb/deeplink/store.ViewProd,Input.ProdID=AAA

Exercise Six – React with CM WebClient

Access the CM WebClient pages and the HSync API resources from the React website.

Follow the instructions in this document - C:\Conference 2023 Workshops\react\03-react-node-app\starter\Instructions.docx

Exercise Seven – HSync API

Prerequisite

- Restore the *Web Technology-hsStart.7z* snapshot using the instructions here –> Restoring Snapshots
- (Optional) To skip the exercise, restore the *Web Technology-hsEnd.7z* snapshot using the instructions here –> Restoring Snapshots and skip to section

HSync

HSync is a tool that can extract data from a CA Plex function, e.g. Field names and data types, and combine that data with a template to generate files. It can generate any text file type, including HTML, JavaScript, Java etc.

We will use the REST API templates to generate an API that can take a request, pass the parameters to call a Plex generated Java function and return the data in JSON format. The advantage of calling Plex functions from the API allows us to utilize our existing business logic to ensure that the data retrieved or updated is handled according to our business rules.

HSync is provided via a group model. An entity can work with the provided templates by inheriting from *TemplateGeneratorEntity* to extract the model information, and API resource functionality can be provided by inheriting any combination of *Ability.CreateAPI*, *Ability.RetrieveAPI, Ability.UpdateAPI,* or *Ability.DeleteAPI*.

The collection of entities can be grouped into an *Application*, and defined by ENT comprises ENT triples. The Application entity inherits the functionality to generate the output files from the template.

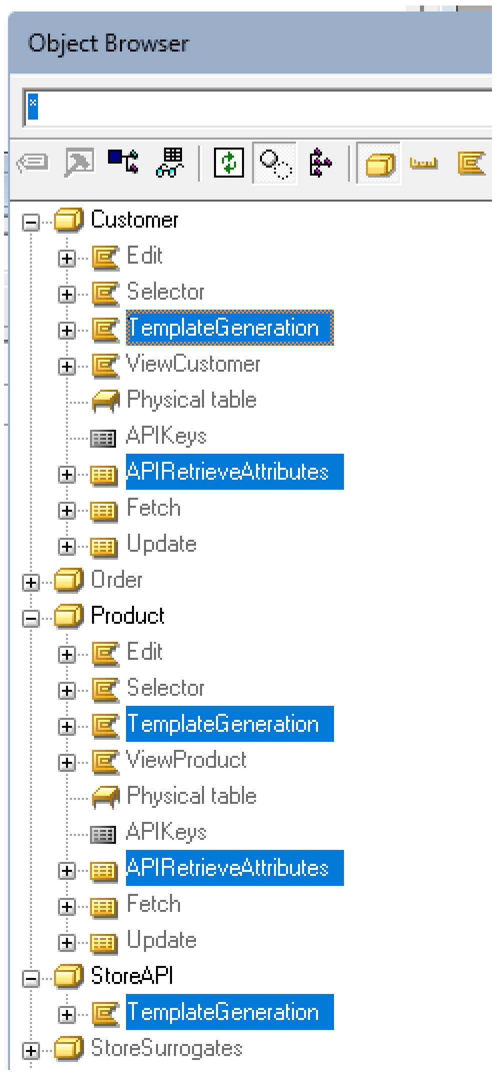
Set up the Plex model

We can set up the existing Product and Customer entities to provide an API to allow their data to be accessed from outside the CA Plex application.

Add the following triples:

Customer	ENT is a ENT	TemplateGeneratorEntity
Customer	ENT is a ENT	Ability.RetrieveAPI
Product	ENT is a ENT	TemplateGeneratorEntity
Product	ENT is a ENT	Ability.RetrieveAPI
StoreAPI	ENT is a ENT	Application
StoreAPI	ENT comprises ENT	Customer
StoreAPI	ENT comprises ENT	Product

Select the inherited 'TemplateGeneration' functions and 'APIRetrieveAttributes' views scoped to our entities (which will show as grey), and make them real (turn them yellow) by right-clicking in the Object Brower and selecting Tools -> Make Real.

