

ICENIUM Hands On Lab

Hybrid Mobile Application Development with Telerik's Icenium & Kendo UI
Mobile

For Limited Circulation Only

Version Information:

Version No.	Release Date	Author(s)	Reviewer(s)
1.0	Jan 28 th 2013	Dhananjay Kumar	Lohith Nagaraj
1.1	Feb 8 th 2013	Abhishek Kant	Anthony Abdullah

Lab Manual for creating a Cross Platform Application using Icenium and KendoUI Mobile

Contents

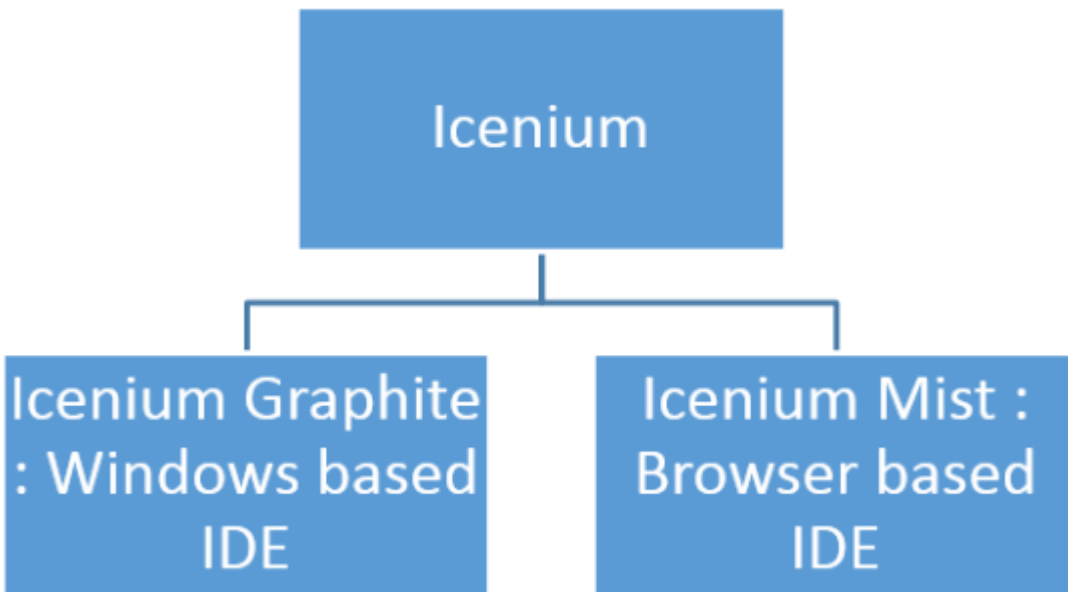
Lab 1: Creating Your First Cross-Platform Application using Kendo UI Mobile and Icenium.....	4
Objective	4
Conclusion	10
Lab 2: Exploring Icenium features for Hybrid Application Development	11
Objective	11
Code Editor.....	11
Certificate Management.....	13
Version Control.....	14
Integration with github Repository	17
Code signing ,Permissions and Publishing the application	24
Conclusion.....	25
Lab 3: Creating a Cross-Platform “Twitter Search” Application using Kendo UI Mobile and Icenium.....	26
Objective	26
Step 1: Create Project	27
Step 2: Create Layout of the Application	29
Step 3: Create Settings View	31
Step 4: Create Tweets View	32
Step 5: Create Template to display tweets	32
Step 6: Create Data Source to Fetch Tweets.....	33
Lab 4: Creating Netflix Movie Explorer.....	36
Objective	36
Step 1: Create Project	37
Step 2: Create Layout of the Application	37
Step 3: Create Views	38

Setting layout of the application	38
Setting transition style of the application.....	39
Step 4: Create DataSource	42
Step 4: Create Template	42
Step 5: Create ListView	44
Run Application	44
Create Movie Detail View with dynamic navigation.....	45
Running the Application.....	49
Conclusion.....	50
Lab 5: Create APK package for Google Play using Icenium	51

Icenium is Telerik's advancement of the traditional IDE. Icenium is an Integrated Cloud Environment (ICE) that helps you easily build Hybrid Mobile Applications by combining the convenience of a local development environment with the power and flexibility of the cloud. If you are new to Hybrid Mobile Application, then it is suggested that you go through the following articles:

- [What is a Hybrid Mobile App?](#)
- [As a Developer why should I worry about Hybrid Application Development?](#)

Icenium provides two different types of development environments. Following picture depicts the IDE options available:



Lab 1: Creating Your First Cross-Platform Application using Kendo UI Mobile and Icenium

Objective

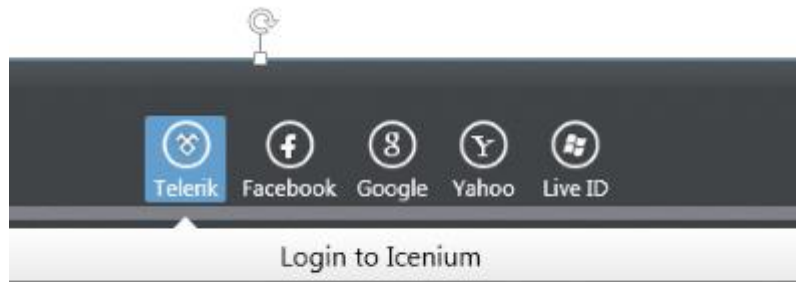
This lab will help you to create first Cross Platform Application using Icenium

In this lab we are going to use Icenium Graphite which is a desktop client to create our first Cross-platform application. To start the development, launch Icenium Graphite. You will be asked to Login to Icenium. You can login using any of the following providers:

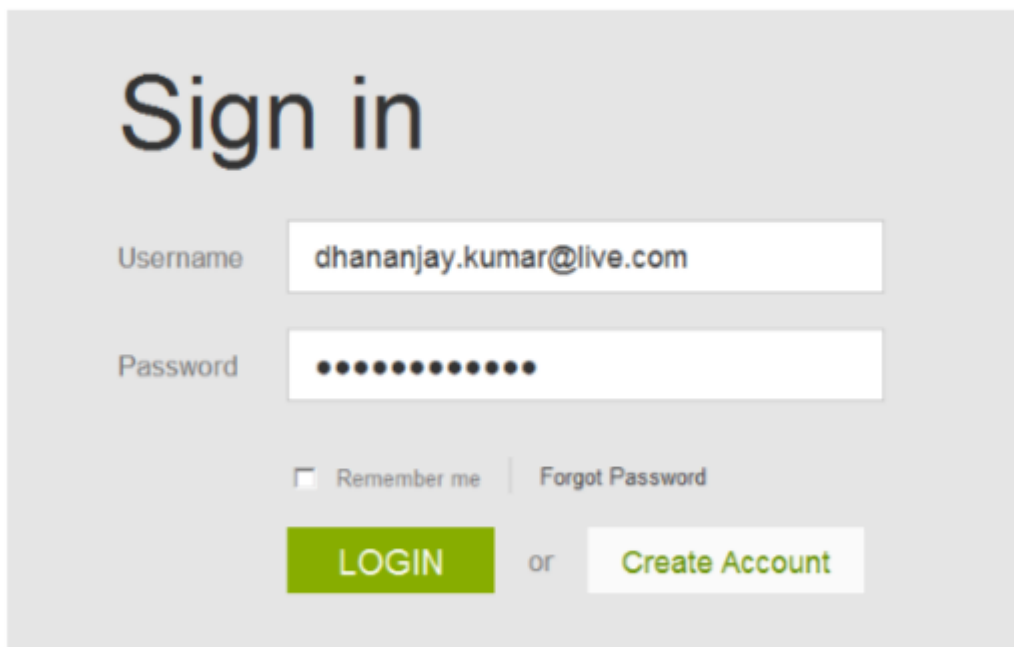
- Telerik Account
- Facebook Account
- Google Account
- Yahoo Account

For Limited Circulation Only

- Microsoft Live Account



Let us choose Telerik account for logging into Icenium.

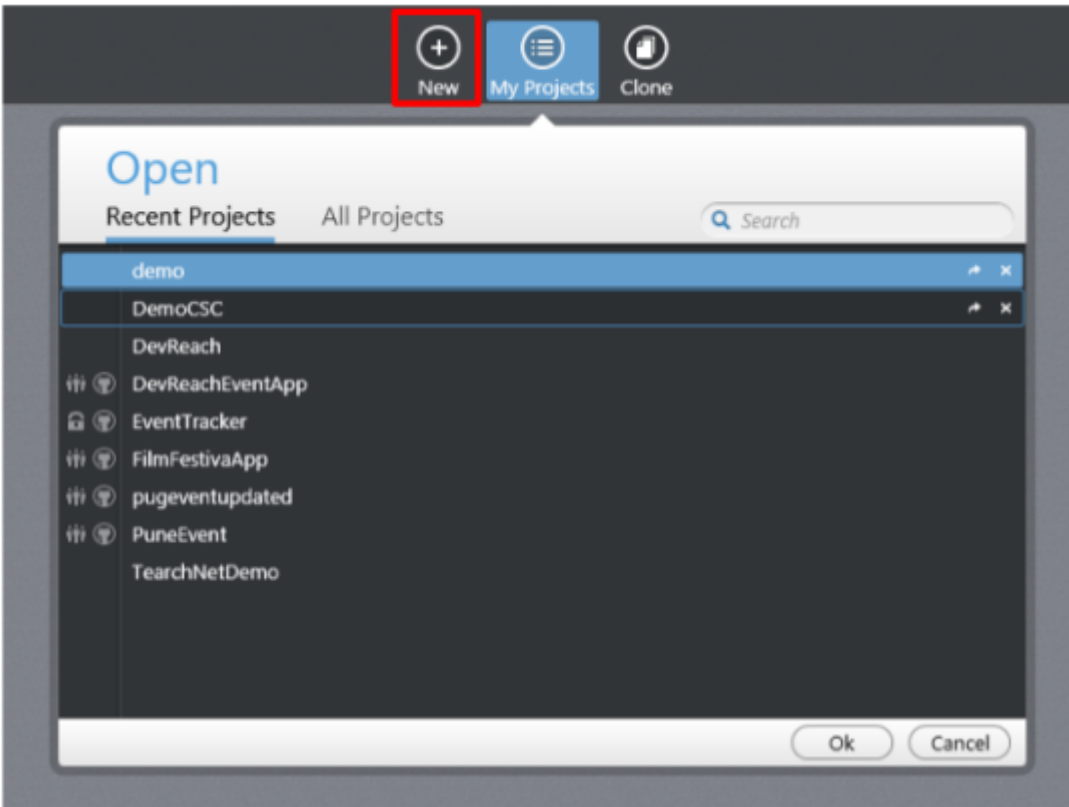
A screenshot of the Icenium "Sign in" form. The form is set against a light grey background. At the top, the text "Sign in" is displayed in a large, dark blue font. Below this, there are two input fields: "Username" with the value "dhananjay.kumar@live.com" and "Password" with a masked password represented by ten black dots. Under the password field, there is a checkbox labeled "Remember me" and a link labeled "Forgot Password". At the bottom of the form, there are two buttons: a green "LOGIN" button and a white "Create Account" button, separated by the word "or".

After successful login you will get a Project dashboard. On project dashboard you can perform following three tasks:

- New - Create New Project
- My Projects - Browse existing Projects
- Clone - Clone a project

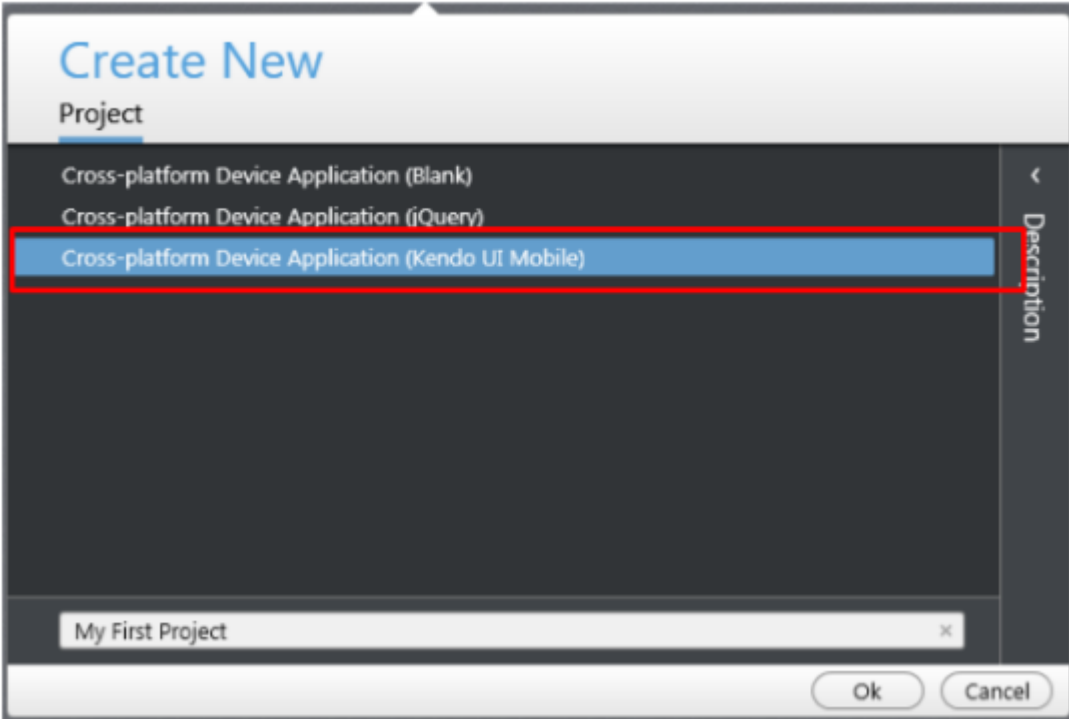
The image below shows a list of existing projects when you select My Projects in the dashboard.

For Limited Circulation Only



To create new project click on **New**. Projects in Icenium can be based on three templates. They are:

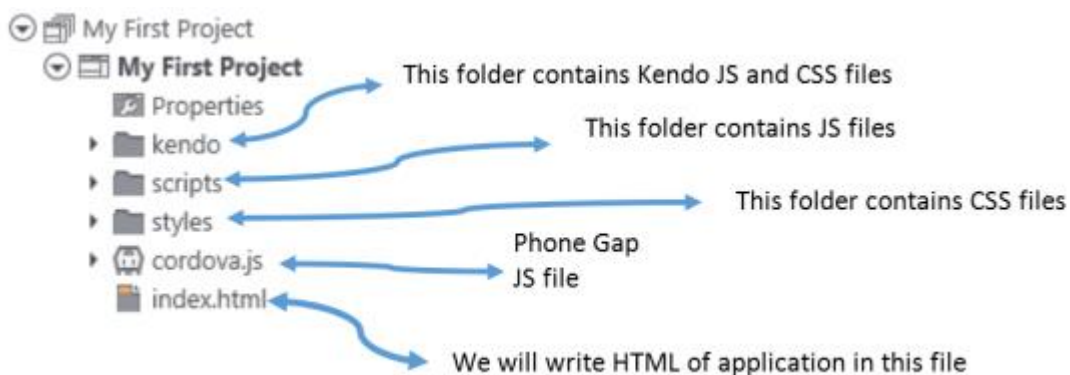
- Blank
- Using jQuery
- Using Kendo UI Mobile



Let us go ahead and create a cross-platform device application using Kendo UI Mobile.

Give a name to the project. You will notice that you will not be asked to select a location for saving the project. That is because, Icenium is a cloud-based IDE and it saves the project in Telerik cloud. Click Ok to create the application.

