Welcome to SAP Montreal
City-Hacking with SAP Co-innovation & Development

May 22 & 23, 2013,
SAP Montreal

TABLE OF CONTENTS:

INSTALLATION PREREQUISITES

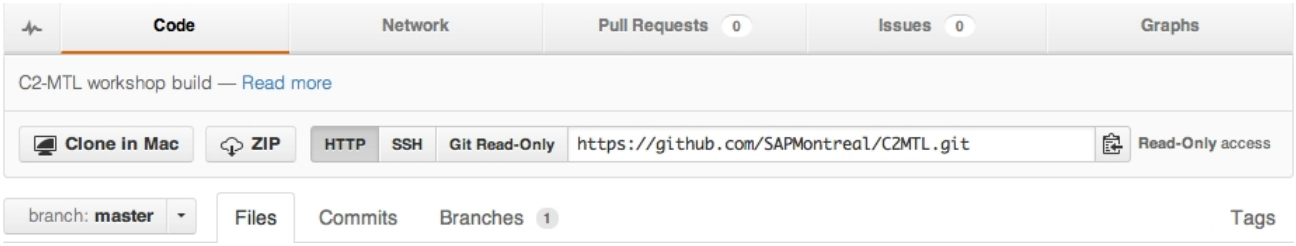
Before you can start with these exercises, you will have to have the following components installed on your local machine:

- Eclipse
- Java 1.6 (no later versions)
- Tomcat
- UI5 Development toolkit for HTML5
- Google Chrome
- Github connectors

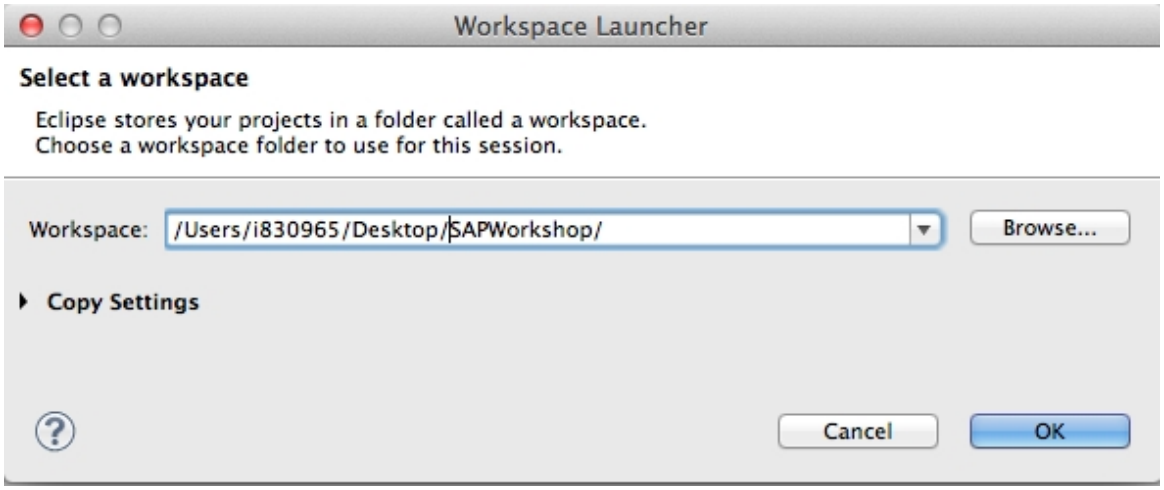
Once you have installed them, you will need to configure the UI5 development kit for Eclipse.

