Container for advanced pages

Prerequisites

This tutorial assumes you know how to create a MobileFirst hybrid application and add Android and iPhone environments. For more information, see the following tutorials:

- Creating your first hybrid application (../../hello-world/creating-your-first-hybrid-application/)
- Previewing your application on iOS (../../hello-world/previewing-application-ios/)
- Previewing your applicaiton on Android (../../hello-world/previewing-application-android/)

Using this tutorial and its companion sample

This tutorial and its companion sample are intended for use with either the IBM® MobileFirst® Consumer Edition or the IBM MobileFirst Enterprise Edition. You cannot use the sample that is associated with this tutorial as is with the free IBM MobileFirst Developer Edition. This sample provides a WAR file that demonstrates the remote load functionality in this tutorial. This WAR file requires a servlet container, such as Apache Tomcat, WebSphere®Application Server Full Profile, or WebSphere Application Server Liberty Profile.

Topics Covered:

- Background of IBM Mobile Conference application
    - Mobile views
    - Form-based authentication - client
    - Form-based authentication - server
    - Simplified user registry and role configuration
- Migrating applications to IBM MobileFirst Platform
    - Working with IBM MobileFirst projects and environments
    - Running the sample on an Android environment
    - Running the sample on an iPhone environment
- IBM MobileFirst Platform Integration
    - Encrypted Cache
    - Camera support
- Proxy considerations

Background of IBM Mobile Conference application

- Mobile web application
- Server programming model
    - JavaServer™ Pages (JSP)
    - JavaServer Pages Standard Tag Library (JSTL)
    - JavaServer Pages Expression Language (EL)
- Client programming model
    - Dojox Mobile
    - Dojo Mobile Device Theming
- Security
    - Container managed
    - Java™ Enterprise Edition (Java EE) form-based login

Mobile views

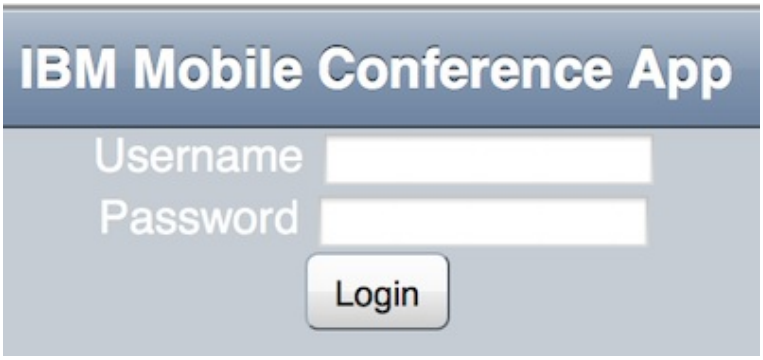
Dojo provides mobile widgets that developers can use to quickly generate mobile views.

```
<div id="locations" dojoType="dojox.mobile.ScrollableView" selected="true">
   <h1 dojoType="dojox.mobile.Heading" label="Locations" fixed="top"></h1>
   <div class="list-category" dojoType="dojox.mobile.RoundRectCategory">USA</div>
   <ul dojoType="dojox.mobile.RoundRectList">
      <c:forEach var="location" items="${applicationScope.locationRegistryList['usa']}"
>
         <li dojoType="dojox.mobile.ListItem" class="mblVariableHeight"
         url="./conferences.jsp?locationId=${location}"
         urlTarget="conferences" transition="slide">
            <div class="list-detail">
               <div>${locationMap[location].city}, ${locationMap[location].state}</div>
            </div>
         </li>
      </c:forEach>
   </ul>
</div>
```

JSTL provides a simple API for iterating over lists to generate Dojox Mobile list items dynamically on the server.

Form-based authentication - client

IBM Mobile Conference application requires users to be registered and authenticated with the enterprise that is hosting the application. The application is protected by a form-based login, by using a mobile web form that is contained within a Dojox Mobile view.



```
<div dojoType="dojox.mobile.View">
      <c:set var="conferences" scope="session" value=""></c:set>
      <h1 style="text-align: center;" dojoType="dojox.mobile.Heading">IBM Mobile Conference App</h1
>
      <form action="j_security_check" method="post" name="loginForm">
        <div class="logonClass">
          <div>
            <span>Username </code> <input id="username" dojoType="dojox.mobile.TextBox"
              name="j_username" />
          </div>
          <div>
            <span>Password </code> <input id="password" type=password name="j_password"
              dojoType="dojox.mobile.TextBox"> </input>
          </div>
          <div style="text-align: center;">
            <input id="login" type="submit" class="mblButton" value="Login"/>
          </div>
        </div>
      </form>
</div>
```

Form-based authentication - server

The IBM Mobile Conference application is configured to use Java Enterprise Edition Role Support.