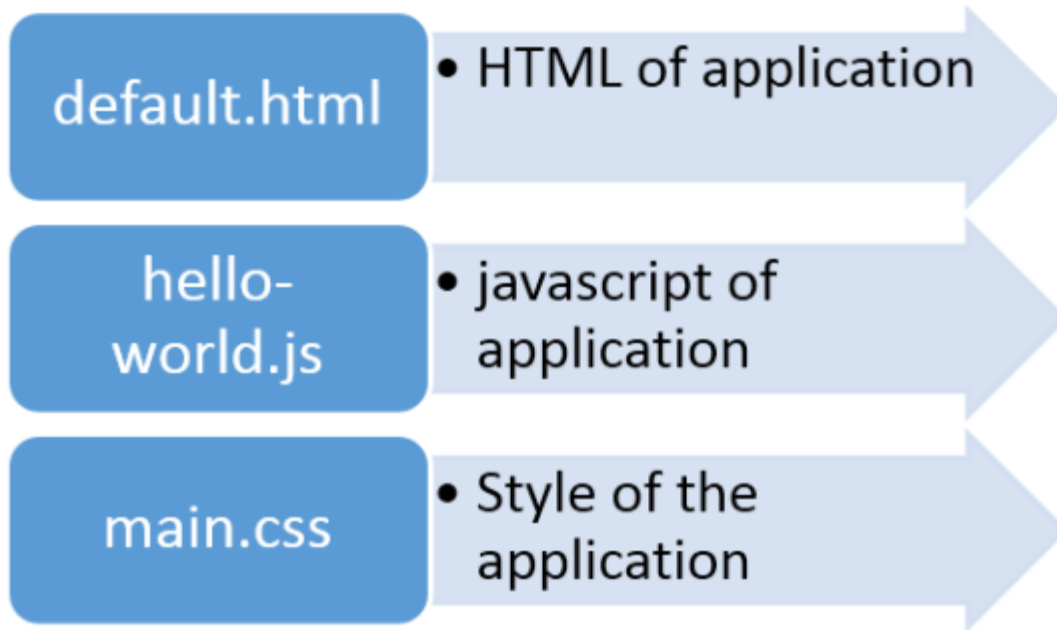
Now that the project is created, let us explore the project structure. The structure of the project is depicted as a tree in the below image:



If you look inside the file index.html, you will find code present already which serves as a reference code. As you can see in the below image, to create Hybrid application using Kendo UI Mobile, you need the following minimum references.

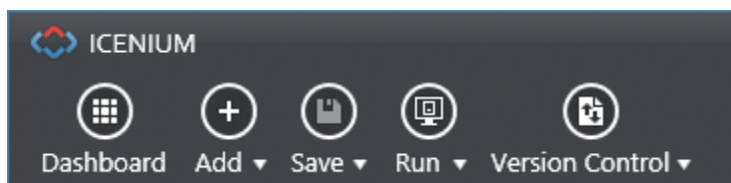

```
<meta charset="utf-8" />
<script src="cordova.js"></script>
<script src="kendo/js/jquery.min.js"></script>
<script src="kendo/js/kendo.mobile.min.js"></script>
<script src="scripts/hello-world.js"></script>
```

There are three default files you will be working on in the project.



However you are free to give any name to the above files.

Let us go ahead and run the default application we just created.

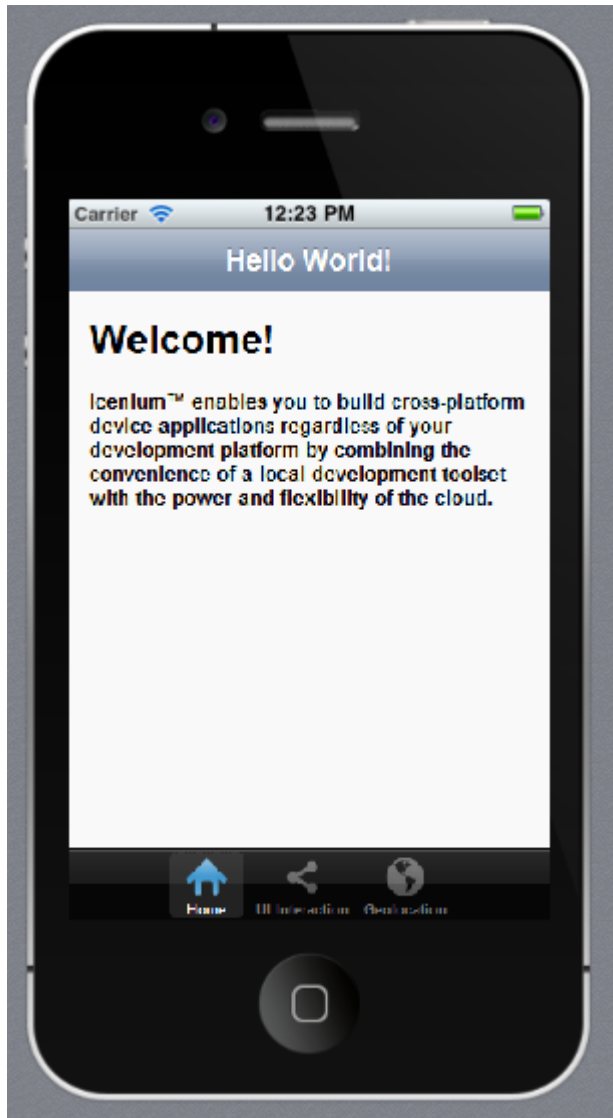


There are two ways you can run the application. You can run application either

- On Device
- In Simulator

On running the app in simulator, you will get an application running in iPhone simulator.

For Limited Circulation Only

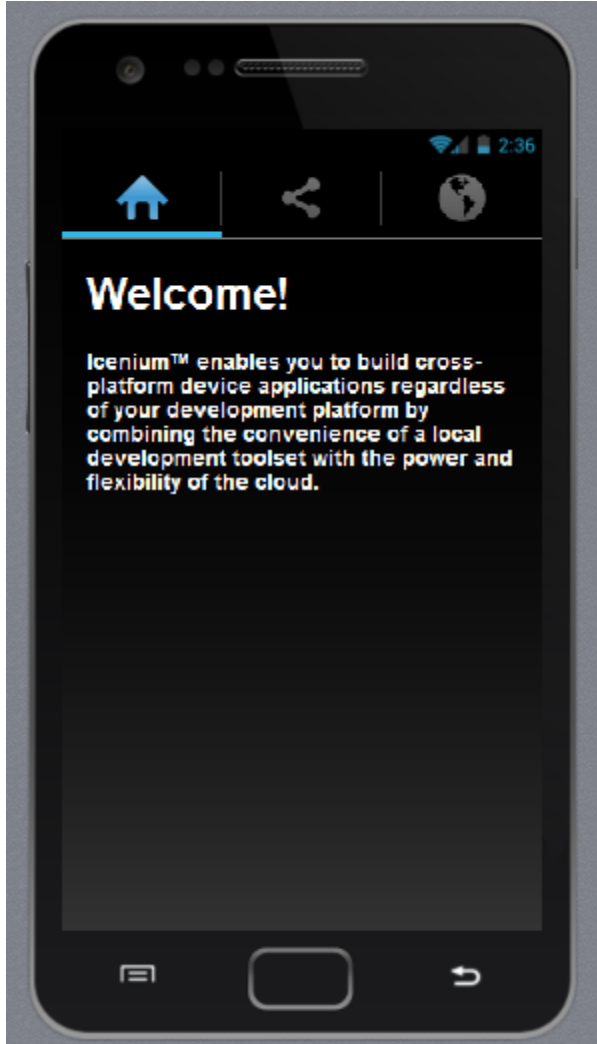


There are four simulator supported in Icenium. The simulators supported are shown in the below image:



For Limited Circulation Only

If you want to run the application in Android Simulator change the Device simulator and you are done



Conclusion

In this lab we created our first cross platform application using Icenium and Kendo UI Mobile.

Lab 2: Exploring Icenium features for Hybrid Application Development

Objective

In this lab we will explore following features of Icenium

Code Editor

Certificate
creation

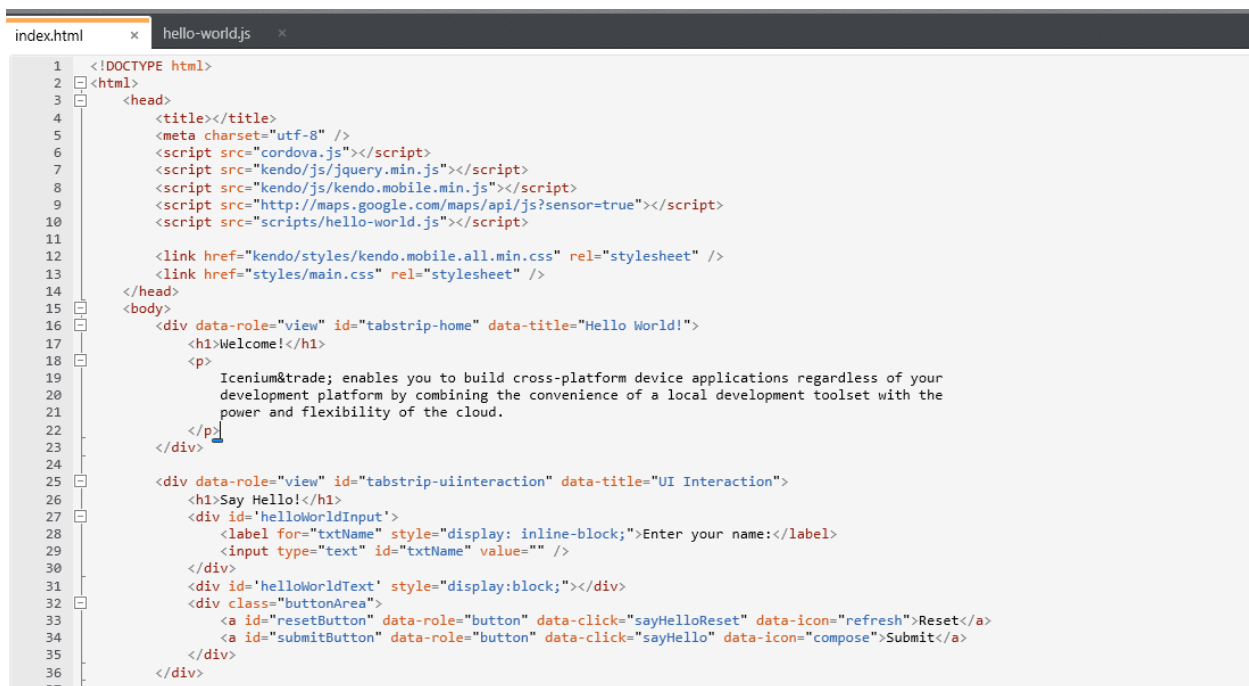
Code Sign and
Publishing

Version
Control

Github
integration

Code Editor

Since we are building a hybrid mobile application, we will be writing a lot of HTML, JavaScript and CSS code. The code editor in Icenium is clean and an elegant editor. It allows you to write code without much distraction.



```
index.html x hello-world.js x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title></title>
5     <meta charset="utf-8" />
6     <script src="cordova.js"></script>
7     <script src="kendo/js/jquery.min.js"></script>
8     <script src="kendo/js/kendo.mobile.min.js"></script>
9     <script src="http://maps.google.com/maps/api/js?sensor=true"></script>
10    <script src="scripts/hello-world.js"></script>
11
12    <link href="kendo/styles/kendo.mobile.all.min.css" rel="stylesheet" />
13    <link href="styles/main.css" rel="stylesheet" />
14  </head>
15  <body>
16    <div data-role="view" id="tabstrip-home" data-title="Hello World!">
17      <h1>Welcome!</h1>
18      <p>
19        Icenium™ enables you to build cross-platform device applications regardless of your
20        development platform by combining the convenience of a local development toolset with the
21        power and flexibility of the cloud.
22      </p>
23    </div>
24
25    <div data-role="view" id="tabstrip-uiinteraction" data-title="UI Interaction">
26      <h1>Say Hello!</h1>
27      <div id="helloWorldInput">
28        <label for="txtName" style="display: inline-block;">Enter your name:</label>
29        <input type="text" id="txtName" value="" />
30      </div>
31      <div id="helloWorldText" style="display:block;"></div>
32      <div class="buttonArea">
33        <a id="resetButton" data-role="button" data-click="sayHelloReset" data-icon="refresh">Reset</a>
34        <a id="submitButton" data-role="button" data-click="sayHello" data-icon="compose">Submit</a>
35      </div>
36    </div>
37  </body>
```

“Just Code” – a code refactoring tool from Telerik, is integrated within the Code editor. Within the Code Editor, you can perform tasks like:

1. Refactoring
2. Navigation
3. Code Commands

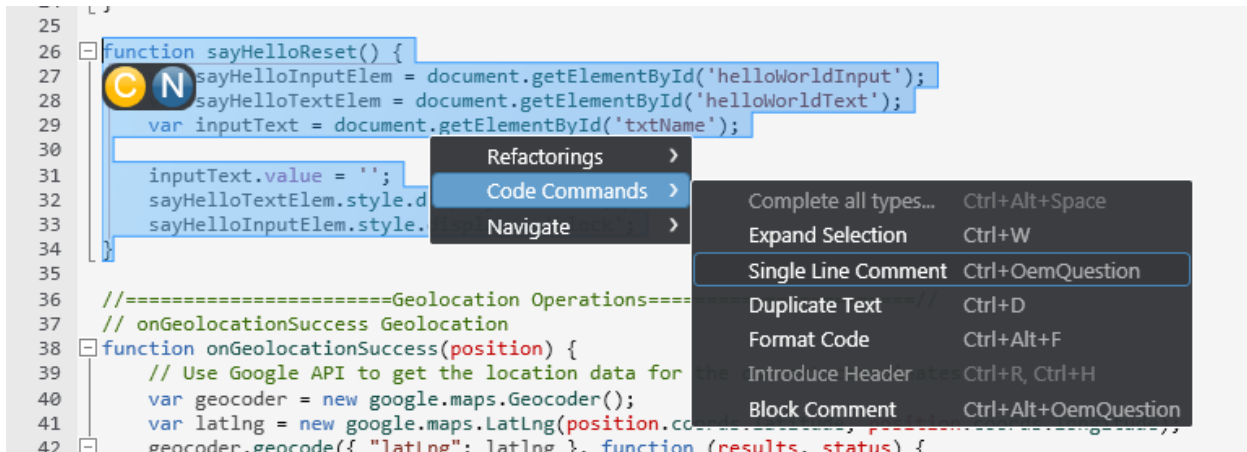
```
//=====Say Hello (Page 1) Operations=====//
function sayHello() {
  var sayHelloInputElem = document.getElementById('helloWorldInput');
  var sayHelloTextElem = document.getElementById('helloWorldText');
  var inputText = document.getElementById('txtName');
  sayHelloTextElem.innerHTML = sayHelloTextElem.innerHTML + inputText.value + '!';
  sayHelloTextElem.style.display = 'block';
  sayHelloInputElem.style.display = 'none';
}
```



If you select Code Commands, you can perform the following tasks:

1. Formatting of Code
2. Expand selection
3. Duplicate a text
4. Put a block comment

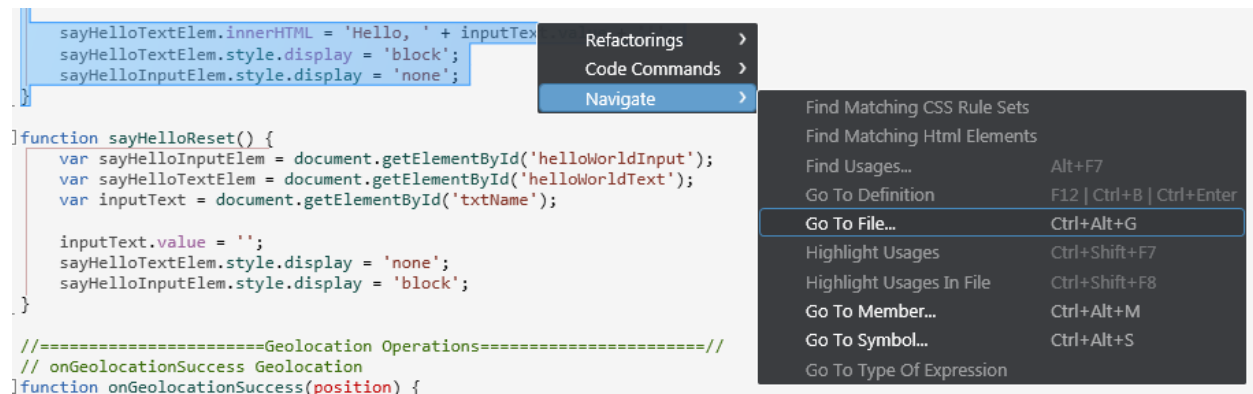
```
25
26 function sayHelloReset() {
27   sayHelloInputElem = document.getElementById('helloWorldInput');
28   sayHelloTextElem = document.getElementById('helloWorldText');
29   var inputText = document.getElementById('txtName');
30
31   inputText.value = '';
32   sayHelloTextElem.style.d
33   sayHelloInputElem.style.
34
35
36 //=====Geolocation Operations=====
37 // onGeolocationSuccess Geolocation
38 function onGeolocationSuccess(position) {
39   // Use Google API to get the location data for
40   var geocoder = new google.maps.Geocoder();
41   var latlng = new google.maps.LatLng(position.co
42   geocoder.geocode({ "latLng": latlng }, function (results, status) {
```



As you can see, the code editor provides short cuts for all the tasks. So if you are a heavy keyboard user you have short cuts to perform most of the tasks the editor supports.