HOW TO IMPORT THE APPLICATION INTO ECLIPSE

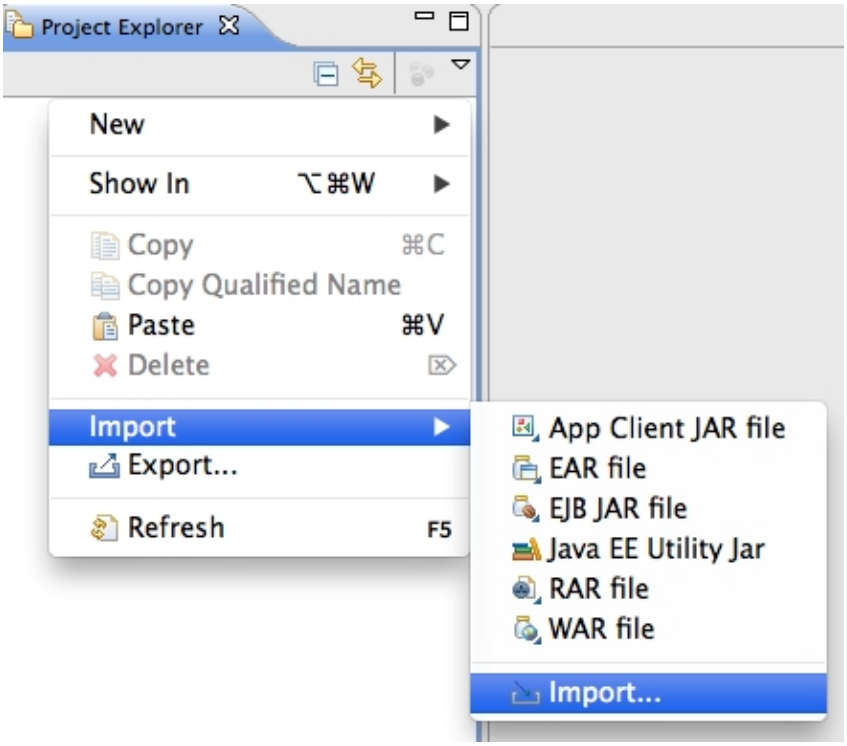
1. Go to https://github.com/SAPMontreal/C2MTL and click **ZIP**. This will download the application code to your computer. Make a note of where you downloaded it.



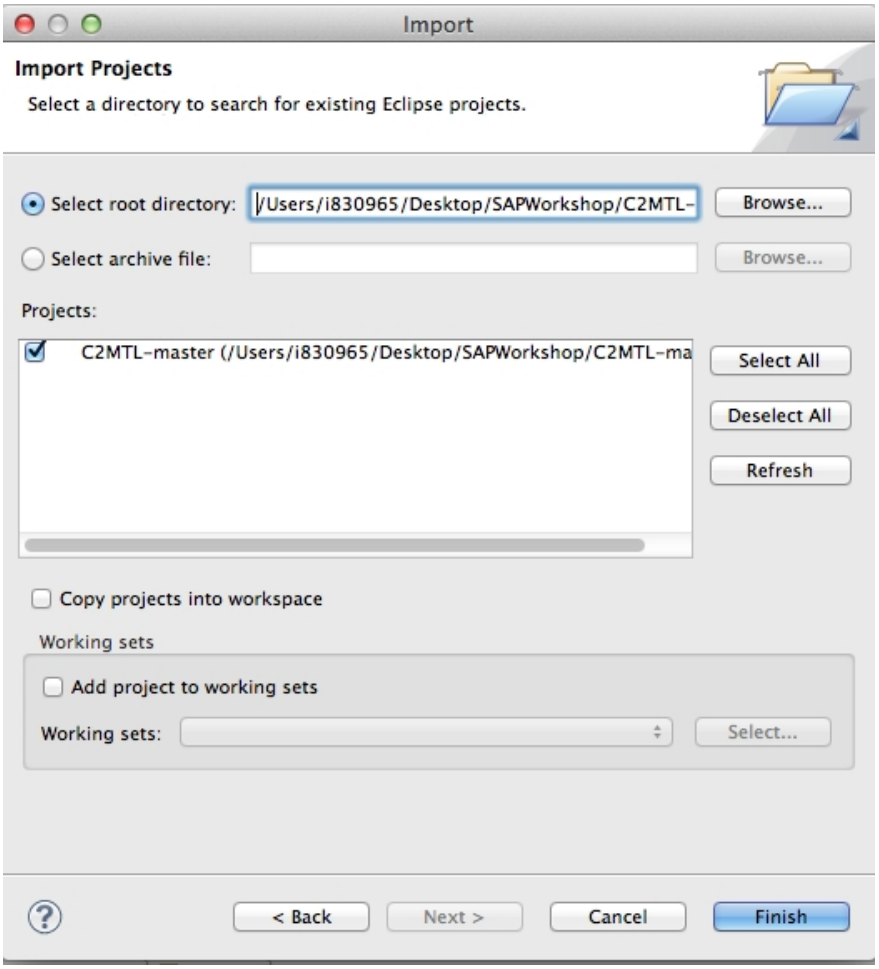
2. Open **Eclipse** and choose a directory for your workspace (where you will be working). Remember the directory since we will need to put the code there. After you pick a directory, Click **OK**.
3. Return to the directory where you saved the app from GitHub. Extract the application to the workspace you created with Eclipse. In this example, it is **/Users/i830965/Desktop/SAPWorkshop**