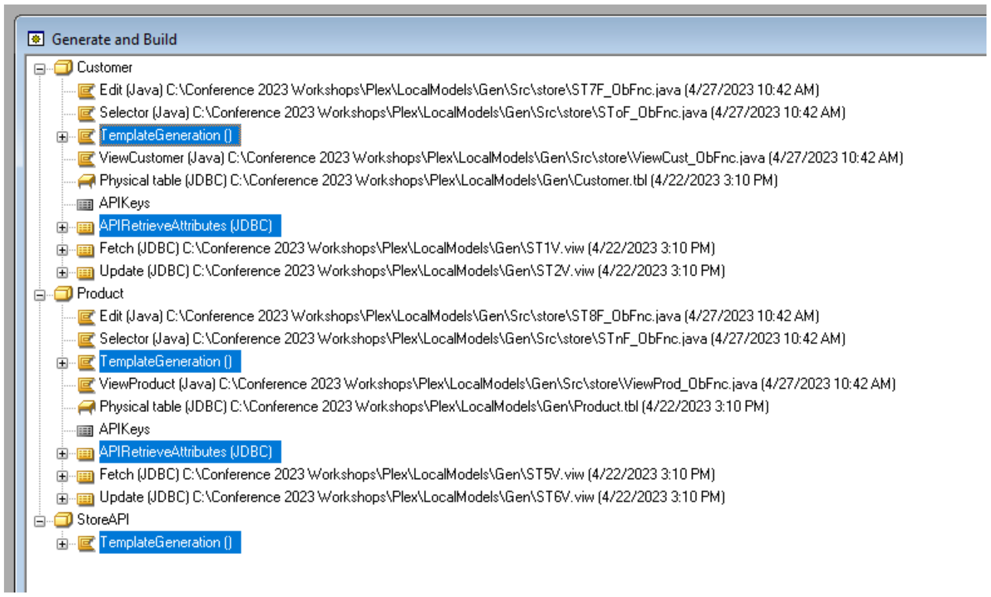


Finally, we'll give our main generator function a meaningful implementation name. Add the following triple:

StoreAPI.TemplateGeneration.TemplateGeneratorCommand FNC file name NME StoreAPIGen

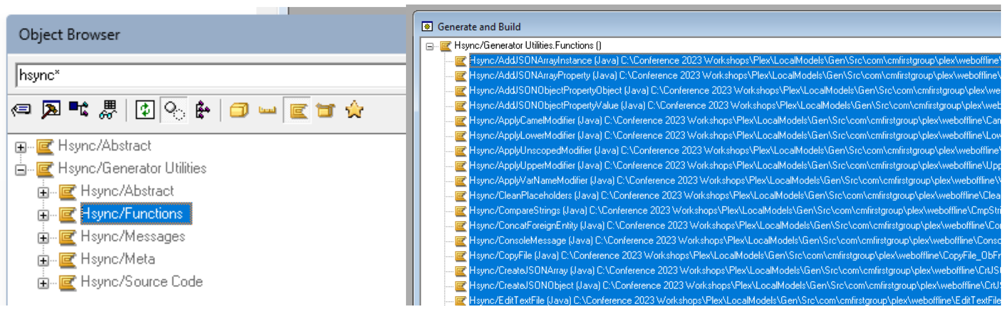
Generating the API

Select all the 'TemplateGeneration' functions and 'APIRetrieveAttributes' views and drag them to the Gen & Build window. Generate them only, they will be built in Eclipse.



```
Generate and Build
⊟ Customer
    Edit (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\store\ST7F_ObFnc.java (4/27/2023 10:42 AM)
    Selector (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\store\SToF_ObFnc.java (4/27/2023 10:42 AM)
  ⊞ TemplateGeneration ()
    ViewCustomer (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\store\ViewCust_ObFnc.java (4/27/2023 10:42 AM)
    Physical table (JDBC) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Customer.tbl (4/22/2023 3:10 PM)
    APIKeys
  ⊞ APIRetrieveAttributes (JDBC)
  ⊞ Fetch (JDBC) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\ST1V.viw (4/22/2023 3:10 PM)
  ⊞ Update (JDBC) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\ST2V.viw (4/22/2023 3:10 PM)
⊟ Product
    Edit (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\store\ST8F_ObFnc.java (4/27/2023 10:42 AM)
    Selector (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\store\STnF_ObFnc.java (4/27/2023 10:42 AM)
  ⊞ TemplateGeneration ()
    ViewProduct (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\store\ViewProd_ObFnc.java (4/27/2023 10:42 AM)
    Physical table (JDBC) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Product.tbl (4/22/2023 3:10 PM)
    APIKeys
  ⊞ APIRetrieveAttributes (JDBC)
  ⊞ Fetch (JDBC) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\ST5V.viw (4/22/2023 3:10 PM)
  ⊞ Update (JDBC) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\ST6V.viw (4/22/2023 3:10 PM)
⊟ StoreAPI
  ⊞ TemplateGeneration ()
```