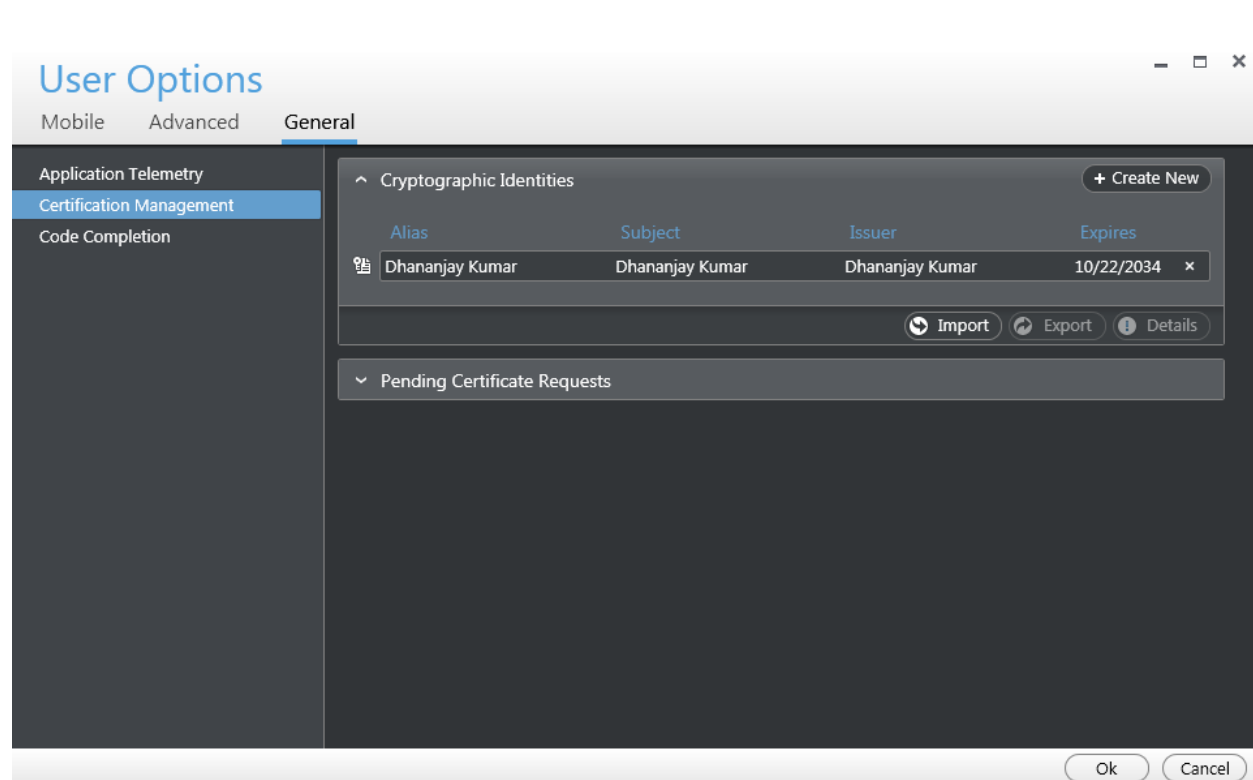
In you select Navigation, you can perform tasks like

1. Go To File
2. Go To Member
3. Go to Symbol



Certificate Management

One of the prime requirements for creating Android applications packages, is to have a certificate created. The package is usually a .apk file. Icenium allows you to create and manage certificate right from the IDE. Below image shows the certificate management feature within Icenium.



Icenium provides two options to create a certificate,

1. Request for a certificate
2. Create self-signed certificate

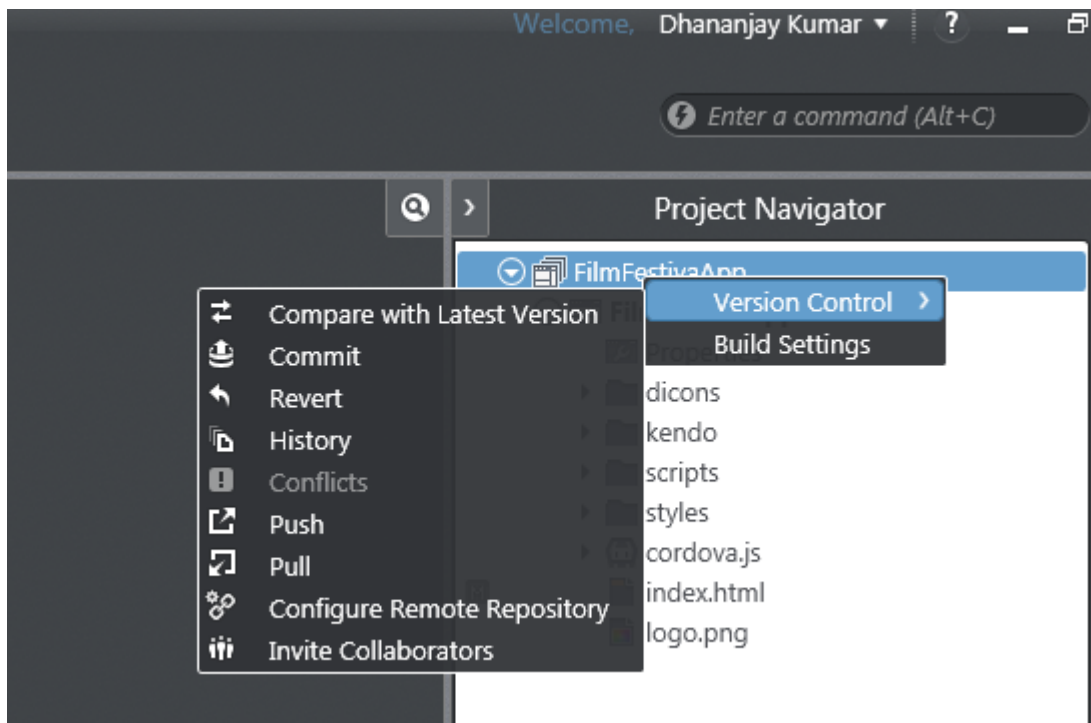


Click on Ok to create Signed Certificate.

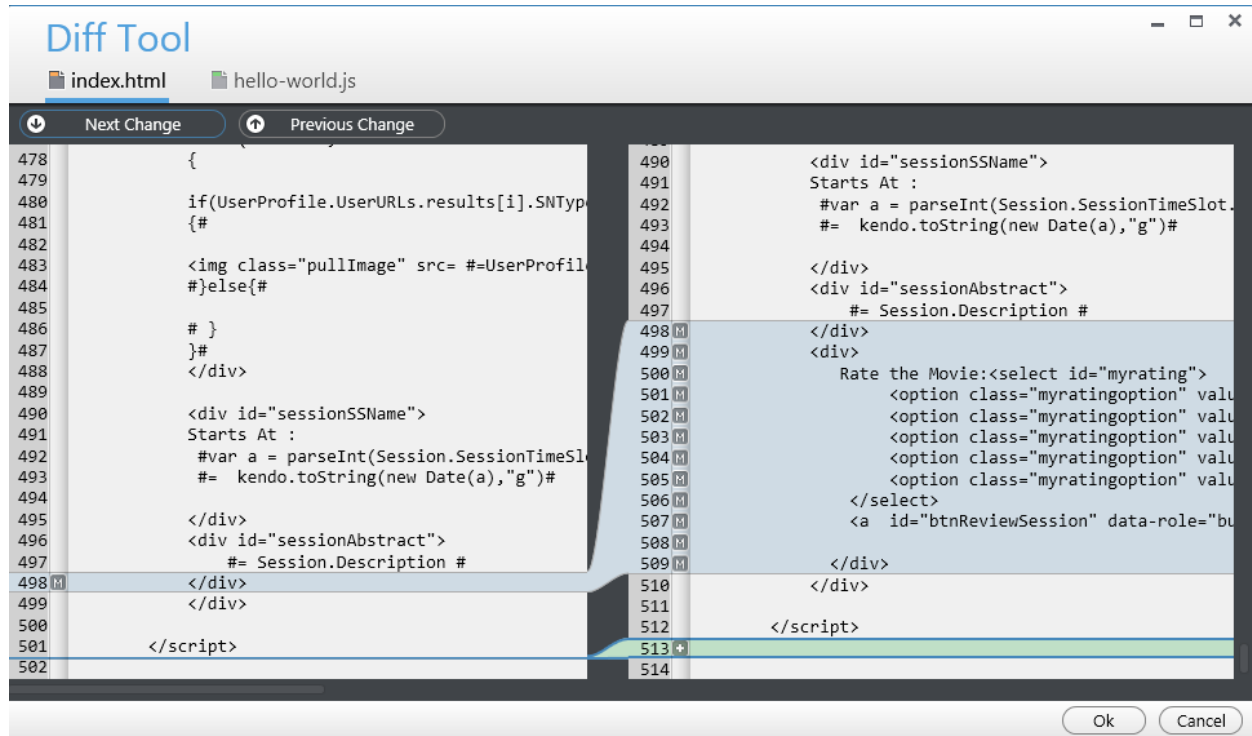
Version Control

Icenium provides complete version control solution for application development. You can work locally and perform the following tasks within Icenium

1. Compare the changes between local and latest versions on server
2. Commit changes
3. Revert changes
4. View history log in repository

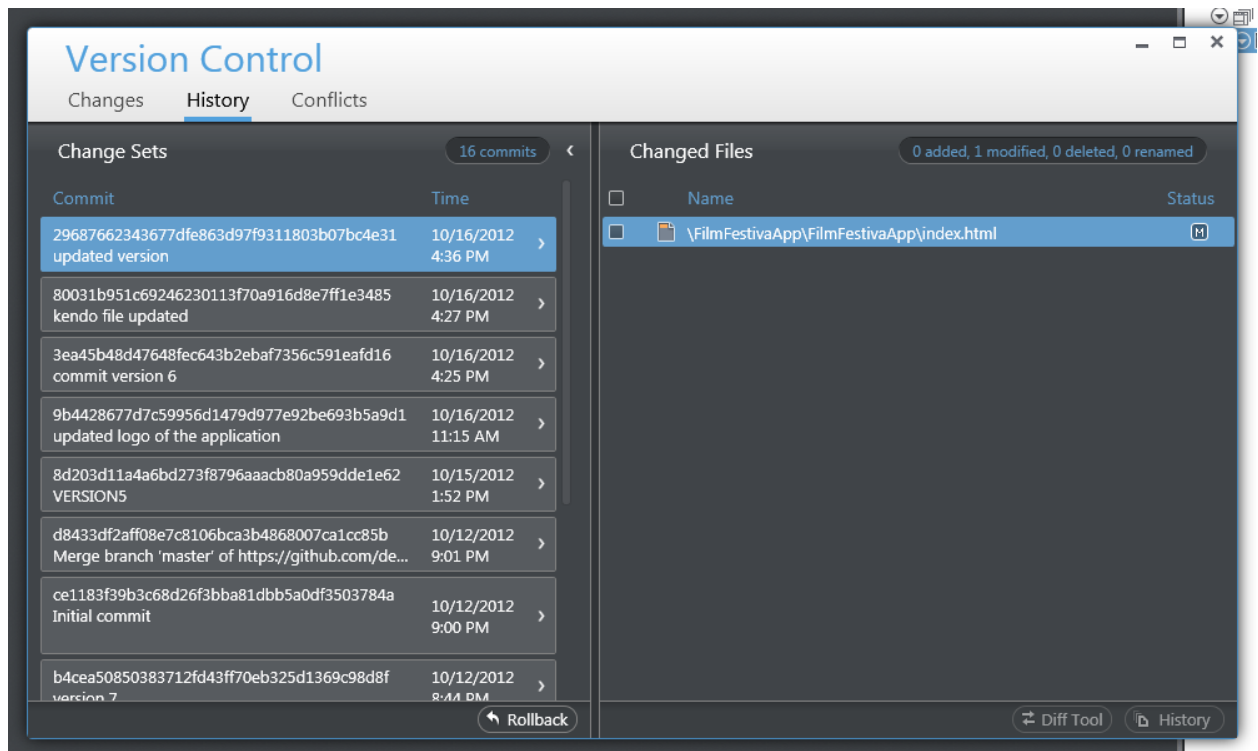


The Diff Tool allows you to compare changes, navigate from one change to another. The image below shows the Diff Tool in Icenium.