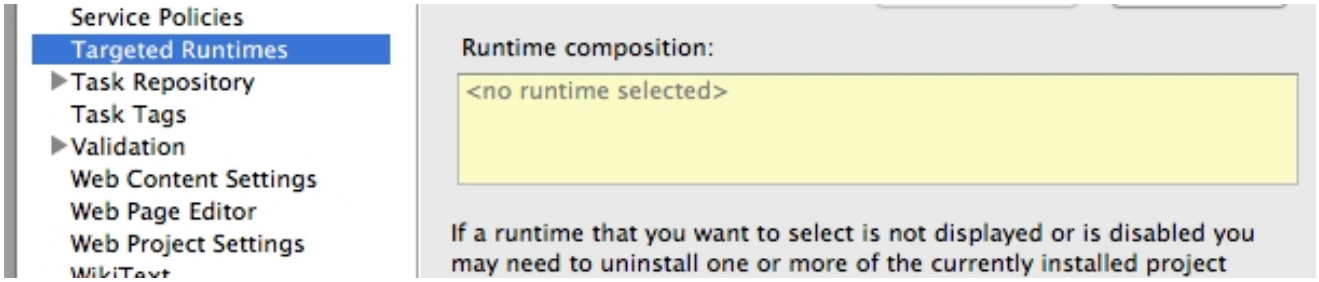
4. Import the application project into Eclipse. Return to Eclipse and right-click in the Project Explorer and select **Import ->Import**



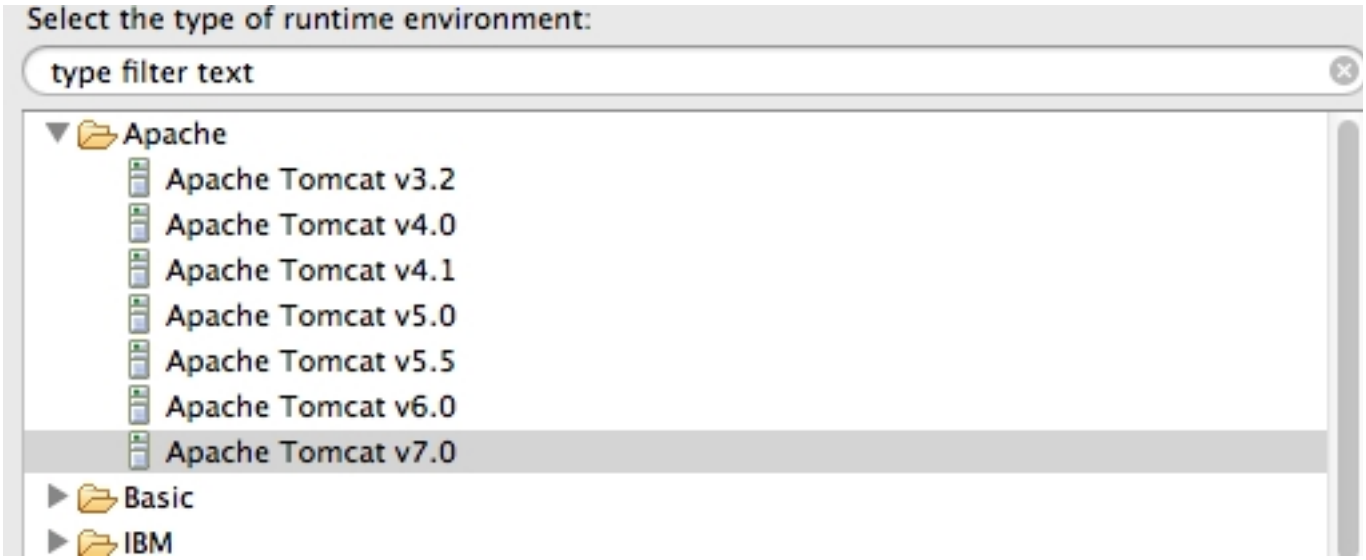
5. Under the **General** folder select **Existing Projects into Workspace** and click **Next**.
6. Click **Browse** and select the extracted folder from Step 3.
7. Once you've done this click **Finish**.



