Nativescript-workshop

Table of contents

Environment configuration

To be able to complete this workshop the following are required:

- Nativescript framework - http://docs.nativescript.org/start/quick-setup
- Akera.io - ?
- Progress AppServer 11.6

Creating and running the nativescript project

1.Open your terminal, navigate to your workspace and execute the command :

tns create nativescript-workshop

2.Navigate into the newly created project and add the target development platform:

If you're on a Mac, start by adding the iOS platform:

tns platform add ios

Next, add the Android platform with the same platform add command:

tns platform add android

***Sometimes the quickest problem solver is **tns remove && tns add 'platform name'**.

3.You can now run your app in an emulator or on devices:

If you're on a Mac, start by running the app in an iOS simulator with the following command:

tns run ios –-emulator

Next, run your app on an Android emulator with the following command:

tns run android -–emulator

If you want to run your app on a connected device use the following commands:

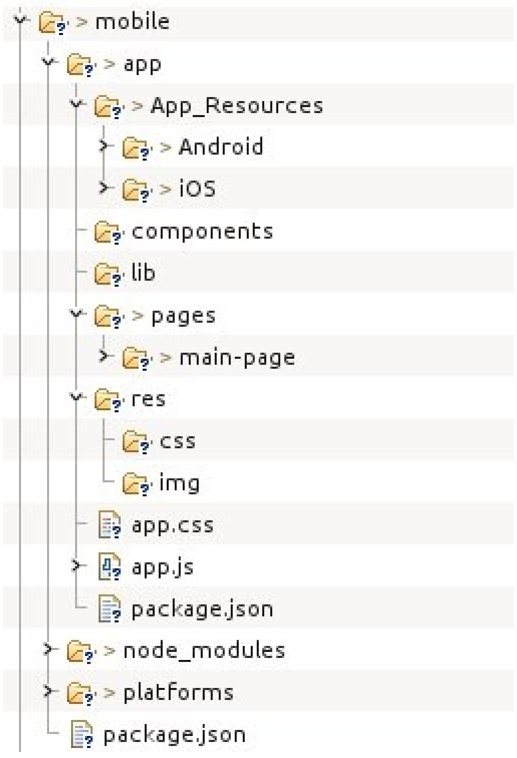
tns device - which will give you a list of all the connectd devices.

tns run 'platform name' --device 'device id' - will deploy and run the app on the specified device.

Directory structure

- Delete the **main-page.xml**, **main-page.js**, **main-view-model.js** and **reference.d.ts** files.
- Copy the contents of **structure1.zip** in the app folder.

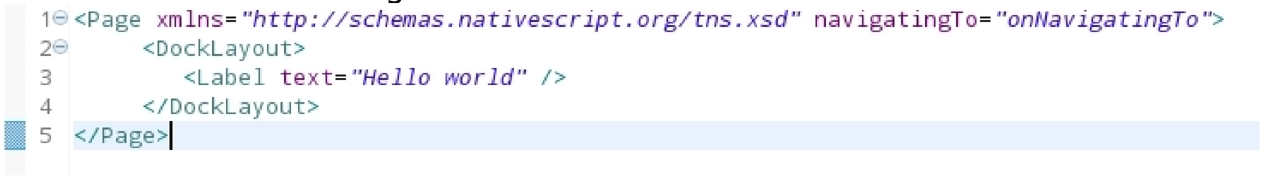
Now our project structure should look like this:



```
> mobile
  > app
    > App_Resources
      > Android
      > iOS
    components
    lib
    > pages
      > main-page
    res
      css
      img
    app.css
    app.js
    package.json
  > node_modules
  > platforms
  package.json
```