There are some additional functions needed for HSync. Look in the Object Browser for the function object "*HSync/Generator Utilities.Functions*" and drag it to the Gen & Build window. Again, generate them only, they will be built in Eclipse.



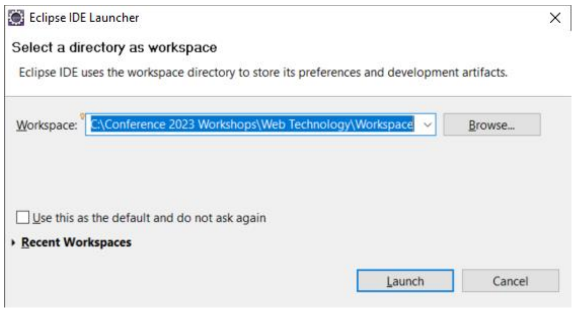
```
Object Browser
hsync*
⊞ Hsync/Abstract
⊟ Hsync/Generator Utilities
  ⊞ Hsync/Abstract
  ⊞ Hsync/Functions
  ⊞ Hsync/Messages
  ⊞ Hsync/Meta
  ⊞ Hsync/Source Code
```

```
Generate and Build
⊟ Hsync/Generator Utilities.Functions ()
    Hsync/AddJSONArrayInstance (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline
    Hsync/AddJSONArrayProperty (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline
    Hsync/AddJSONObjectPropertyObject (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\we
    Hsync/AddJSONObjectPropertyValue (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\web
    Hsync/ApplyCamelModifier (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline\Can
    Hsync/ApplyLowerModifier (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline\Low
    Hsync/ApplyUnscopedModifier (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline
    Hsync/ApplyUpperModifier (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline\Upp
    Hsync/ApplyVarNameModifier (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline\
    Hsync/CleanPlaceholders (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline\Clea
    Hsync/CompareStrings (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline\CmpStri
    Hsync/ConcatForeignEntity (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline\Co
    Hsync/ConsoleMessage (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline\Consc
    Hsync/CopyFile (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline\CopyFile_ObFr
    Hsync/CreateJSONArray (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline\CrtJS
    Hsync/CreateJSONObject (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline\CrtJ
    Hsync/EditTextFile (Java) C:\Conference 2023 Workshops\Plex\LocalModels\Gen\Src\com\cmfirstgroup\plex\weboffline\EditTextFile
```

Calling the HSync API Generator

Open Eclipse by clicking on the icon in the Taskbar -

You will be prompted to select the Workspace location. Make sure this is set to *C:\Conference 2023 Workshops\Web Technology\Workspace*



Eclipse IDE Launcher

Select a directory as workspace

Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Conference 2023 Workshops\Web Technology\Workspace

☐ Use this as the default and do not ask again

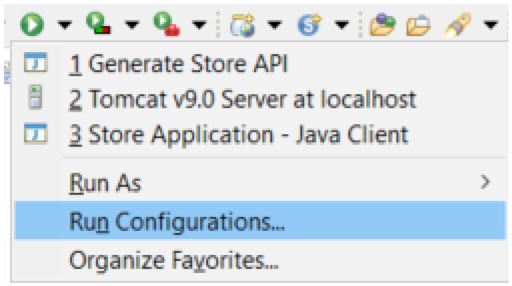
Recent Workspaces

Click the “Launch” button.