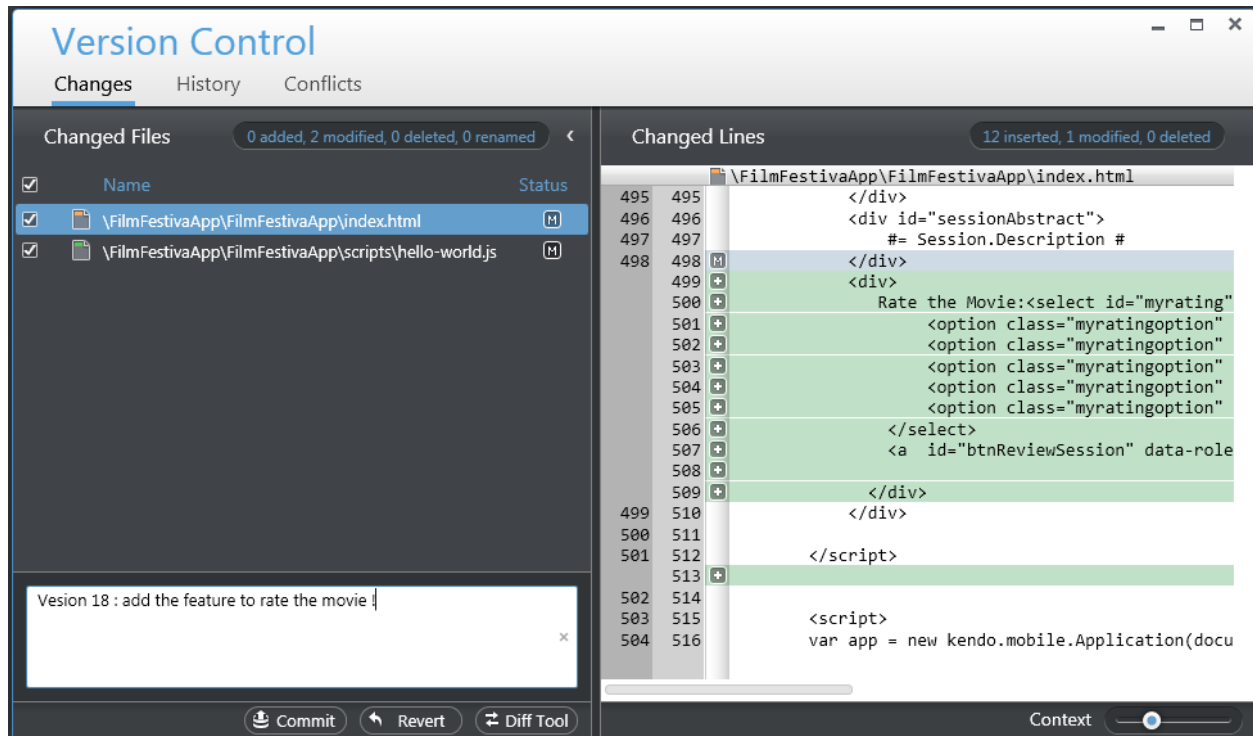


You can view the

1. History
2. Change Sets
3. Conflict etc. as shown in the above image,



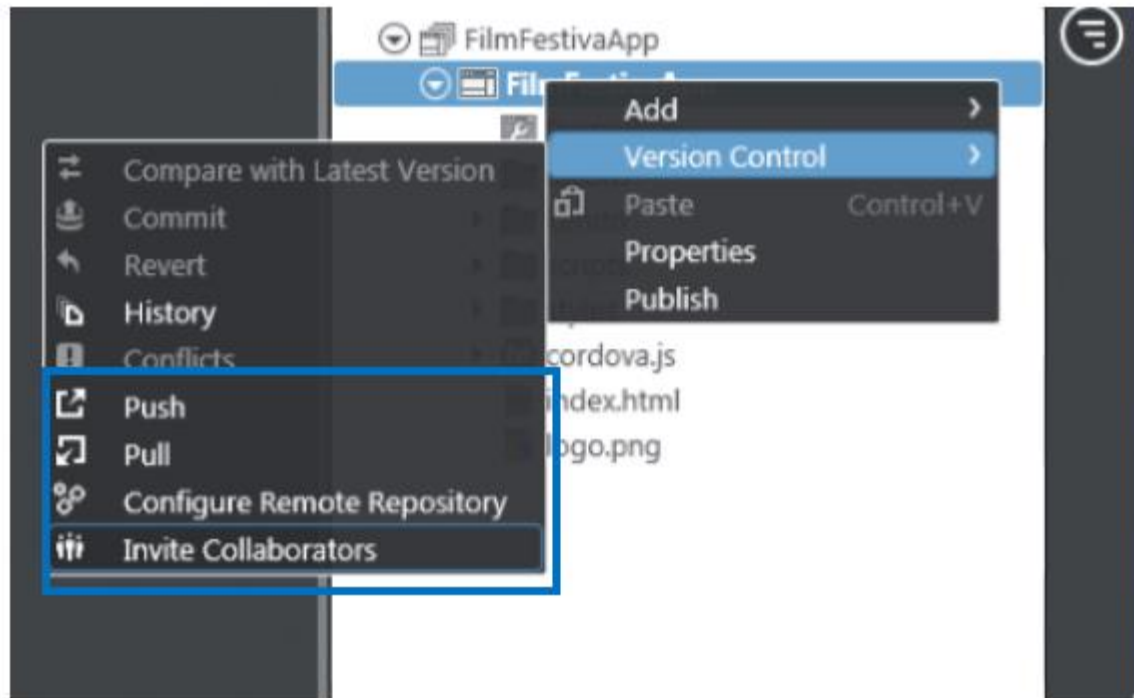
Icenium allows you to commit or revert changes. While committing you need to provide comment for that particular change. Committing code through Icenium is shown in the below image:



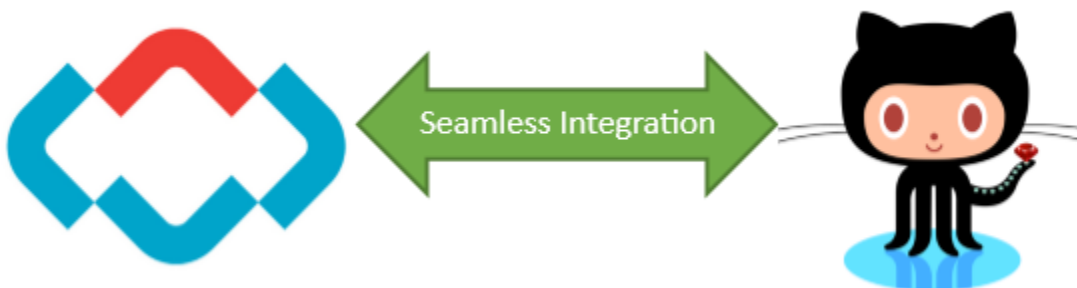
[Integration with github Repository](#)

Yes, you read it corectly. You can push and pull your project to and from your Github (or favourite Git repository) repository right from Icenium. Icenium allows you to:

1. Configure the repository
2. Pull from the repository
3. Push into the repository
4. Invite the Collaboratores



One of the best parts of Icenium is that you can right from the IDE we can Invite Collaborators . Before pushing to GitHub we need to commit the changes to the repository. Commit comments will be displayed next in the GitHub repository comment. When you create a project in Icenium, it saves that project in cloud and allows you to do version control. Apart from cloud integration, Icenium allows you to push and pull your project within a GitHub Repository as well.



For Limited Circulation Only

To integrate projects from Icenium to a GitHub repository, first create a repository in GitHub. You can create that on the [Github site](#) by clicking on **New Repository** link button



To create a new repository, you need to provide following information. I am symbolically providing information here to create a new repository.

For Limited Circulation Only


Owner debugmodedotnet


Repository name repoforcrossmobileapp ✓

Great repository names are short and memorable. Need inspiration? How about [north-american-lana](#).

Description (optional)

This is githib repo for my first cross mobile application

Public  Anyone can see this repository. You choose who can commit.

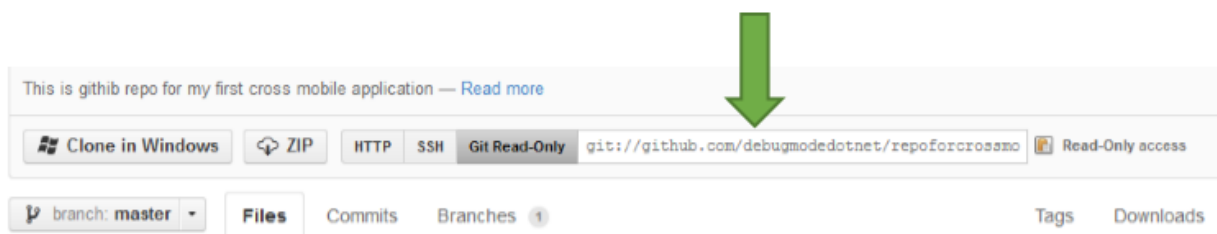
Private  You choose who can see and commit to this repository.

Initialize this repository with a README
This will allow you to `git clone` the repository immediately.

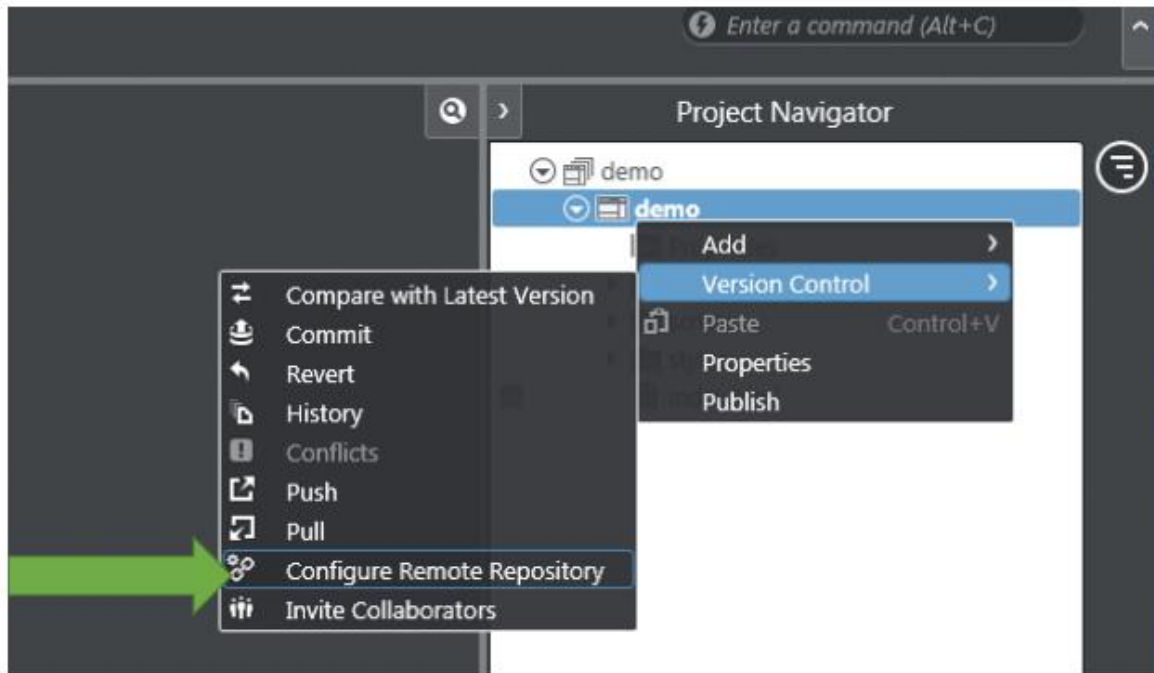
Add .gitignore: **None**

Create repository

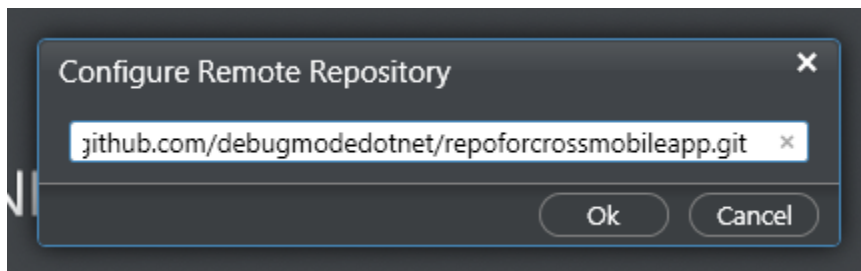
Once repository is created on GitHub, you will get a URL to work with that repository.



As of now, you have created a repository in GitHub. Now let us switch back to Icenium. Right-click on the project, then select Version Control. In Version Control, select the option of **Configure Remote Repository**.

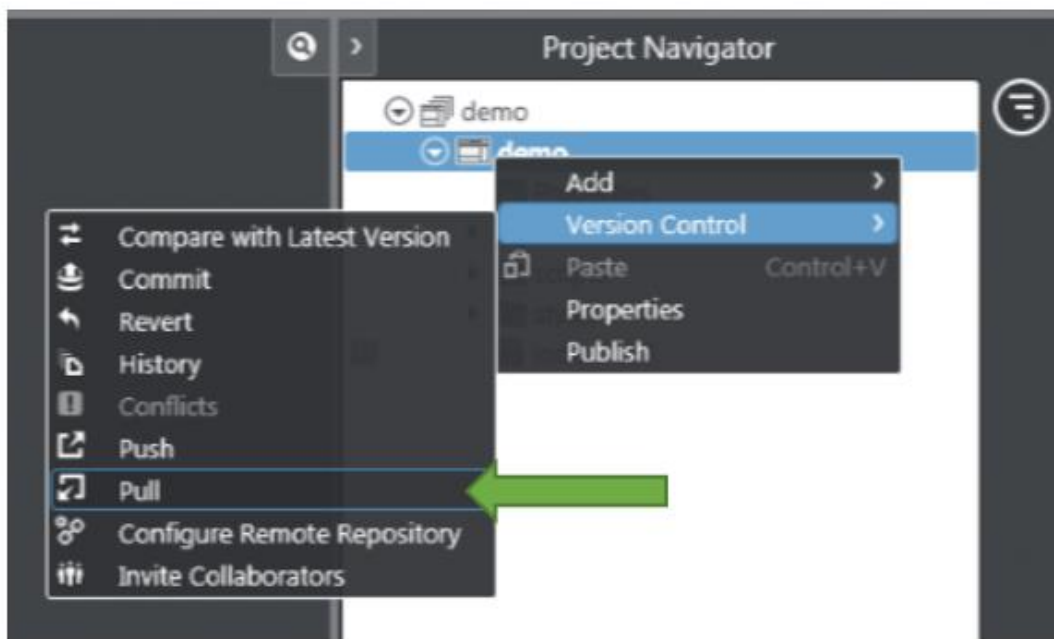


In the **Configure Remote Repository** dialog, provide the URL of Github repository.

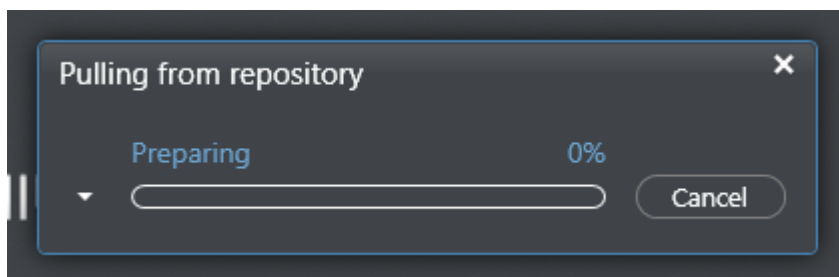


For Limited Circulation Only

Next, right-click on the project and select Version Control and select the option of **Pull**

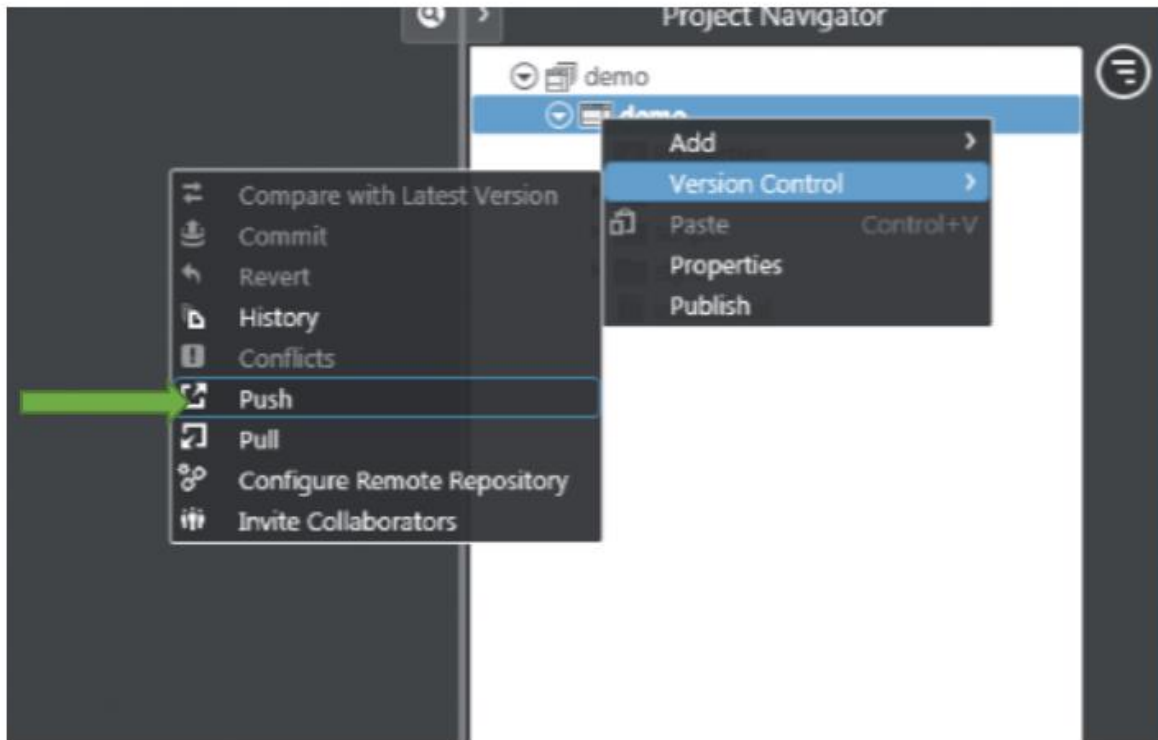


You will see that Icenium will start pulling the code from the remote repository.