8. Fix the project properties. To do this, inside the project explorer right-click the project and select **Properties**. The Properties window appears.

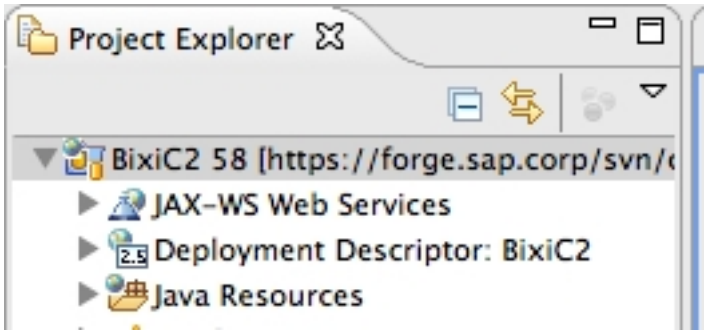


9. Add a targeted run time. Search for **Targeted Runtimes** in the search bar, or select it from the left hand side.
10. Select **Apache Tomcat v7.0** runtime under Apache and click **Finish**.

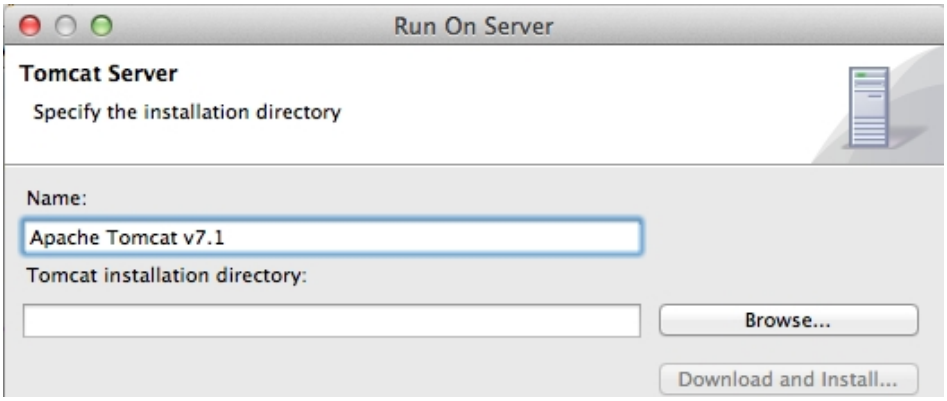

Select the type of runtime environment:
type filter text
Apache
Apache Tomcat v3.2
Apache Tomcat v4.0
Apache Tomcat v4.1
Apache Tomcat v5.0
Apache Tomcat v5.5
Apache Tomcat v6.0
Apache Tomcat v7.0
Basic
IBM

HOW TO RUN THE APPLICATION FOR THE FIRST TIME

1. Open **Eclipse**.
2. In Eclipse, right-click on the project **BixiC2** in the Project Explorer.



3. Select **Run As -> Run on Server**.
4. Open the folder named Apache and select **Tomcat 7** (or whichever version of Tomcat you downloaded.)
5. Click **Browse** and navigate to the folder where you extracted the Tomcat download and select it.



6. Click **Next/Finish**. The app will start and open a small browser in Eclipse.

   NOTE: Do **not** use this Eclipse browser as it does not support HTML5 very well. For the purposes of these exercises, we will use **Google Chrome** exclusively. If you plan to repeat these exercises on your own machine later, do the following:
   a) Open Google Chrome, and under **Settings**, make it your default browser.
   b) Open Eclipse and choose **Window -> Web Browser** and make sure **Default system web browser is selected**.