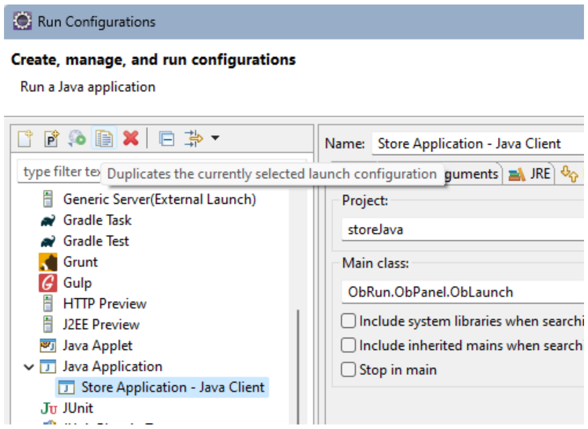
Once Eclipse has loaded, refresh the workspace by selecting the ‘storeAPI’ project and pressing F5 to build the generated Plex Java functions.

We can now create a shortcut in Eclipse to call our the “StoreAPIGen” function that we just generated from our Plex model. We already have a shortcut set up to launch the Java Store application, so we can start by copying that and adapting it to call our new function.

Look for the following icon on the Eclipse toolbar and click on the down arrow. Select ‘Run Configurations…’



In the Run Configurations panel, select the ‘Store Application – Java Client’ then click on the ‘Duplicate’ toolbar button above it.


Run Configurations

Create, manage, and run configurations

Run a Java application

type filter te
Duplicates the currently selected launch configuration
guments
JRE
Generic Server(External Launch)
Gradle Task
Gradle Test
Grunt
Gulp
HTTP Preview
J2EE Preview
Java Applet
Java Application
Store Application - Java Client
JUnit

Name: Store Application - Java Client

Project: storeJava

Main class: ObRun.ObPanel.ObLaunch

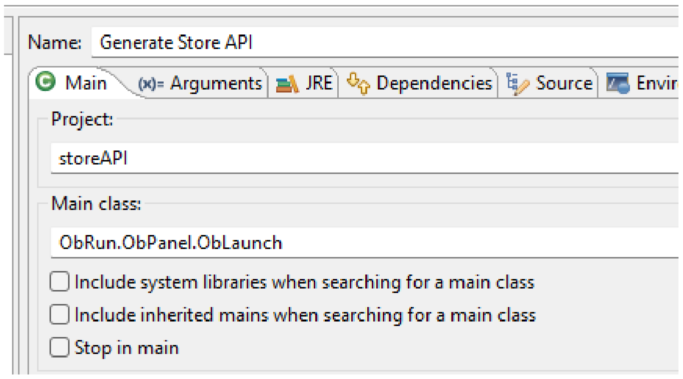
☐ Include system libraries when searchi

☐ Include inherited mains when search

☐ Stop in main

The Name field will show as “Store Application - Java Client (1)”. Rename it to “Generate Store API”.

Next, change the Project field from “storeJava” to “storeAPI”.