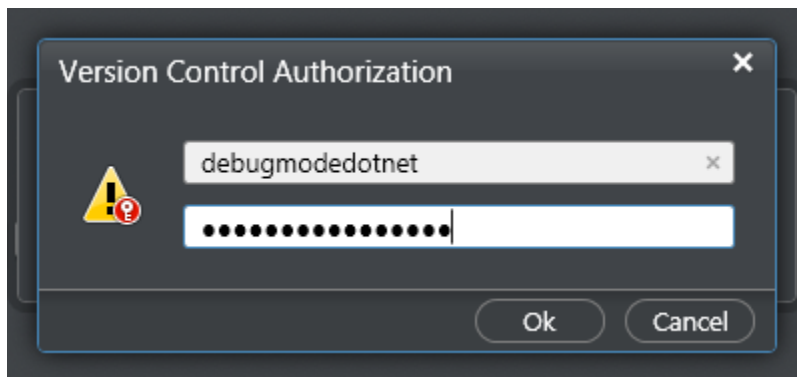


Now, to push the project to your GitHub repository, right click on the project and select **Version Control** then **Push** option.

For Limited Circulation Only

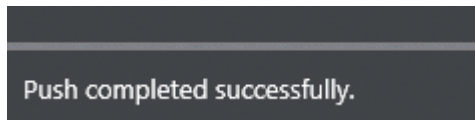


When Icenium starts to push your code to the remote repository, it will ask for version control authorization. Provide the appropriate credentials to proceed.



For Limited Circulation Only

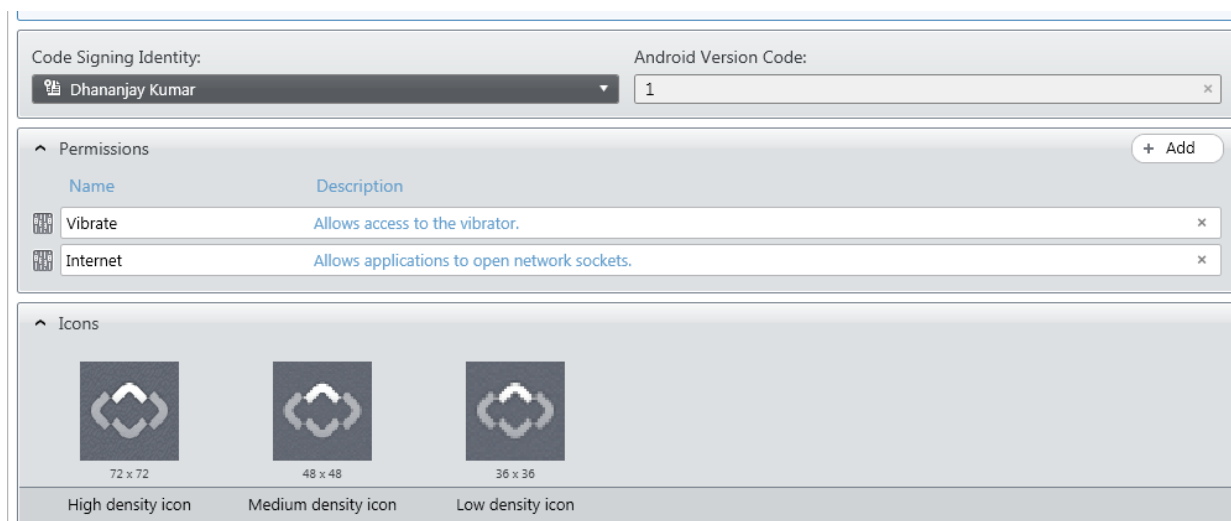
On successful completion of the push, you will get a message at bottom which says “Push completed successfully”.



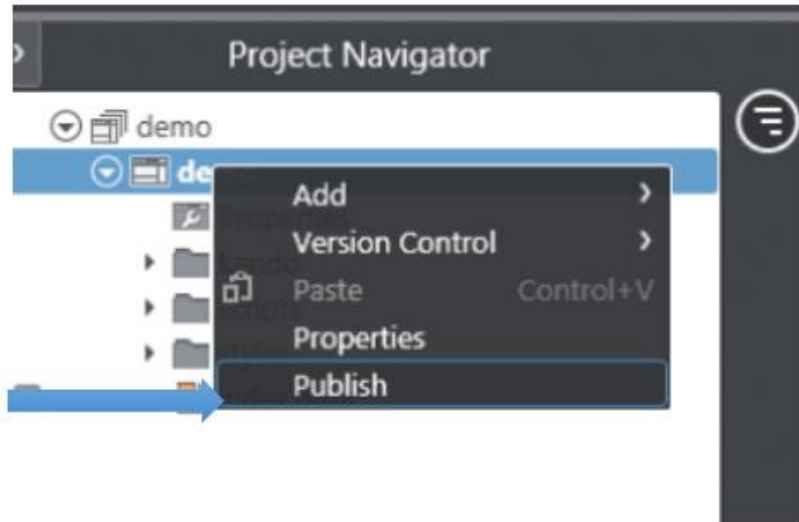
So in this way you can integrate project from Icenium to Github repository.

[Code signing ,Permissions and Publishing the application](#)

Icenium, also allows you to sign the code, configure the permissions and set the icons of the application all from within the IDE.



In Icenium with a simple right-click on the project , you can publish the application.



Conclusion

In this lab you learned about different features of Icenium.

For Limited Circulation Only

Lab 3: Creating a Cross-Platform “Twitter Search” Application using Kendo UI Mobile and Icenium

Objective

In this lab, we will create a Twitter Search Application using Kendo UI Mobile. While building this application, you will learn about the following widgets of Kendo UI Mobile:

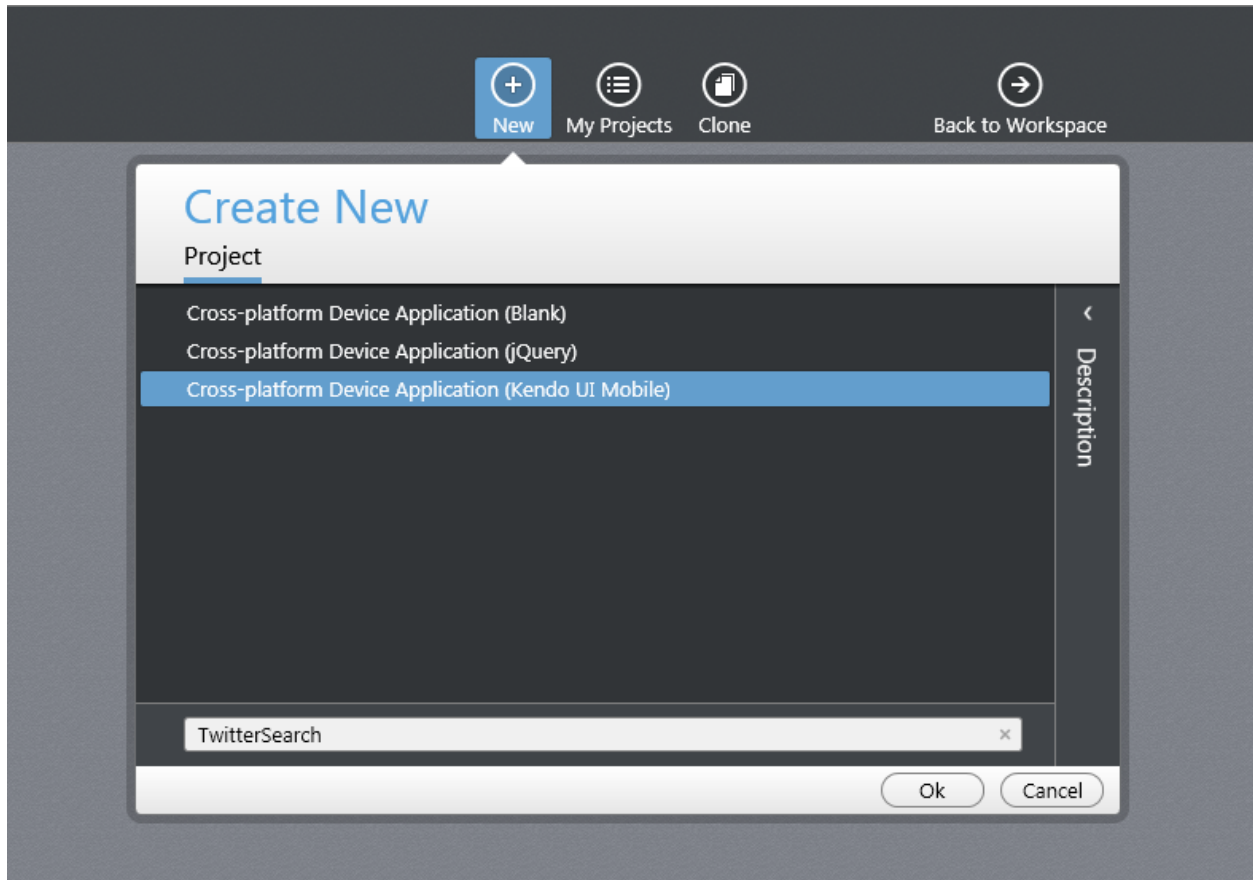
- Kendo UI Mobile View
- Kendo UI Mobile ListView
- Kendo UI DataSource
- Kendo UI Template
- Kendo UI ModalView
- Working with HTML5 localStorage

The Twitter Search Application which we will build in this lab will search tweets using Twitter API which returns results as JSON payload. The finished application will look like something like below image:



Step 1: Create Project

Launch Icenium Graphite and create new project. Select **Cross-platform Device Application** (Kendo UI Mobile) project template.



After creating project open **index.html** delete all the code from the body section except the script at the bottom of the page. After deleting the generated code, index.html should look like below,

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
  <script src="cordova.js"></script>
  <script src="kendo/js/jquery.min.js"></script>
  <script src="kendo/js/kendo.mobile.min.js"></script>
  <script src="http://maps.google.com/maps/api/js?sensor=true"></script>
  <script src="scripts/hello-world.js"></script>

  <link href="kendo/styles/kendo.mobile.all.min.css" rel="stylesheet" />
  <link href="styles/main.css" rel="stylesheet" />
</head>
<body>

  <script>
    var app = new kendo.mobile.Application(document.body, { transition: "slide",
layout: "mobile-tabstrip" });
  </script>
</body>
</html>
```

Next open **hello-world.js** JavaScript file and delete all the generated codes except the following lines of code:

```
// JavaScript Document
// Wait for PhoneGap to load
document.addEventListener("deviceready", onDeviceReady, false);

// PhoneGap is ready
function onDeviceReady() {

}
```

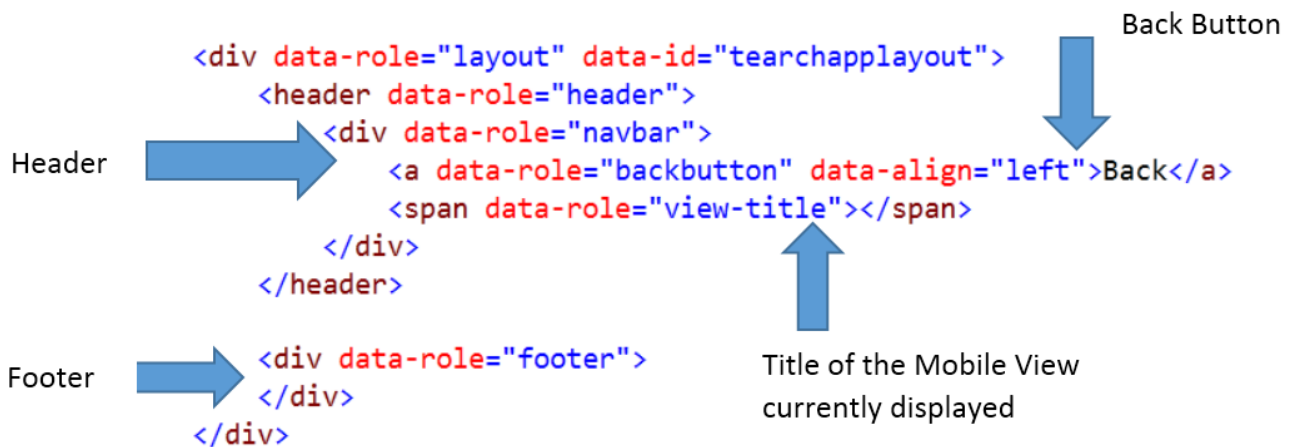
By now, you have a project where you can start writing the code to finish the Twitter Search application.

Step 2: Create Layout of the Application

Application layout is a very essential part of any application. Layout defines how header and footer of the application should look like. Layout should be adaptive in nature and render according to the platform. To create layout, you have to follow following steps:

1. Create a div
2. Set data-role attribute of div as **layout**
3. Set data-id of the div. This value will be used in mobile initialization.
4. Create header of application in layout div using `<header>` tag
5. Create footer of application in layout div. To create footer create a div inside layout div and set data-role attribute of div as footer

You can create application layout with header and footer as show below:



You can see that there are many important parts of layout. For example in header we create a navigation bar and within the navigation bar we have the back button and the title of the currently displayed view. Back button and view title are optional. It is purely on the requirement of the application that whether you want them as part of header or not. You can just have an image with logo of the application as layout header as well. In that case remove view title and back button and put an image as header of the application.

In Footer we will create a TabStrip.



In tabstrip there are two buttons. In next step you need to create two views to navigate to. At this point body of index.html should look like below:

```
<body>
  <div data-role="layout" data-id="tearchaplayout">
    <header data-role="header">
      <div data-role="navbar">
        <a data-role="backbutton" data-align="left">Back</a>
        <span data-role="view-title"></span>
      </div>
    </header>

    <div data-role="footer">
      <div data-role="tabstrip">
        <a href="#tweetsview" data-icon="home">Home</a>
        <a href="#settingview" data-icon="settings">Settings</a>
      </div>
    </div>

  </div>
  <script>
    var app = new kendo.mobile.Application(document.body, { transition: "slide",
      layout: "tearchaplayout " });
  </script>
</body>
```

Step 3: Create Settings View

You can create a view by following the steps below:

1. Create a div
2. Set **data-role** attribute of div as **view**
3. Set **data-show** attribute to a JavaScript function. This function will be called each time view is displayed in the application
4. Set **data-title** attribute to set the title of the view.

```
<div id="settingview" data-role="view" data-title="Settings" data-show="readsettings">  
  <input type="text" id="searchterm" class="searchterm" />  
  <a data-role="button" id="savebutton" data-click="savesettings" />  
</div>
```

On button click, text from the input box will be saved in local storage. Twitter search will be performed on the text saved in local storage. We are going to use HTML5 local storage API to save data locally. If you notice the data-click attribute of the button is set to javascript function **savesettings**. **savesettings** function is defined as following .

```
function savesettings() {  
  localStorage["searchterm"] = $('#searchterm').val();  
}
```

Write this function in hello-world.js file. When user navigates to Settings view, search term which is saved in the local storage should be displayed to the user. For that, we are setting data-show attribute of **settings** view to JavaScript function **readsetting**. Definition of the readsetting function is shown below.

```
var termtosearch="#kendoui" ;  
function readsettings() {  
  if (localStorage.searchterm) {  
    $('#searchterm').val(localStorage["searchterm"]);  
    termtosearch = localStorage["searchterm"];  
  }  
  else {  
    $('#searchterm').val("#kendoui");  
    termtosearch = "#kendoui";  
  }  
}
```

In this function, we are checking that if there is any setting saved. If not then we are displaying default text "#kendoui".

Step 4: Create Tweets View

Now you need to create Tweets view. Tweets will be fetched on basis of search term and displayed in this view. To display the tweets create a Kendo UI Mobile ListView inside Tweets view. ListView is a KendoUI widgets used to show repeated data.

We will create the Tweet view in the same way as we created the Settings view.