7. Open Chrome and go to this URL: http://localhost:8080/BixiC2.

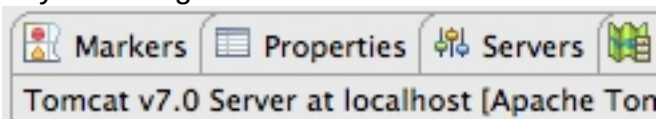
HOW TO RE-RUN THE APPLICATION AFTER MAKING CHANGES

Because you will make frequent changes to the code in the course of the workshop, you will have to re-run the app. This section explains how.

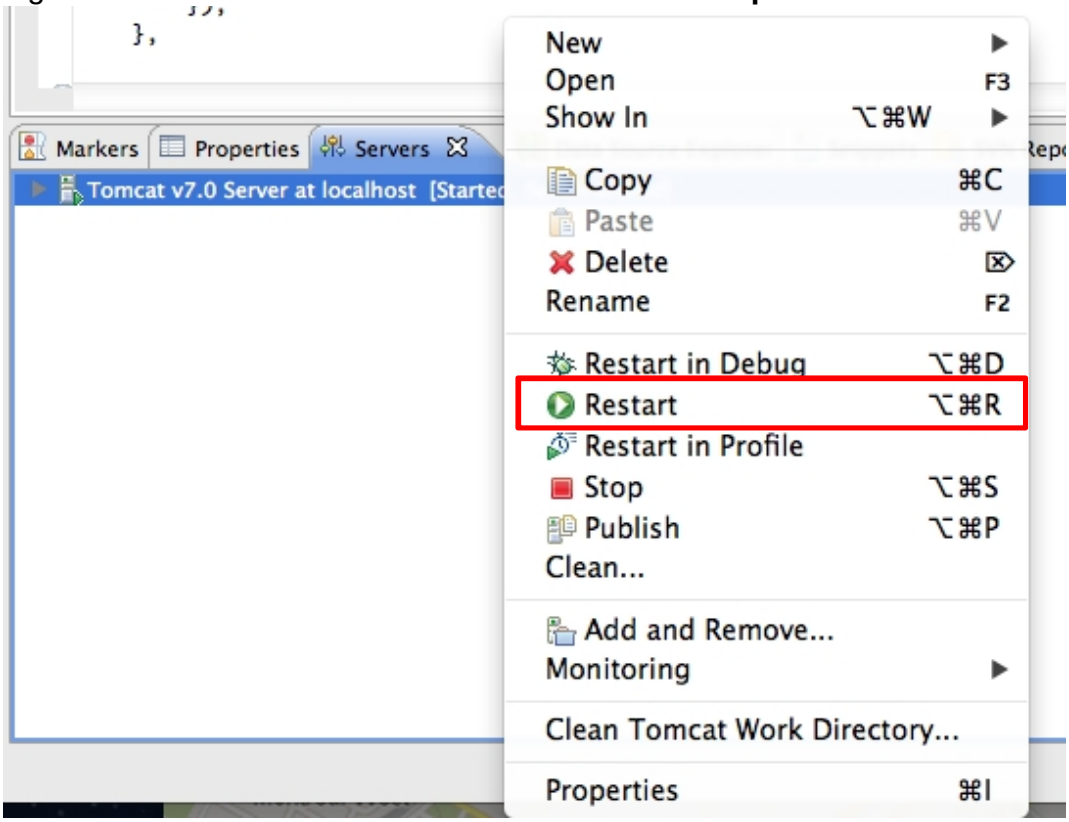
NOTE: If you have left the Eclipse instance running since the last time you ran the app, you can skip to the next section. If you closed it, complete the following steps.

As soon as you save changes to a file, Eclipse automatically republishes the file to the Web-server, Tomcat. If you need to republish data to your Web-server, you can do so manually.