Adding ui components

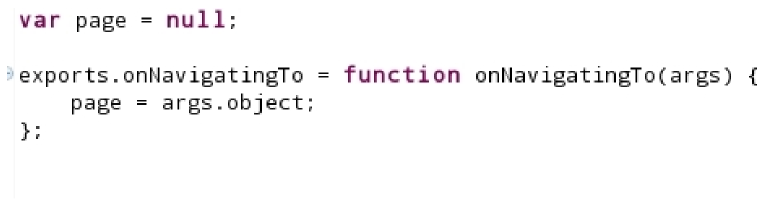
When we open **main-page.xml** located in **app/pages/main-page/** we should see the following code:



```
<Page xmlns="http://schemas.nativescript.org/tns.xsd" navigatingTo="onNavigatingTo">
    <DockLayout>
        <Label text="Hello world" />
    </DockLayout>
</Page>
```

This page currently contains three UI components a **<Page>**, a layout **<DockLayout>** and a **<Label>** .

The logic of the main-page is located in a **main-page.js** or a **main-page.ts** file:



```
var page = null;

exports.onNavigatingTo = function onNavigatingTo(args) {
    page = args.object;
};
```

We will now create the main-page of our application:

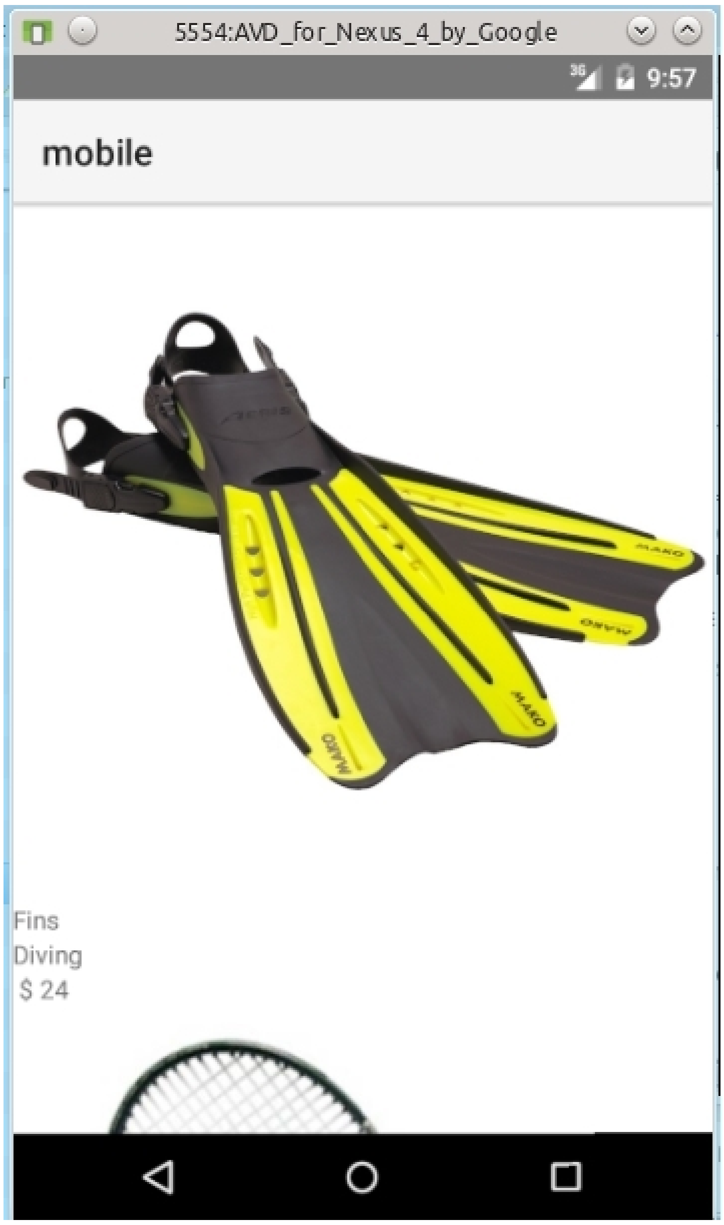
1.For the UI part we will add a **<Repeater>** that will display products from the sports2000 database:

```
       <Page xmlns="http://schemas.nativescript.org/tns.xsd"
navigatingTo="onNavigatingTo">
        <DockLayout>
          <ScrollView dock="bottom">
              <Repeater id="productsList">
               <Repeater.itemTemplate>
                  <DockLayout>
                    <Image dock="top" src="{{ imgSrc }}" stretch="aspectFit" />
                    <StackLayout dock="bottom">
                           <Label text="{{ itemname || 'Downloading...' }}" />
                           <Label text="{{ category1 || 'Downloading...' }}" />
                           <Label text="{{ ' $ ' + price || 'Downloading...' }}" />

                    </StackLayout>
                  </DockLayout>
                 </Repeater.itemTemplate>
            </Repeater>
          </ScrollView>
         </DockLayout>
       </Page>
```