```
<div id=" tweetsview " data-role="view" data-title="Tweets" data-show="showtweets">
</div>
```

Inside Tweets view we need to create a ListView to display tweets. A ListView can be created by following the below steps:

1. By setting data-role attribute of as ListView.
2. For endless scroll set data-endlessScroll attribute to true
3. Set data-style attribute of the ListView.

```
<div id=" tweetsview " data-role="view" data-title="Tweets" data-show="showtweets">
  <ul data-role="listview"
    id="tweetlistview"
    data-endlessscroll="true"
    data-style="inset">
  </ul>
</div>
```

In the above code, we are performing the following operations:

1. Creating Kendo UI Mobile view by setting **data-role** property of HTML div element as **view**
2. Setting **data-show** attribute of view to JavaScript function **showtweets**
3. Creating ListView inside mobile view.
4. ListView is being created by setting **data-role** attribute of HTML element as **listview**

Step 5: Create Template to display tweets

Now you need to create a template. Template defines how data would be displayed in the ListView. To create Kendo Template, You need to create a script with type **text/x-kendo-template**. Values from data source will be rendered by putting property name as following

#= PROPERTYNAME #

Let us create Tweet Template as following.

```
<script id="tweetTemplate" type="text/x-kendo-template">
  <div class="tweets">
    <img class="pullImage" src=#= profile_image_url# alt="#= from_user #" />#= text #
    <div class="metadata">
      <a class="sublink" target="_blank" href="http://twitter.com/\\#!/#=
from_user #/status/#= id_str #" rel="nofollow">#= kendo.toString(new
Date(Date.parse(created_at)), "dd MMM HH:mm") #</a>
      <a class="sublink" href="http://twitter.com/#= from_user #"
rel="nofollow">#= from_user #</a>
    </div>
  </div>
</script>
```

To make data rendering more immersive you need to set the style of data in CSS. Notice that in the above template we are setting a style class name to class attribute of tag. These classes are defined in CSS. Open main.css and define following styles:

```
.tweets {
  padding: .5em .7em;
  font-size: .8em;
  line-height: 1.4em;
}
.pullImage {
  width: 64px;
  height: 64px;
  border-radius: 3px;
  float: left;
  margin-right: 10px;
}
.sublink {
  font-size: .9em;
  font-weight: normal;
  display: inline-block;
  padding: 3px 3px 0 0;
  text-decoration: none;
  opacity: .8;
}
```

Step 6: Create Data Source to Fetch Tweets

After creating data template, you need to fetch tweets from Twitter based on a search term and create data source. You need to create data source in **showtweets** JavaScript function.

```
function showTweets(e) {

  var lastID;
  var tweets = new kendo.data.DataSource({
```

For Limited Circulation Only

```
serverPaging: true,
transport: {
  read: {
    url: "http://search.twitter.com/search.json",
    dataType: "jsonp" // JSONP is required for cross-domain AJAX
  },
  parameterMap: function (options) {
    var parameters = {
      q: termtosearch,
      since_id: lastID //additional parameters sent to the remote service
    }

    return parameters;
  }
},
change: function () {
  var item = this.view()[0];

  if (item) {
    lastID = item.id_str;
  }
},
schema: { // describe the result format
  data: "results" // the data which the data source will be bound to is in the
"results" field
}
});

$("#tweetlistview").kendoMobileListView({
  dataSource: tweets,
  pullToRefresh: true,
  appendOnRefresh: true,
  template: $("#tweetTemplate").text()
});
}
```

In above function we are performing the following task:

1. Creating Kendo data source
2. Implementing pull to refresh
3. Binding data source to tweetlistview.

Conclusion

In this way you can create a Twitter Search Application. In this lab we learnt about

1. Kendo UI Mobile View
2. Kendo UI Mobile ListView

For Limited Circulation Only

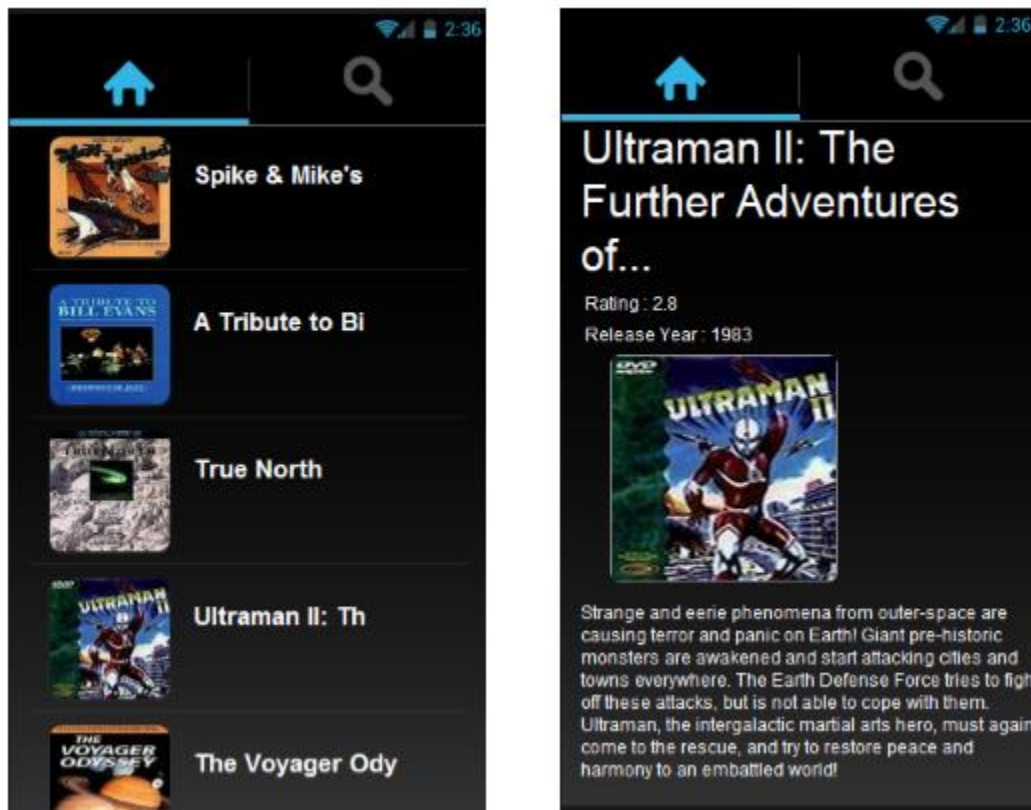
3. Template
4. Data Source
5. Consuming Service
6. Working with local storage

Lab 4: Creating Netflix Movie Explorer

Objective

In this lab you will create Netflix Movie Explorer Application. Idea is to fetch movies information from Netflix OData feed. Read more about Netflix OData API [here](#)

The finished application at the end of this lab is shown as below:



While creating this application, you will learn about

- KendoUI Mobile ListView
- KendoUI Mobile View
- Navigation between views
- Kendo DataSource and OData.
- Kendo UI Template

For Limited Circulation Only

Step 1: Create Project

Follow lab 3 for more details about creating project

Launch Icenium Graphite and create new project. Select **Cross-platform Device Application** (Kendo UI Mobile) project template to create project.

After creating project open **index.html** delete all the codes from the body section.

Next, open **hello-world.js** JavaScript file and delete all the generated code. After deleting code hello-world.js should look like below:

```
// JavaScript Document
// Wait for PhoneGap to load
document.addEventListener("deviceready", onDeviceReady, false);
// PhoneGap is ready
function onDeviceReady() {

}
```

By now you have a project which is ready for you to code your new application.

Step 2: Create Layout of the Application

Follow lab 3 for more details about Layout of the Application

In the layout, we have defined header and footer. Header contains the view title and shows the title of the current view. Layout also contains the footer. Inside footer we have two buttons which are part of the tabstrip widget.

```
<div data-role="layout" data-id="mobile-tabstrip">
  <header data-role="header">
    <div data-role="navbar">
      <span data-role="view-title"></span>
    </div>
  </header>

  <div data-role="footer">
    <div data-role="tabstrip">
      <a href="#moviesview" data-icon="home">Movies</a>
      <a href="#searchview" data-icon="search">Settings</a>
    </div>
  </div>
</div>
```

Step 3: Create Views

Next you need to create Mobile Views. We need to create two views - moviesview and searchview. In moviesview we will display movies details from Netflix and some search settings in searchview. So let us go ahead and create two views with simple label. Views can be created as following

```
<div data-role="view" id="moviesview" data-title="Movies">
    <h1>Hello Movies</h1>
</div>
<div data-role="view" id="searchview" data-title="Search">
    <h1>Hello Search</h1>
</div>
```

Setting layout of the application


You can set layout of the application at two levels:

1. At Application level
2. At View level.

There may be scenario where different view of your application requires different layout. Kendo UI mobile provides you option to set layout at the application level and also at the view level.

Application level layout can be set as following.

```
<script>
    var app = new kendo.mobile.Application(document.body, {
        transition: "slide",
        layout: "mobile-tabstrip",
        initial: "moviesview"
    });
</script>
```



Layout set as Application level

Now mobile-tabstrip is set as the layout of the application.

View level layout can be set as following.

```
<div data-role="view" id="moviesview"  
    data-title="Movies"  
    data-layout="viewlevellayout">  
    <h1>Hello Movies</h1>  
  
</div>
```

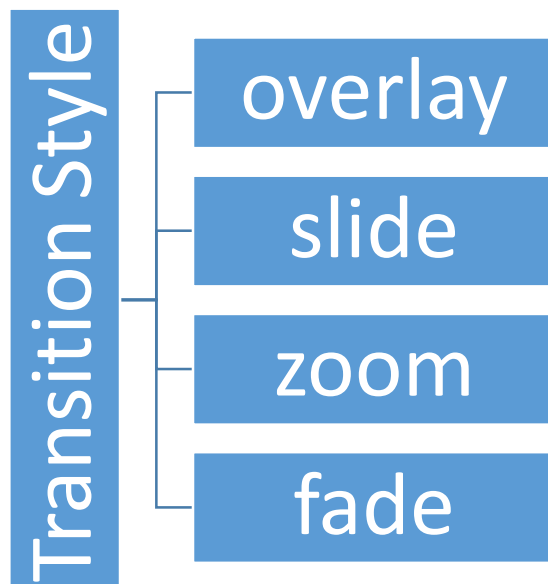


So a layout is applied at the view level by setting the data-layout attribute of the view itself.

If both application level and view level layout is set then always view level layout has more precedence than application level layout.

Setting transition style of the application

Kendo UI supports four transition styles, they are as follows.




Default transition style is "slide". You can apply transition style in three ways:

1. At the application level
2. At the view level
3. At the control level


When set at application level same transition style will be applied to entire application. All views of the application will adhere to the same transition style. At application level transition style can be set by providing value of transition property in Kendo mobile initialization.

```
<script>
  var app = new kendo.mobile.Application(document.body,
  {
    transition: "slide",
    initial: "mainview",
    layout: "applayout"
  });
</script>
```



Other option to set transition style is at view level. On navigating to this view user will experience transition style set at the view.

```
<div data-role="view"
  id="otherview"
  data-title="Other View"
  data-transition="fade">
</div>
```



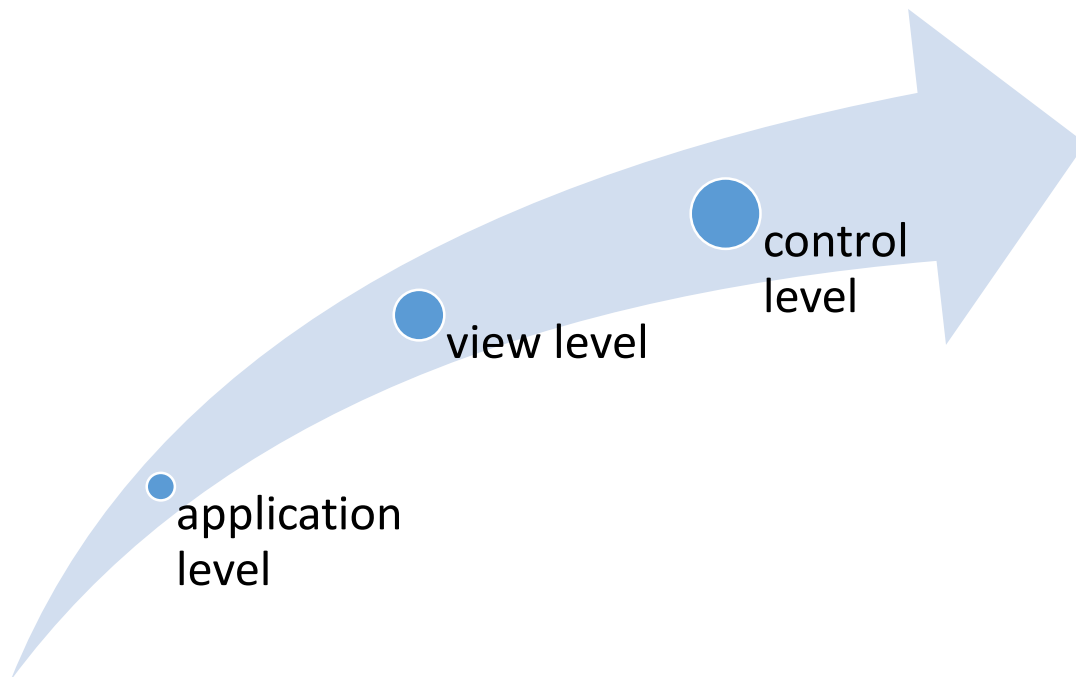
We can set transition style at control level as well. In following case we are applying transition style to a kendo button. User will experience zoom transition behavior while navigating to view set in the href property of the kendo button.

```
<a id="navigate"  
  href="#otherview"  
  data-role="button"  
  data-transition="zoom">
```

Navigate to other view

```
</a>
```

If transition style is set at all the three levels then Control level has highest priority and application level has lowest priority.



In this way you can apply different transition style to the application

For Limited Circulation Only

Step 4: Create DataSource

We will create data source from OData feed of Netflix. We will fetch first 30 movies from Netflix and create data source. You can find Title of all the Movies from Netflix ODATA from below link.

<http://odata.netflix.com/Catalog/Titles>

We can create KendoUI data source as following,

```
var movieTitleData;
movieTitleData = new kendo.data.DataSource(
    {
        type: "odata",
        endlessScroll: true,
        batch: false,
        transport: {
            read: {
                url:
                "http://odata.netflix.com/Catalog/Titles?$select=Id,ShortName,BoxArt&$top=100",
                dataType: "jsonp",

                data: {
                    Accept: "application/json"
                }
            }
        }
    }
);
```

There are few points worth discussing about creating data source

1. Data is fetched from OData feed of Netflix
2. Since data source is reading OData feed hence **type** of data source is set as **odata**
3. In transport attribute url is set as url of OData feed
4. datatype attribute is set as jsonp to handle cross domain call.
5. data attribute is set to accept json format of data

Step 4: Create Template

Now you need to create a template. Template defines how data would be displayed in the ListView. To create Kendo Template, You need to create a script with type **text/x-kendo-template**. Values from data source will be rendered by putting property name as following

#= PROPERTYNAME #

Let us create Movies Template as following.