Name: Generate Store API

Project: storeAPI

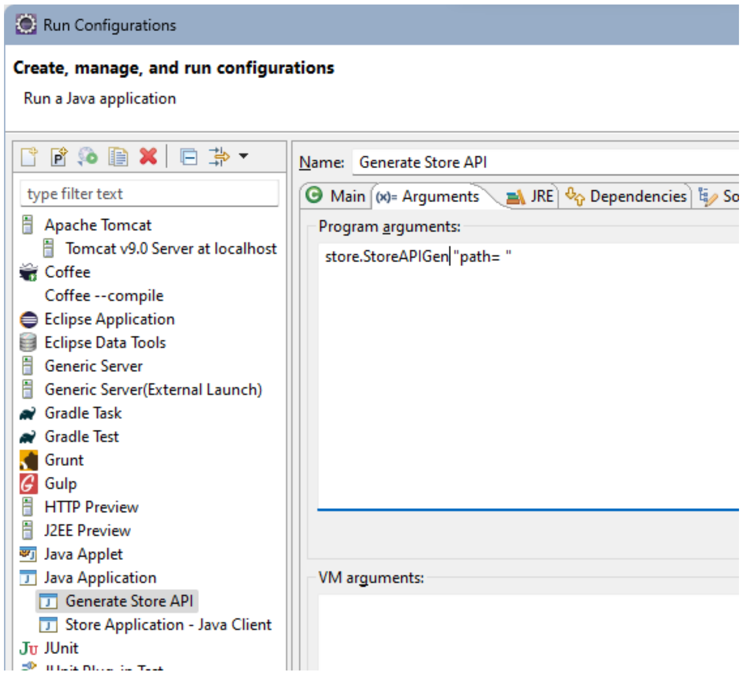
Main class: ObRun.ObPanel.ObLaunch

- [ ] Include system libraries when searching for a main class
- [ ] Include inherited mains when searching for a main class
- [ ] Stop in main

Next, click on the 'Arguments' tab and change the 'Program arguments' field to…

store.StoreAPIGen "path= "

This will call our plex function "StoreAPI.TemplateGeneration.TemplateGeneratorCommand". This function reads properties from the *storeAPI\properties\obclient.properties* file to determine the template and output file locations and other properties.



Run Configurations

Create, manage, and run configurations

Run a Java application

Name: Generate Store API

Program arguments:

```
store.StoreAPIGen "path= "
```

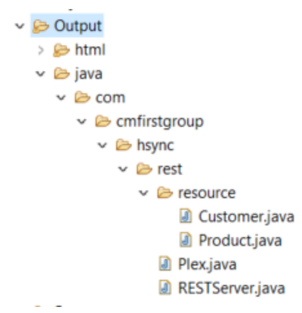
VM arguments:

You can now click the 'Run' button. The console at the bottom of the screen should show the following message:

```
Starting HSync Generator...
Reading from Template Folder: C:/Conference 2023 Workshops/Web
Technology/Workspace/storeAPI/Templates
Generating to Output Folder: C:/Conference 2023 Workshops/Web
Technology/Workspace/storeAPI/Output
5 files processed.  0 errors found.
```

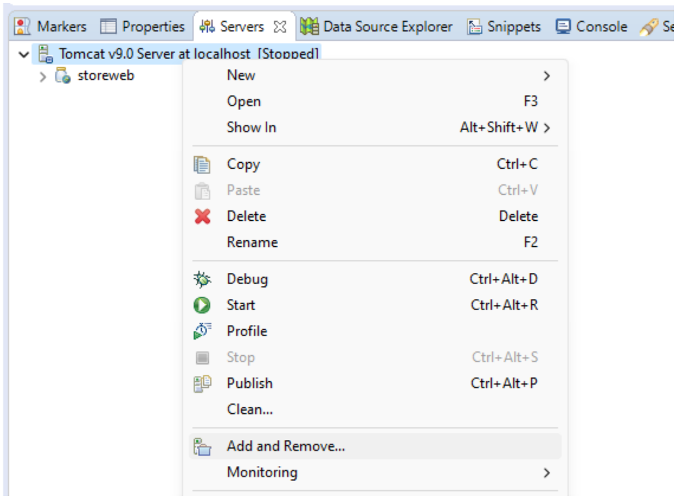
Refresh the workspace by selecting the 'storeAPI' project and pressing F5.

In the Project Explorer, you can find the generated API code under the storeAPI project.

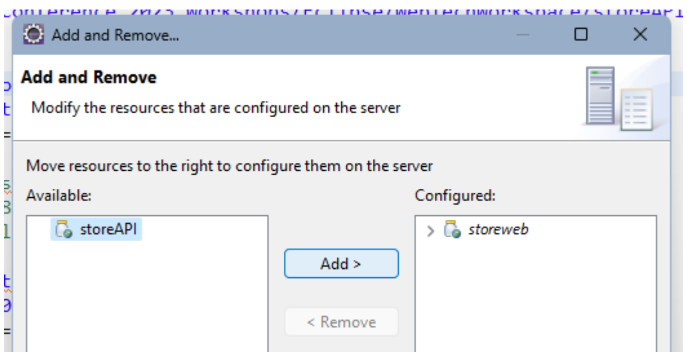


Publish the API

To publish the storeAPI project we need to add it to Tomcat on the *Servers* tab. Right-click on the Tomcat Server and select 'Add and Remove…'.