2.And for the Bussiness logic we will add the following:

```
//Using http methods requires to load "http" module.
var http = require("http");

var page = null;
var url = http://192.168.1.220:3000;  //the server url

var productsList = null;
var products = [];

exports.onNavigatingTo = function onNavigatingTo(args) {
    page = args.object;

    //get the ui repeater component 'Product List'
    productsList = page.getViewById("productsList");

   //set the product list items
    productsList.items = products

   //async load the products
    if (products.length === 0)
           loadItems();
};

function loadItems() {
    http.getJSON(url + '/api/Items').then(function(rsp) {
        for (var key in rsp) {
            var prod = rsp[key];

       //set each product image source
            prod.imgSrc = url + '/res/img/products/' + prod.itemnum + '.jpg';

       //add the product to the list
            products.push(prod);
        }

        //manually refresh the list
        productsList.refresh();
    }, function(err) {

 //print the error to the console
        console.log(err.message);
 //show the error in an alert box
        alert(err.message);

    });
}
```

The result should look like this:



Fins
Diving
$ 24

Styling

1.In the app.css located in the app folder we will add the following css properties:

```
ActionBar {
        background-color:#1995dc;
        color: white;
}

.hr {
        height:1;
        background-color:LightGray;
}
```

2.Next we need to style the main-page .In order to do this we need to create the main-page.css file in /res/css/ folder and add the following css properties:

```
.prodImg {
        width:100%;
        height:200;
        padding:20;
}


.prodTitle {
        color:black;
        margin-left:15;
        font-size:25;
        font-weight:bold;
}


.prodCateg {
        margin-left:15;
        font-size:18;
}


.prodPrice {
        margin-left:15;
        margin-bottom:15;
        font-size:18;
}
```

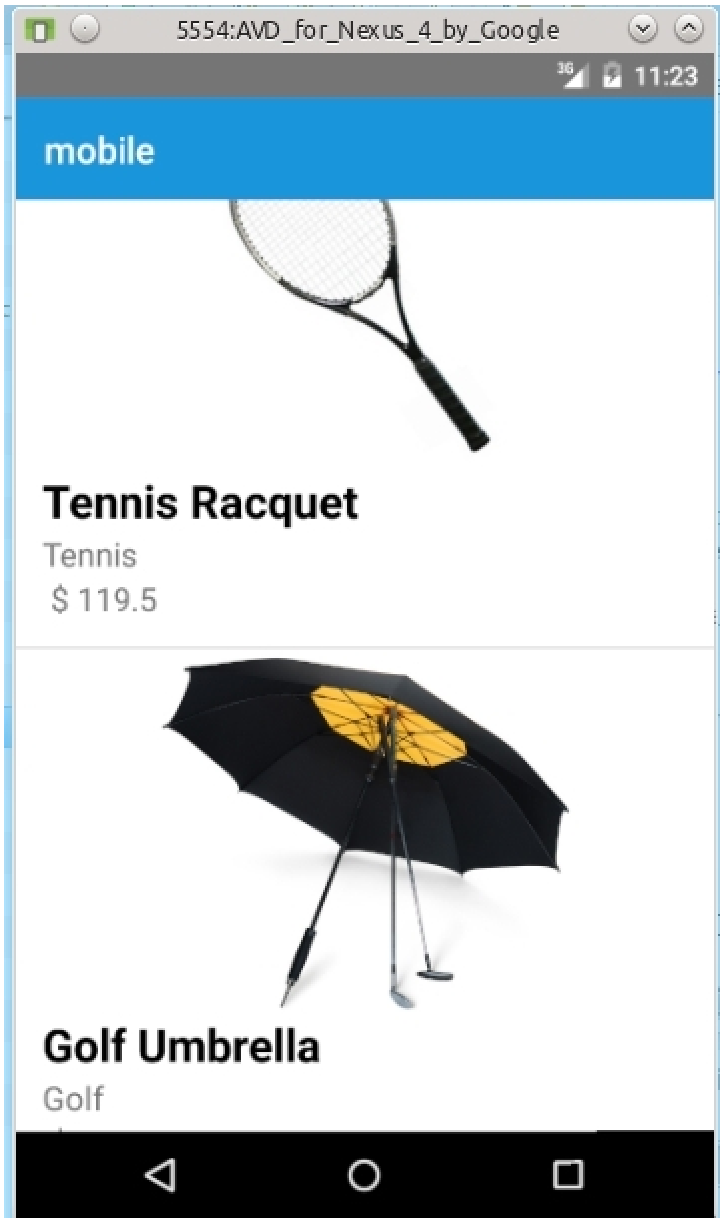
3.In the main-page.js file we apply the css file by using the following method:

```
page.addCssFile('~/res/css/main-page.css');
```