```
<script id="movieTemplate" type="text/x-kendo-template">
  <div>
    <a href="\#moviedetailview?Id=#:data.Id#"
      class="km-listview-link"
      data-role="listview-link">
      <h4>#=data.ShortName.substring(0,15)#</h4>
      <img src= "#=data.BoxArt.MediumUrl#" />
    </a>
  </div>
</script>
```

Some important point about movietemplate :

1. There is an anchor element with href set to **moviedetailview**. We need anchor element with href to navigate to other view (detail view) on selecting of a particular movie item.
2. While navigating id of the movie is being passed as query parameter. On basis of this movie id we will create data source for moviedetail view
3. Movie image and first fifteen character of movie short name will be rendered in the listview.

To make data rendering more immersive you need to set the style of data in CSS. You will notice that, in the above template we are setting style for different element of moviesview. These classes are defined in CSS. Open main.css and define following styles:

```
#moviesview
h4 {
  display: inline-block;
  font-size: 0.9em;
  margin: 1em 0 .5em 1em;
}

#moviesview
img {
  float: left;
  width: 80px;
  height: 80px;
  margin: 0;
  -webkit-box-shadow: 0 1px 3px #333;
  box-shadow: 0 1px 3px #333;
  -webkit-border-radius: 8px;
  border-radius: 8px;
}
```

Step 5: Create ListView

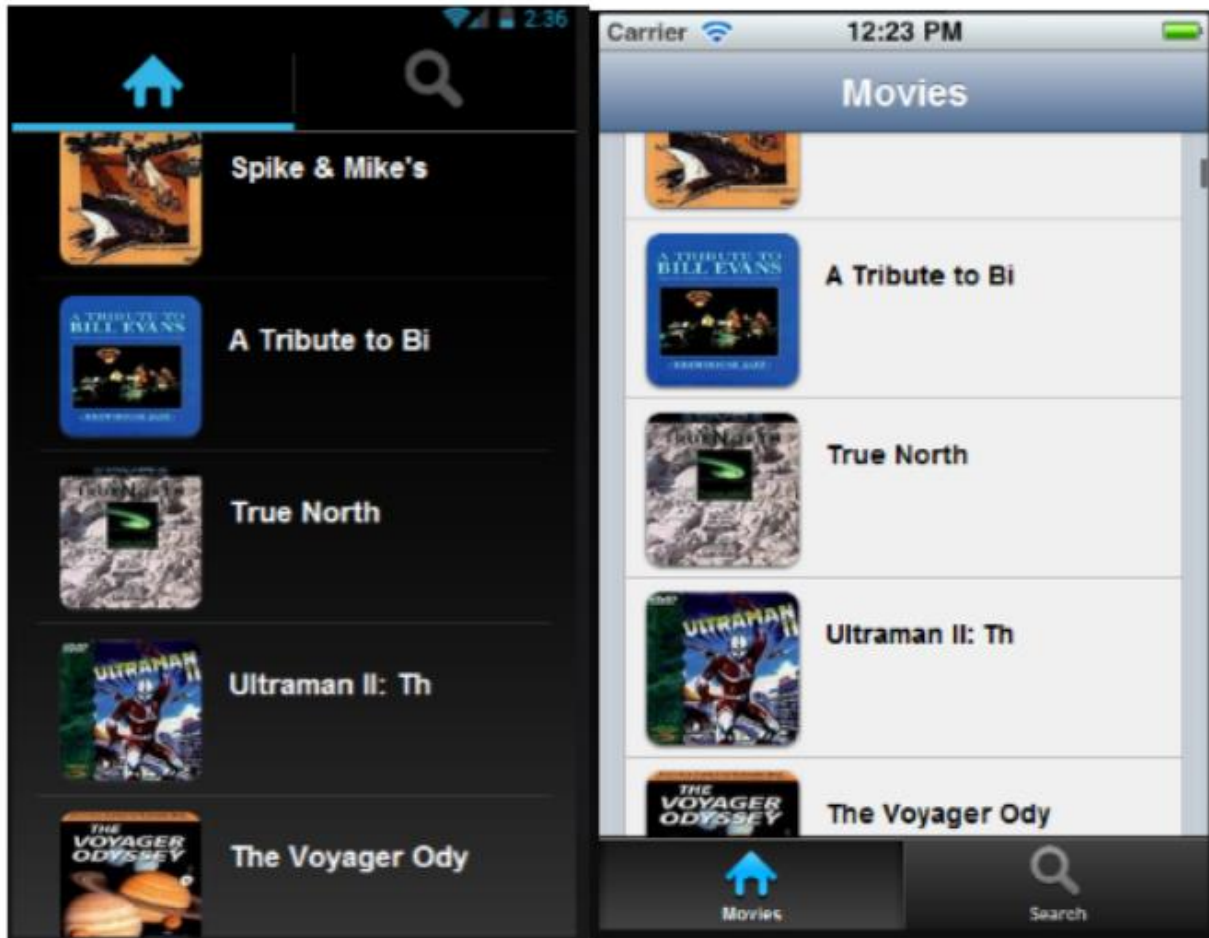
As of now we have created datasource and template. Now we need to create ListView in the moviesview. ListView can be created as below

```
<div data-role="view" id="moviesview"
    data-title="Movies">
  <ul id="movietitlelist"
    data-source="movieTitleData"
    data-endlessscroll="true"
    data-template="movieTemplate"
    data-role="listview"
    data-style="inset">
  </ul>
</div>
```

In ListView we are setting data-source attribute to movieTitleData. We created this data source in previous step. Next we are setting data-template to moviesTemplate created in previous step.

Run Application

Go ahead and run the application. You should be able to view 100 movies in ListView from Netflix OData feed.



[Create Movie Detail View with dynamic navigation](#)

On selecting an item user will be navigated to Movies Detail View. In this view details of selected movie will be displayed.

```
<div data-role="view"
      id="movieDetailView"
      data-title="Movie Details"
      data-show="showMovieDetails">

</div>
```

For Limited Circulation Only

Now we need to create template to display data. In Template we will display name, average rating, release year, movie image and synopsis.

```
<script id="movieDetailTemplate" type="text/x-kendo-template">
  <div>

    <div class="sname">#=data.ShortName#</div>
    <div class="stitle">Rating : #=data.AverageRating#</div>
    <div class="stitle">Release Year : #=data.ReleaseYear#</div>
    <img class="sponsorthumbnail" src=#=data.BoxArt.MediumUrl# /> </br>
    <div class="sdetail">
      #=data.Synopsis#
    </div>
  </div>

</script>
```

To make data rendering more immersive you need to set some styles on the data in CSS. You will notice that, in the above template we are setting style of different class for different element. These classes are defined in CSS stylesheet. Open main.css and define the following styles:

```
.sname
{
    font-size: 30px;
    font-weight: 100;
    margin-left: 10pt;
}
.stitle
{
    font-size: 12px;
    font-weight: 100;
    margin-left: 12pt;
    margin-top:5pt;
}
.sponsorthumbnail
{
    vertical-align: middle;
    display: inline-block;
    margin-top:5pt;
    margin-left: 25pt;
    margin-right: 6pt;
    margin-bottom:10pt;
    height: 150px;
    width: 150px;
    border-radius: 6px;
    border-style: solid;
    border-color: #999;
    border-width: 1px;
    background-size: 100% auto;
    background-repeat: no-repeat;
    background-position: center center;
```

For Limited Circulation Only

```
        background-color: white;
    }
    .sdetail
    {
        font-size: 12px;
        font-weight: 100;
        margin-left: 10pt;
    }
}
```

As of now we have created view and template to show movie detail. Next we need to create data source and bind data to template in **showMoviesDetails** function.

We can read the parameter passed to the view as shown below:

```
function showMovieDetails(e)
{
    var query = e.view.params.Id.toString();
    console.log(query);
}
```

We are converting the query parameter which was passed as integer to string format. Next you need to create a data source which will fetch this particular movie from Netflix OData feed.

```
var moviedetailData = new kendo.data.DataSource(
    {
        type: "odata",
        endlessScroll: true,
        batch: false,
        transport: {
            read: {
                url:
"http://odata.netflix.com/Catalog/Titles?$select=Id,ShortName,BoxArt,Synopsis&$top=100",
                dataType: "jsonp",

                data: {
                    Accept: "application/json"
                }
            }
        },
        filter: { filters: [{ field: "Id", operator: "eq", value: query } ] }
    });
```


For Limited Circulation Only

In data source you need to apply a filter to fetch detail of selected movie. Filter can be applied while creating Kendo data source. In above scenario filter is applied on Id field with eq i.e. equality operator.

```
filter: { filters: [{ field: "Id", operator: "eq", value: query }] }
```

After creating data source you need to dynamically set data to the template. That can be done as following

```
var movieDetailTemplate = kendo.template($("#movieDetailTemplate").text());

moviedetailData.fetch(function () {

    var item = moviedetailData.at(0);
    view.scrollerContent.html(movieDetailTemplate(item));
    kendo.mobile.init(view.content);
});
```

Eventually **showMoviesDetails** function will be as follows. This function will be called when user will select a movie to view the movie details.

```
function showMovieDetails(e) {
    var query = e.view.params.Id.toString();
    var view = e.view;

    var moviedetailData = new kendo.data.DataSource(
        {
            type: "odata",
            endlessScroll: true,
            serverFiltering: true,
            transport: {
                read: {
                    url:
"http://odata.netflix.com/Catalog/Titles?$select=Id,ShortName,BoxArt,AverageRating,ReleaseYear,Synopsis&$top=100",
                    dataType: "jsonp",

                    data: {
                        Accept: "application/json"
                    }
                }
            },
            filter: { filters: [{ field: "Id", operator: "eq", value: query }] }
        });

    var movieDetailTemplate = kendo.template($("#movieDetailTemplate").text());

    moviedetailData.fetch(function () {

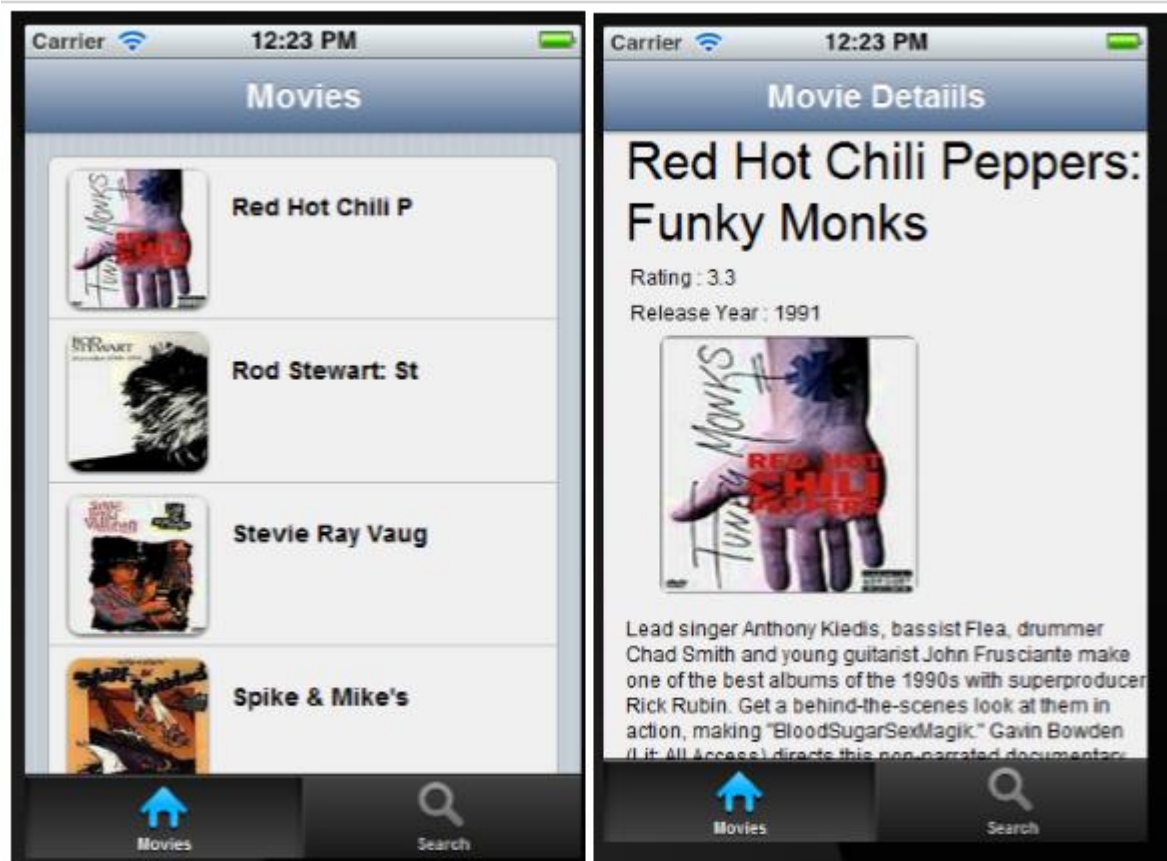
        var item = moviedetailData.at(0);
```

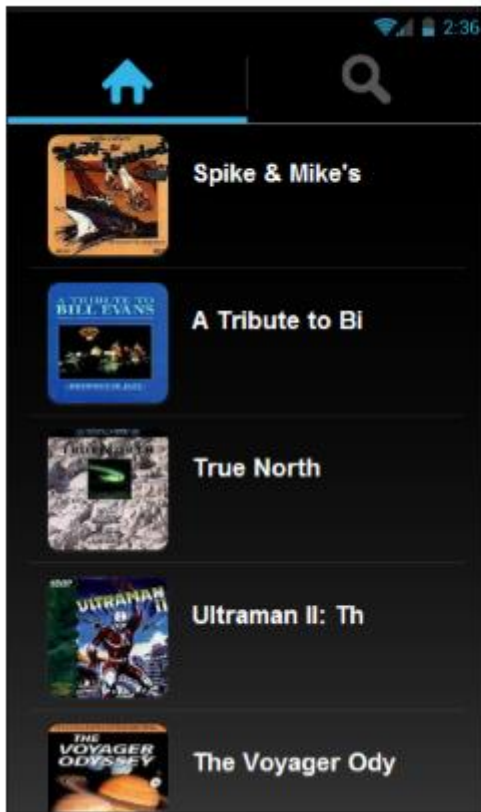
For Limited Circulation Only