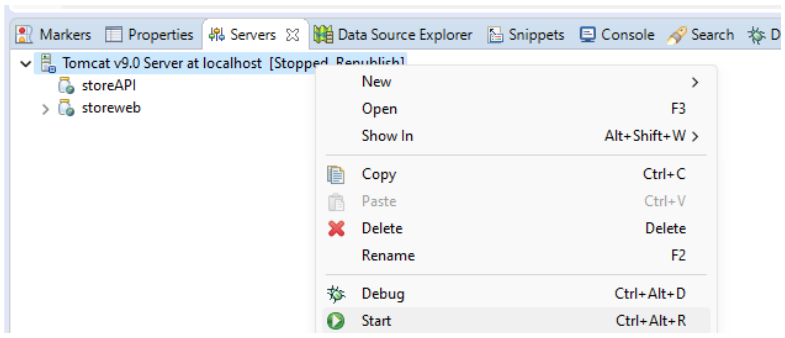
Select 'storeAPI' and press 'Add >' to add it to the publish list.


Add and Remove...

Add and Remove

Modify the resources that are configured on the server

Move resources to the right to configure them on the server

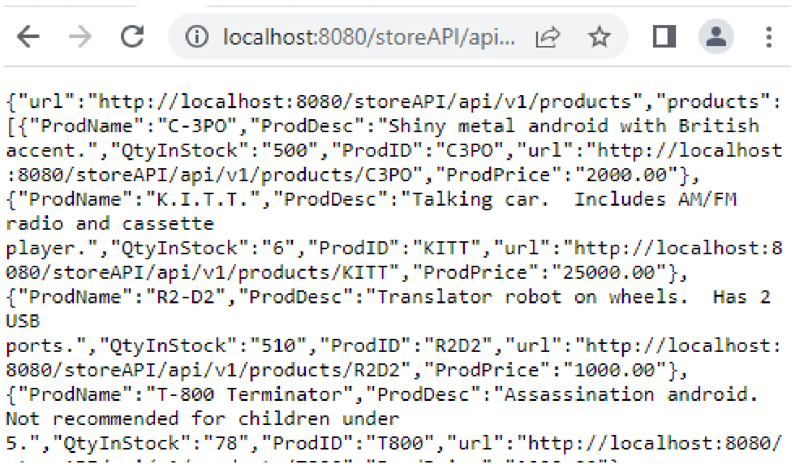


Publish and start the API by right-clicking on Tomcat and selecting ‘Start’. When the status shows as ‘Started’, it’s ready to Test.

As retrieval APIs use the ‘GET’ HTTP verb which is the verb used to request typical websites, we can preview the API in a browser by visiting the URLs below.

http://localhost:8080/storeAPI/api/v1/products

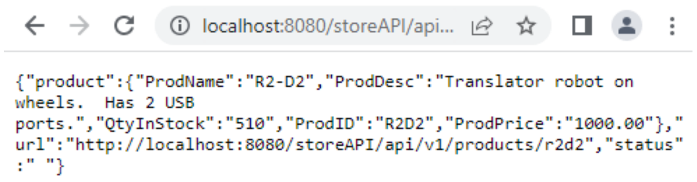
As no individual key is specified, this returns all products in a JSON array:



```
{"url":"http://localhost:8080/storeAPI/api/v1/products","products":
[{"ProdName":"C-3PO","ProdDesc":"Shiny metal android with British
accent.","QtyInStock":"500","ProdID":"C3PO","url":"http://localhost
:8080/storeAPI/api/v1/products/C3PO","ProdPrice":"2000.00"},
{"ProdName":"K.I.T.T.","ProdDesc":"Talking car.  Includes AM/FM
radio and cassette
player.","QtyInStock":"6","ProdID":"KITT","url":"http://localhost:8
080/storeAPI/api/v1/products/KITT","ProdPrice":"25000.00"},
{"ProdName":"R2-D2","ProdDesc":"Translator robot on wheels.  Has 2
USB
ports.","QtyInStock":"510","ProdID":"R2D2","url":"http://localhost:
8080/storeAPI/api/v1/products/R2D2","ProdPrice":"1000.00"},
{"ProdName":"T-800 Terminator","ProdDesc":"Assassination android.
Not recommended for children under
5.","QtyInStock":"78","ProdID":"T800","url":"http://localhost:8080/
```