4.And for main-page.xml :

```
<DockLayout>
   <Image dock="top" src="{{ imgSrc }}" stretch="aspectFit" class="prodImg" />
      <StackLayout dock="bottom">
         <Label text="{{ itemname || 'Downloading...' }}" class="prodTitle" />

         <Label text="{{ category1 || 'Downloading...' }}" class="prodCateg" />

         <Label text="{{ ' $ ' + price || 'Downloading...' }}" class="prodPrice" />

         <StackLayout class="hr" />
      </StackLayout>
</DockLayout>
```

The app should look a bit better than the last time:



Events and navigation

In order to see the product details on item tap we need to create a event handler for that action that will navigate to the product-details page passing the product data as a navigation context:

1.Add the xml declaration:

<DockLayout tap="showDetails">

2.The code-behind:

Required modules:

```
var frameModule = require("ui/frame");
var topmost = frameModule.topmost();
```

ShowDetails method:

```
exports.showDetails = function(args) {
var data = args.object.bindingContext;
var navigationEntry = {
  moduleName: "pages/product-details/product-details",
      context: {
      productData: data
  },
      animated: true
  };
topmost.navigate(navigationEntry);
};
```

3.The product-details page:

- Create the folder **/pages/product-details**
- Navigate to **/pages/product-details** and add create the **product-details.xml** file and add the following xml code:

```
<Page xmlns="http://schemas.nativescript.org/tns.xsd" navigatingTo="onNavigatingTo">
<ScrollView>
    <DockLayout>
        <Image dock="top" src="{{ imgSrc }}" stretch="aspectFit" class="prodImg" />
        <StackLayout dock="bottom">
            <StackLayout class="hr" />
            <Label text="{{ catdescription || 'Downloading...' }}"
class="prodDesc"textWrap="true" />
        </StackLayout>
     </DockLayout>
</ScrollView>
</Page>
```

And for **product-details.js** :

```
var observable = require("data/observable");
var viewModel = new observable.Observable();

var page = null;
var productData = null;

exports.onNavigatingTo = function onNavigatingTo(args) {
    page = args.object; //add the css file

    page.addCssFile('~/res/css/product-details.css');

    //get the navigation context
    productData = page.navigationContext.productData;

    //set the page title
    page.actionBar.title = productData.itemname;

    viewModel.set("imgSrc", productData.imgSrc);
    viewModel.set("catdescription", productData.catdescription);

    page.bindingContext = viewModel;
}
```

Next create **product-details.css** file in **res/css/** folder and add the following styles:

```
.prodImg {
        width:100%;
        height:200;
        padding:20;
}


.prodDesc {
        font-size:20;
        padding:20;
}
```