1. Look for the **Servers** tab near the bottom of Eclipse (if you can't find it you can display it by choosing **Window -> Show View -> Servers**):



2. Right-click the Tomcat server in the list and select **Republish** or **Restart**.



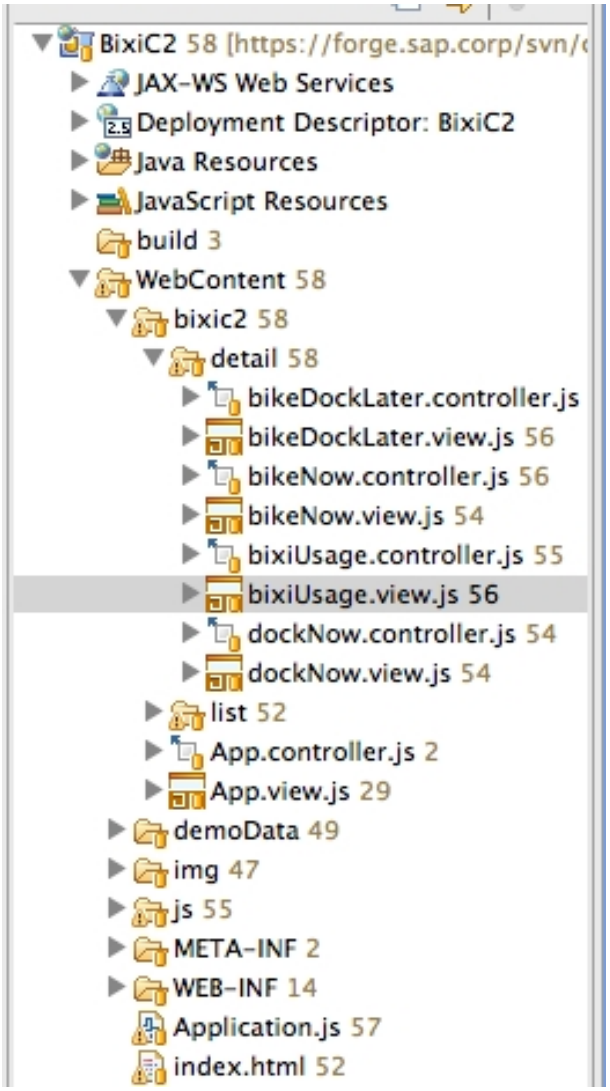
3. Wait a few seconds and then go to the URL http://localhost:8080/BixiC2.

HOW TO IMPLEMENT TOUCH EVENTS ON CHARTS

Like most SAPUI5 controls, MAkit charts allow us to add events to show additional information. In this exercise, you will modify some source code by implementing an event to display the current time period.

Imagine the following scenario: The user can see the Bixi usage for an entire day but he is not quite sure what time the point on the graph occurs at.
We can add an event to the chart so when the user double-taps a point on the line, he receives a message that displays the time the point of data was recorded at.

1. Navigate to the bikeUsage.view.js file inside Project Explorer inside Eclipse.
   Choose **BixiC2 -> WebContent -> bixic2 -> detail -> bixiUsage.view.js**



2. Double-click the file to open the Javascript Editor.
3. Scroll down until you see the function **createLineChart : function(oModel) { }**.
4. You will find a 'TODO' comment. Some lazy programmer has forgotten to do his work. Here is your chance to make it better.
5. You must add an Event Handler to **oChart** for the "doubletap" event. You can do this by using the attach event function.

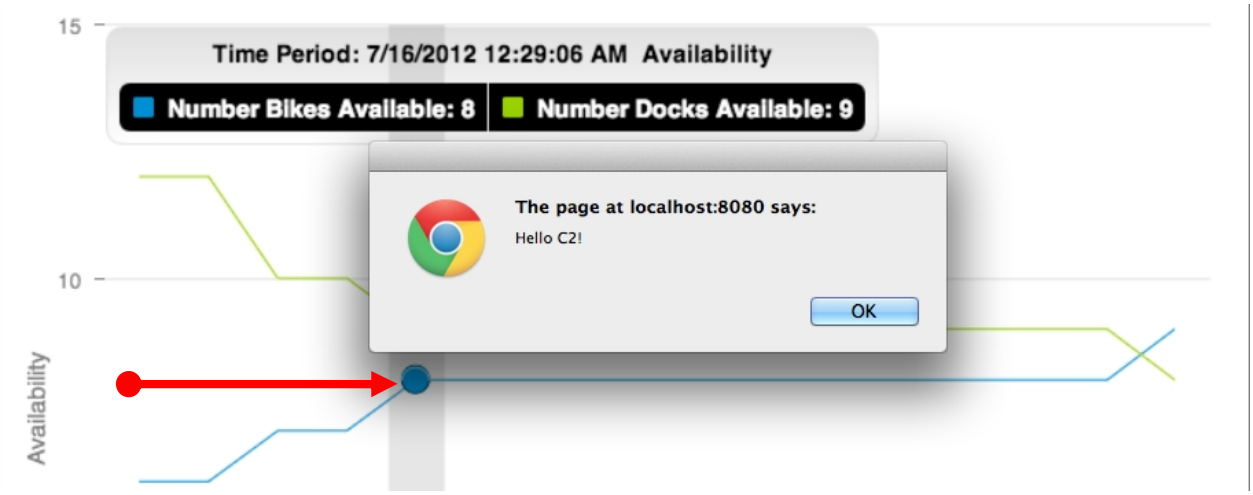
```
        // TODO: Implement Touch Events for oChart in the workshop!

        return oChart;
    },
```