http://localhost:8080/storeAPI/api/v1/products/r2d2

By specifying a key in the URL, the API will return a single record in JSON format.



```
{"product":{"ProdName":"R2-D2","ProdDesc":"Translator robot on
wheels.  Has 2 USB
ports.","QtyInStock":"510","ProdID":"R2D2","ProdPrice":"1000.00"},"
url":"http://localhost:8080/storeAPI/api/v1/products/r2d2","status"
:" "}
```

You can also try out the Customers API.

http://localhost:8080/storeAPI/api/v1/customers

http://localhost:8080/storeAPI/api/v1/customers/c1

We now have a way for external applications written on any platform to consume data and processes from our CA Plex application in a standard RESTful manner.

Exercise Eight – React with HSync API

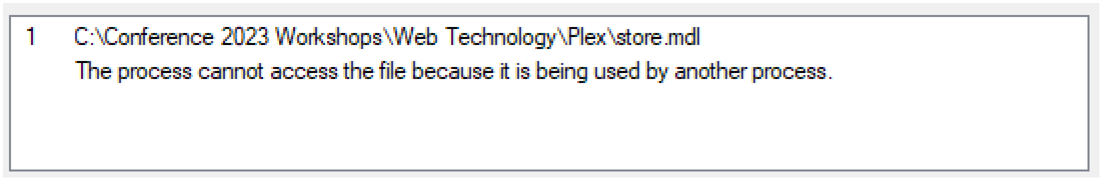
Restoring Snapshots

As there may not be time to complete all the exercises on time, a number of snapshots have been created to allow you to restore files before or after an exercise. These snapshots are 7-zip compressed files which contain the Plex and Eclipse workspace folders. An exercise may list the relevant snapshots to apply.

Here are the instructions to restore a snapshot.

Close Plex and Eclipse

Close the applications as they may lock files preventing them from being saved or restored. If something is left open you may see a message like this from 7-zip.



```
1    C:\Conference 2023 Workshops\Web Technology\Plex\store.mdl
     The process cannot access the file because it is being used by another process.
```

(Optional) Save your progress

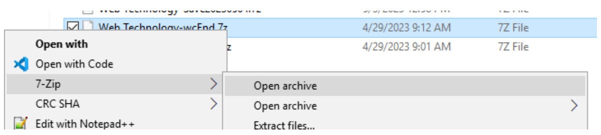
Right-click on the "Web Technology" folder inside the "C:\Conference 2023 Workshops" folder and select 7-zip -> Add to Web Technology.7z.

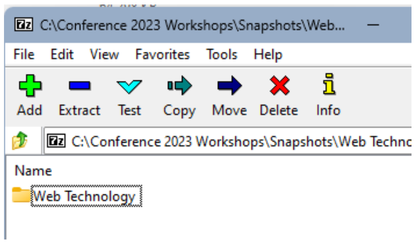
Delete the "Web Technology" folder

Select the folder and delete it.

Open your Snapshot

The exercise you're working on may direct you to restore a snapshot by name. Look in the "C:\Conference 2023 Workshops\Snapshots" folder and right-click on the file with that name, e.g. *Web Technology-wcStart.7z* and select 'Open Archive'. This will open the snapshot in 7-zip.




C:\Conference 2023 Workshops\Snapshots\Web...
File
Edit
View
Favorites
Tools
Help
Add
Extract
Test
Copy
Move
Delete
Info
C:\Conference 2023 Workshops\Snapshots\Web Techno
Name
Web Technology

Restore the Snapshot

Go back to the "C:\Conference 2023 Workshops" folder and drag the Web Technology folder from 7-zip into that folder. It should take a few seconds to restore everything.