```
        view.scrollerContent.html(movieDetailTemplate(item));
        kendo.mobile.init(view.content);
    });
}
```

Running the Application

Go ahead and run the application. You should be able to view the movies and navigate to details of a selected movie.





Conclusion

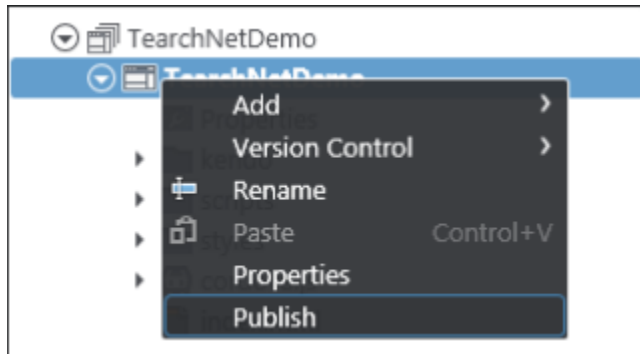
In this lab we learnt about

1. Working with Odata
2. Dynamic navigation between views
3. Working with template and ListView

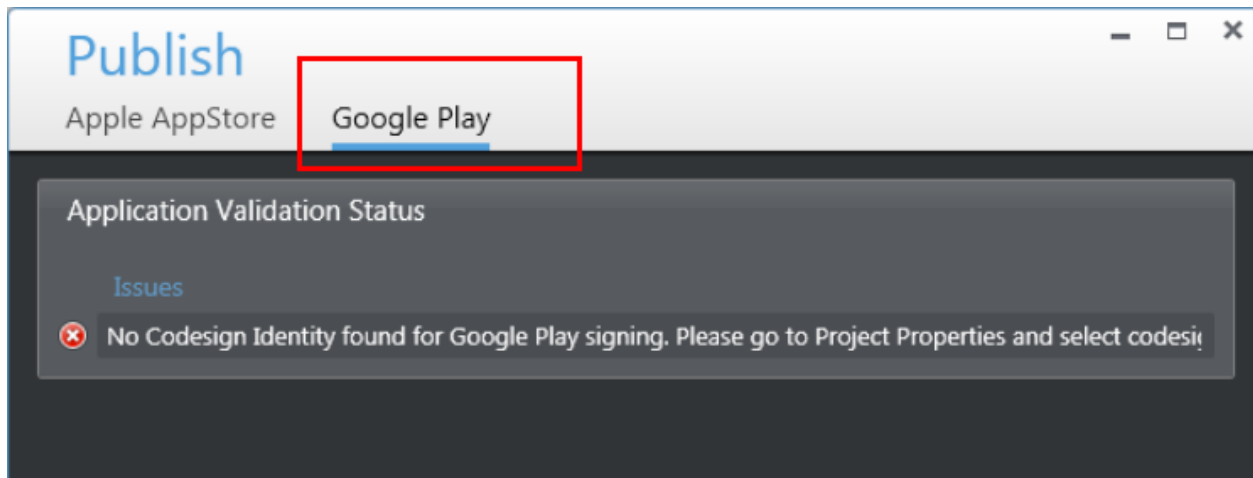
Lab 5: Create APK package for Google Play using Icenium

In this lab we will take a look at creating APK package for submission to Google Play using Icenium. Follow the below defined walkthrough to create APK package:

To create package, right click on the project and select Publish.

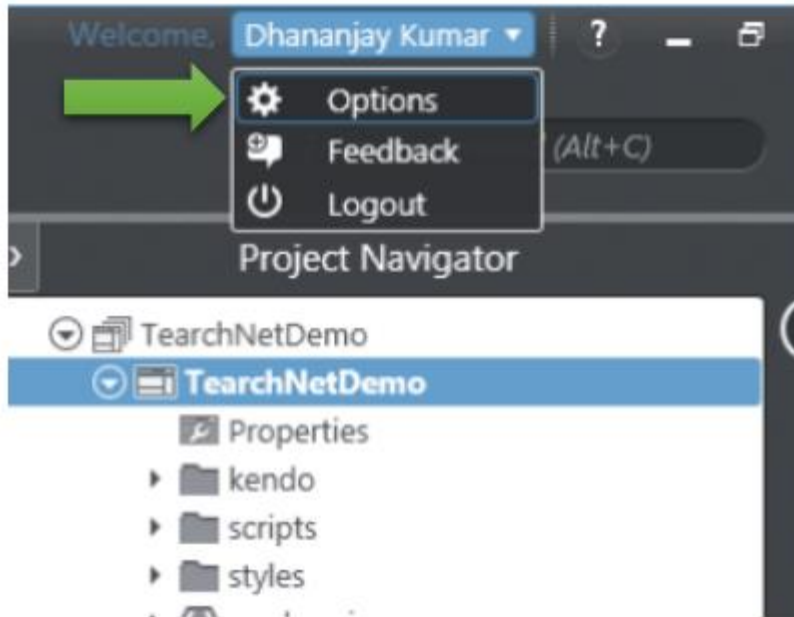


Next, in the publish dialog click on Google Play. You will get an error message that there is no certificate or Code Sign Identity found for Google Play Signing.

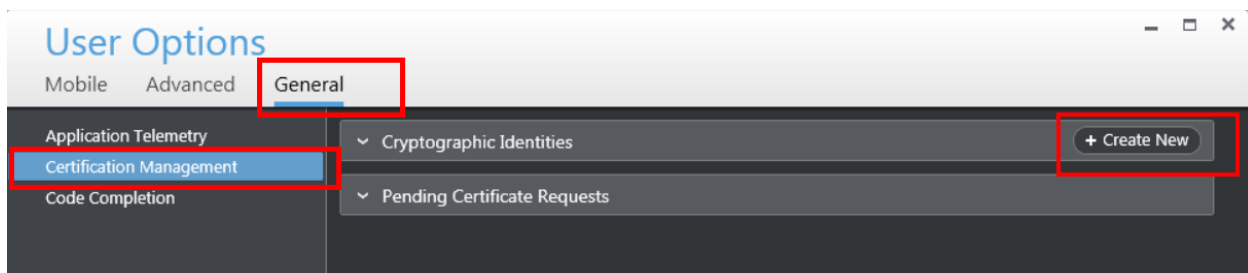


To solve this issue click on the option in Icenium Graphite IDE.

For Limited Circulation Only



In Users Option window select **General** tab and then Certificate Management option.



In Certificate Management option you will get an option to create New Certificate. Click on Create New to create new certificate. You can either use

1. Self-signed identity
2. Request for the Certificate

Let us go ahead and request for the Self-signed identity. In Self-signed identity window you need to provide following vital information,

- Country
- Type of the self-signed identity. In this choose Google Play as option. Other option available is Generic.
- Configure Start Date and End Date

For Limited Circulation Only



Create a Self-Signed Identity

Name: Dhananjay Kumar

Email: dhananjay.kumar@live.com

Country: India

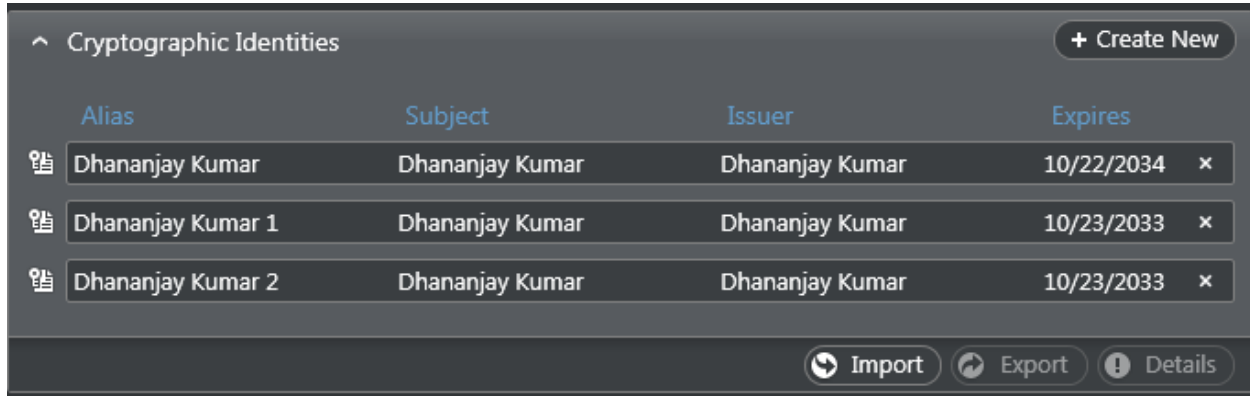
Type: Google Play

Start Date: 12/24/2012

End Date: 10/23/2033

Ok Cancel

After creating Self-signed identity you can find them in Cryptographic Identities section. Below you can see a list of three self-signed identity.



Cryptographic Identities

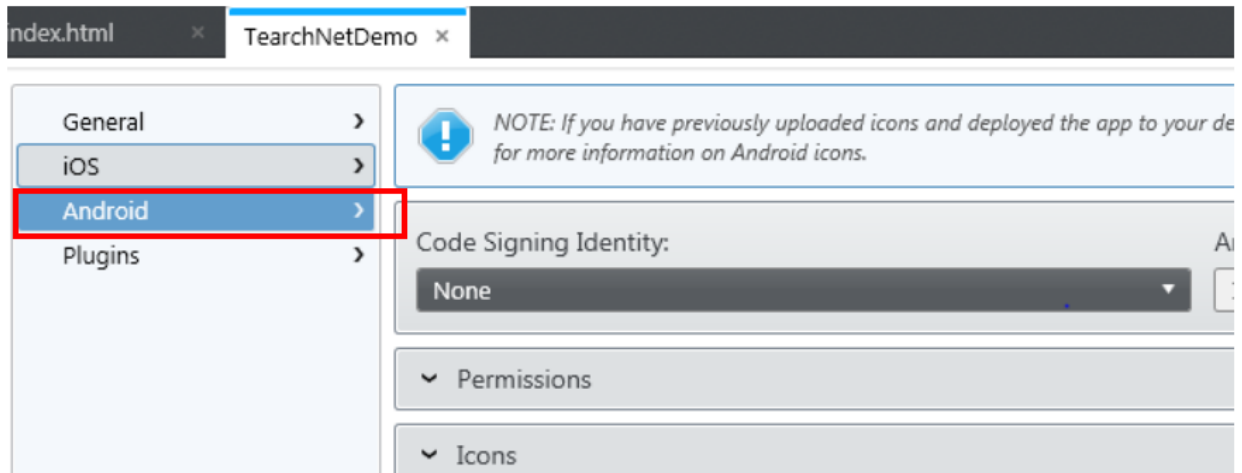
+ Create New

Alias	Subject	Issuer	Expires
Dhananjay Kumar	Dhananjay Kumar	Dhananjay Kumar	10/22/2034
Dhananjay Kumar 1	Dhananjay Kumar	Dhananjay Kumar	10/23/2033
Dhananjay Kumar 2	Dhananjay Kumar	Dhananjay Kumar	10/23/2033

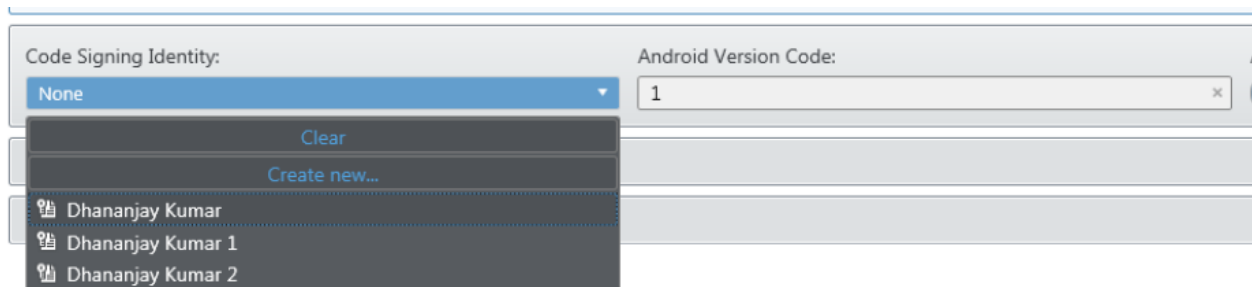
Import Export Details

After creating Self-signed identity right click on the project and select **properties** and in the properties windows select Android tab. Here you can set various application properties for android platform.

For Limited Circulation Only

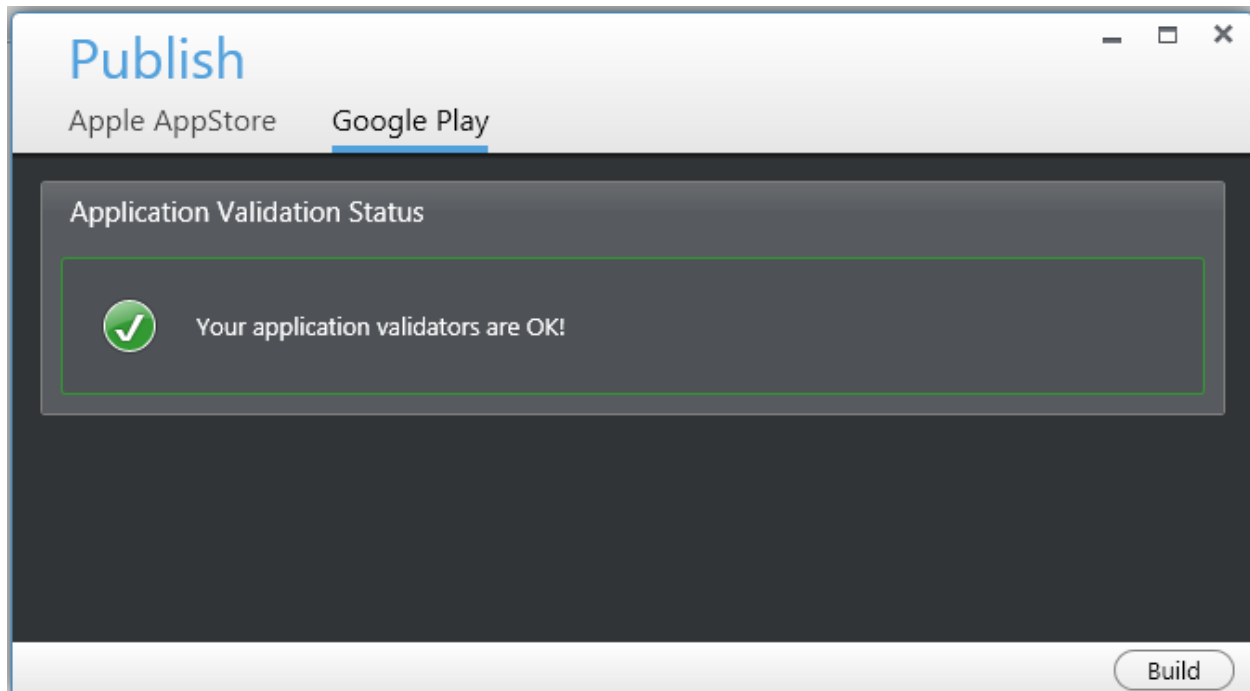


From the Code Signing Identity drop down, select any existing certificate to associate it with the application.

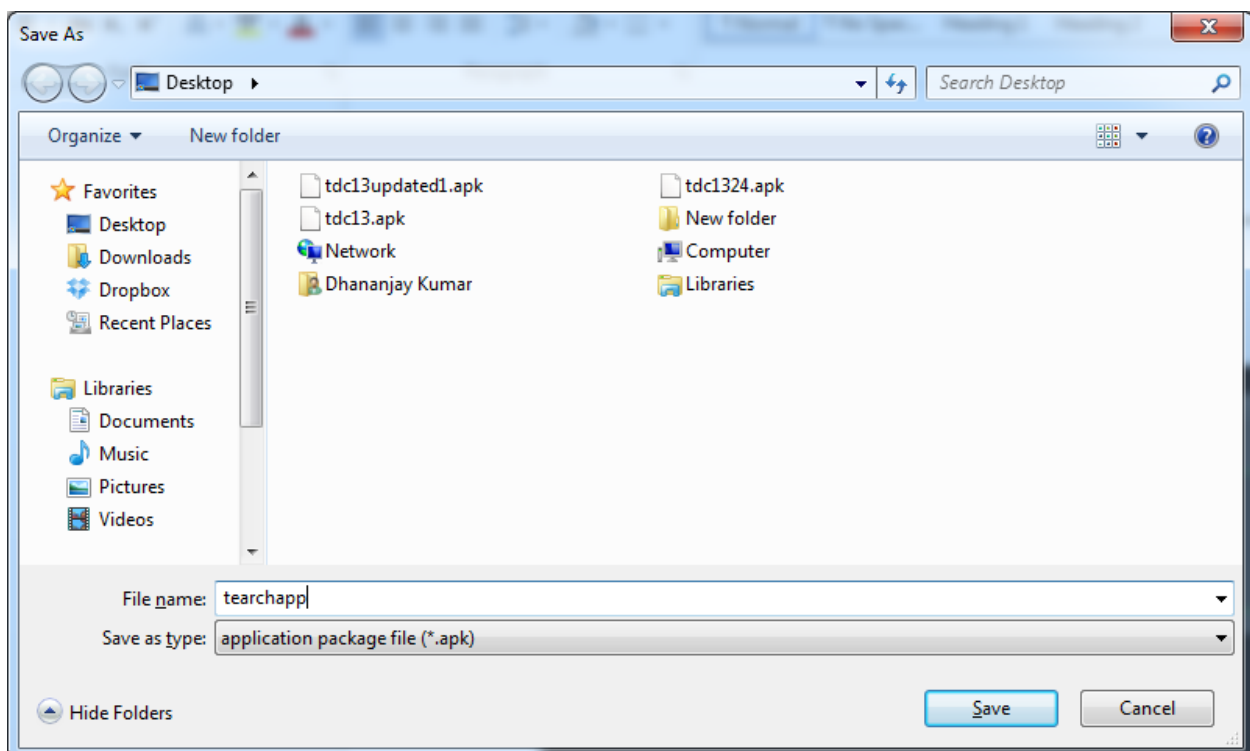


You can set icons, application permissions for Google Play here. After associating self-signed identity, right click on the project and select publish option. You will get Application Validation Status message as OK.

For Limited Circulation Only



Next, click on the Build button to create the package. Icenium will build the application in cloud and ask you to give a name to the apk package and save it locally.



For Limited Circulation Only

In this case we saved APK with name tearchapp. Now you can submit the APK file to Google Play to publish application.

For Limited Circulation Only

This page has been left blank intentionally