6. Add the following code below to the TODO:

```
oChart.attachEvent("doubletap", function(eventObject) { alert("Hello C2!");
} );
```

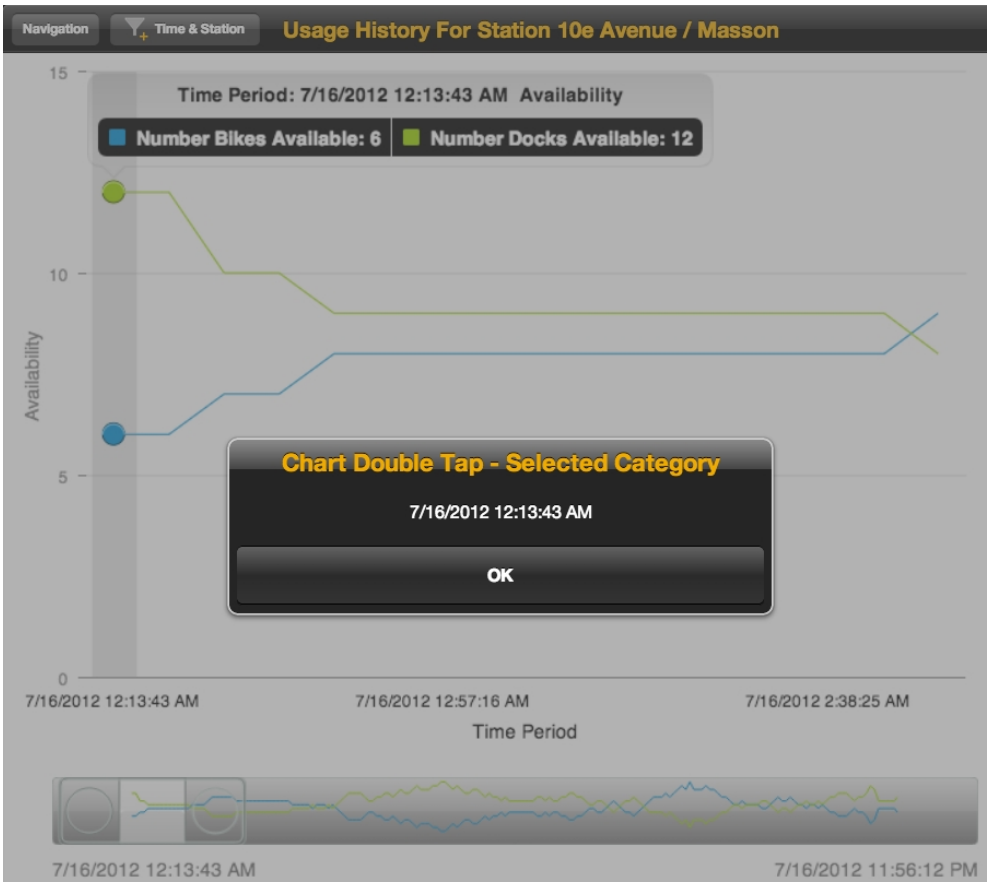
7. Save the file. Now let's test it.
8. Open Chrome and go to the URL http://localhost:8080/BixiC2.
9. Navigate to the **Bixi Usage** view, filter the data and double click on the little **green dot** or the **blue dot**. You should see a dialogue that says **Hello C2!**



10. It doesn't really help our users. The SAPUI5 developers have created something nicer, the **sap.m.MessageBox** control. It's up to us however to use it to display some meaningful information about our Bixi bikes.
11. Remove the code you added in step 6 and replace it with the following:

```
oChart.attachEvent("doubletap", function(eventObj) {
      jQuery.sap.require("sap.m.MessageBox");
      var chartObj = eventObj.oSource;
      var msg = eventObj.oSource.getSelectedCategory();
      sap.m.MessageBox.show(msg, sap.m.MessageBox.Icon.INFORMATION,
            "Chart Double Tap - Selected Category",
            [ sap.m.MessageBox.Action.OK ]);
});
```

12. Once you've finished inserting the above code, repeat steps 8 and 9 and you should see something like this:



13. You've completed this exercise, now move on to the next one.

HOW TO CHANGE A GRAPH VIEW TO DISPLAY TABULAR DATA

Imagine the situation where your user, instead of wanting to see a Bar Chart, would rather see a table of data. We can do this easily with SAPUI5 and MAkit.

1. In Eclipse, inside the Project Explorer locate the **bikeDockLater.view.js** file under this path: **WebContent -> bixic2 ->detail**.
2. Double-click the file to open the Javascript Editor.
3. Scroll down until you see the `createBarChart : function(oModel) {}` function.

```
/**
 * Creates and returns the bar chart thats bound to
 *
 * @returns {sap.makit.Chart} a bar chart
 */
createBarChart : function(oModel) {

    var oChart = new sap.makit.Chart({
        width : "100%",
        height : "80%",
        type : sap.makit.ChartType.Column,
        showRangeSelector : true,
        showTableView : false,
        valueAxis : new sap.makit.ValueAxis({
        }),
```

4. Inside the chart constructor locate the type of chart to be created:
   `type : sap.makit.ChartType.Column`

5. Just below this, you will see a configuration option called: `showTableView : false,`
   Change this value to **`true`** and rerun the app to see what happens. Instead of a bar chart view, you will see a table.

That's cool, but what if you want to have a button to switch back and forth betrween data display types? Try asking your SAP buddy to show you how to do this. Or, if you would rather continue to the other exercises, skip this and move on.

HOW TO ADD CIRCLES FOR BICYCLES TO THE I NEED A BIKE NOW VIEW

We already have a view that displays circles of data for each Bixi station for the available docks, but what if the user needs a bike instead? We've done most of the work, you just need to draw the circles.

To do this we will be using the Google Maps API version 3.

1. Open Eclipse and inside the Project Explorer navigate to the **bikeNow.controller.js** file under this path: **WebContent -> bixic2 -> detail**
2. Double-click the file to open it in the Javascript Editor.
3. First you need an object that will contain the configuration options for our circles. To do this copy and paste this code below to the TODO called **Draw Circles**. This is located inside the function `drawDataCircles: function(oMap) {}`.

```
var oPopOptions = {
      clickable: true,
      strokeOpacity: 0.8,
      strokeWeight: 2,
      fillOpacity: 0.35,
      map: oMap,
      center: new google.maps.LatLng(fLat, fLong),
};
```

4. Next you must decide the circle color and the circle size. Stations with less than 3 bikes will be red, stations with more than 3 docks others will be green. They will grow in size based on a radius that is calculated. To do this, copy and paste this sample code right below where you pasted the previous code:

```
if (bCanGetABike) {
       oPopOptions.strokeColor = '#36D792';
       oPopOptions.fillColor = '#36D792';
       oPopOptions.radius = iRadius;
} else {
       oPopOptions.strokeColor = '#FF0000';
       oPopOptions.fillColor = '#FF0000';
       oPopOptions.radius = 100;
}
```

5. Now all you have to do is create the circle and add it to our list of circles, **aCircles**. To do this, copy the code below and paste it under the previous section.

```
var circle = new google.maps.Circle(oPopOptions);
aCircles.push({
      oCircle: circle,
      name : sName,
      nbBikes: iNumFreeBikes
});
```

6. That’s it! You should now rerun the app and visit the “I Need a Bike Now!” view to see the results. Great Job!