- White list or pattern-based URIs of secured resources
- Support for logout in addition to Java Enterprise Edition standard logout functionality

```
<security-role>
   <role-name>worklightadmin</role-name>
</security-role>
<security-constraint>
   <display-name>SecurityConstraint</display-name>
   <web-resource-collection>
      <web-resource-name>Conference</web-resource-name>
      <url-pattern>/common/location.jsp</url-pattern>
      <url-pattern>/common/confs.jsp</url-pattern>
      <url-pattern>/common/presenter.jsp</url-pattern>
      <url-pattern>/common/sessions.jsp</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>worklightadmin</role-name>
   </auth-constraint>
</security-constraint>
<login-config>
   <auth-method>FORM</auth-method>
   <form-login-config>
      <form-login-page>/common/logon.jsp</form-login-page>
      <form-error-page>/common/logonError.jsp</form-error-page
>
   </form-login-config>
</login-config>
```

```
<div style="position: relative; float: left;">
   <form method="post" action="conferenceLogout.jsp" name="logout"
>
      <span data-dojo-type="dojox.mobile.ToolBarButton"
         data-dojo-props='onClick:function(){
            document.forms["logout"].submit();
         },
         transition:"slide"'>Logout</code>
      <input type="hidden" name="logoutExitPage" value="/">
   </form>
</div>
```

Simplified user registry and role configuration

Make these changes in your **server.xml** file, depending which server you use:

MobileFirst Development Server (Liberty):

```
<server description="new server">
   <featureManager>
      <feature>servlet-3.0</feature>
      <feature>jndi-1.0</feature>
      <feature>jdbc-4.0</feature>
      <feature>localConnector-1.0</feature>
      <feature>restConnector-1.0</feature>
      <feature>jsp-2.2</feature>
      <feature>appSecurity-1.0</feature>
      <feature>ssl-1.0</feature>
   </featureManager></p>
   ...
   <httpEndpoint host="*" httpPort="10080" httpsPort="10443" id="defaultHttpEndpoint">
      <tcpOptions soReuseAddr="true"/></p>
   </httpEndpoint>
   ...
   <applicationMonitor updateTrigger="mbean"/></p>
   ...
   <basicRegistry id="worklight" realm="worklightRealm">
      <user name="demo" password="demo"/>
      <user name="monitor" password="demo"/>
      <user name="deployer" password="demo"/>
      <user name="operator" password="demo"/>
      <user name="admin" password="admin"/>
   </basicRegistry>
   ...
   <webApplication contextRoot="conference" id="ContainerForAdvancedPagesWar" location="module_45_1_Int
egrateExistingWebContentWar.war" name="ContainerForAdvancedPagesWar">
      <application-bnd>
         <security-role name="worklightadmin">
            <user name="admin"/>
         </security-role>
      </application-bnd>
   </webApplication>
   <application context-root="/ContainerForAdvancedPages" id="ContainerForAdvancedPages" location="Contain
erForAdvancedPages.war" name="ContainerForAdvancedPages" type="war">
      <classloader commonLibraryRef="worklight-6.3.0">
         <privateLibrary>
            <fileset dir="${wlp.user.dir}/shared/resources" includes="org.hsqldb.hsqldb_2.2.5.jar"/>
         </privateLibrary>
      </classloader>
   </application>
   ...
</server>
```

Tomcat server:

```
<GlobalNamingResources>
<!-- Editable user database that can also be used by
     UserDatabaseRealm to authenticate users
-->
<Resource auth="Container"
    description="User database that can be updated and saved"
    factory="org.apache.catalina.users.MemoryUserDatabaseFactory
"
    name="UserDatabase"
    pathname="conf/tomcat-users.xml"
    type="org.apache.catalina.UserDatabase"/>
</GlobalNamingResources>
```

Add role and user in **tomcat-users.xml**:

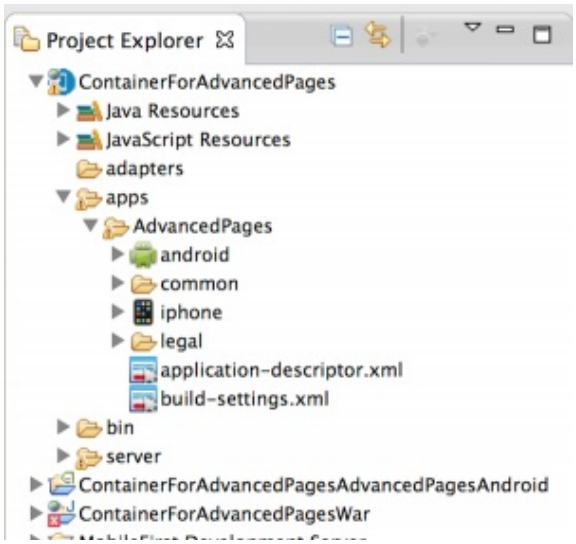
```
<role rolename="worklightadmin"/>
<user username="tomcat" password="tomcat" roles="worklightadmin"/>
```

Migrating applications to IBM MobileFirst Platform

By using mobile web technology, you can deploy applications to the widest variety of devices. The presence of application stores (for example: Apple App Store and Google Play) adds a dimension where the hosting and marketing of these applications makes broader-reaching applications less relevant. IBM MobileFirst Platform provides the solution to building cross platform applications that can be distributed through the application stores by using the hybrid application programming model. In the hybrid model, developers typically package the application HTML, CSS, and JavaScript™ code as part of the application that is deployed to the application store. In this tutorial, you see the remote loading of dynamic content capability, where the HTML, CSS, and JavaScript code is hosted externally from the natively packaged hybrid.

Working with IBM MobileFirst projects and environments

This tutorial is accompanied by a MobileFirst Project called **ContainerForAdvancedPages**, which contains an **AdvancedPages** hybrid application. This application already has iPhone and Android environments added to it.



MobileFirst Development Server

Running the sample on an Android environment

Make the following changes in **AdvancedPages/android/nativeResources/src/com/AdvancedPages/AdvancedPages.java**:

```
public void onInitWebFrameworkComplete(WLInitWebFrameworkResult result){
    if (result.getStatusCode() == WLInitWebFrameworkResult.SUCCESS) {
        WL.getInstance().hideSplashScreen();
        super.loadUrl("http://localhost:10080/conference/common/location.jsp");
    }
    else {
        handleWebFrameworkInitFailure(result);
    }
}
```

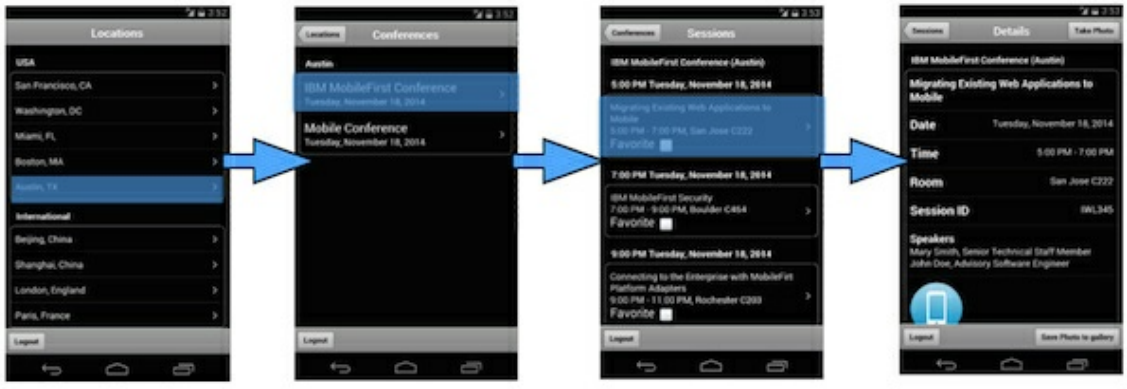
If running on Tomcat server, change port to 8080.

1. If running on an actual Android device, replace `localhost` with your machine's public IP address. Make sure the device and your server are on the same network.
2. Right-click the WAR and select **Run As > Run on Server** and choose the MobileFirst Development Server.
3. Right-click the hybrid application and select **Run As > Run on MobileFirst Development Server**.

   > Note: Due to same domain restrictions of the browser (web view), the remotely loaded application and MobileFirst Development Server must either be co-located on same host and port, or have a common proxy host and port.

4. Right-click the Android project and select **Run As > Android Application**.

After the application is installed, authenticate with username / password: "admin" / "admin", or whatever credentials you specified in your server.xml file. Each view for the IBM Mobile Conference app is loaded through Ajax XHR where each view is generated by the JSP responsible for each distinct view.



Running the sample on an iPhone environment

Make the following changes in **AdvancedPages/iphone/nativeResources/Classes/AdvancedPages.m**:

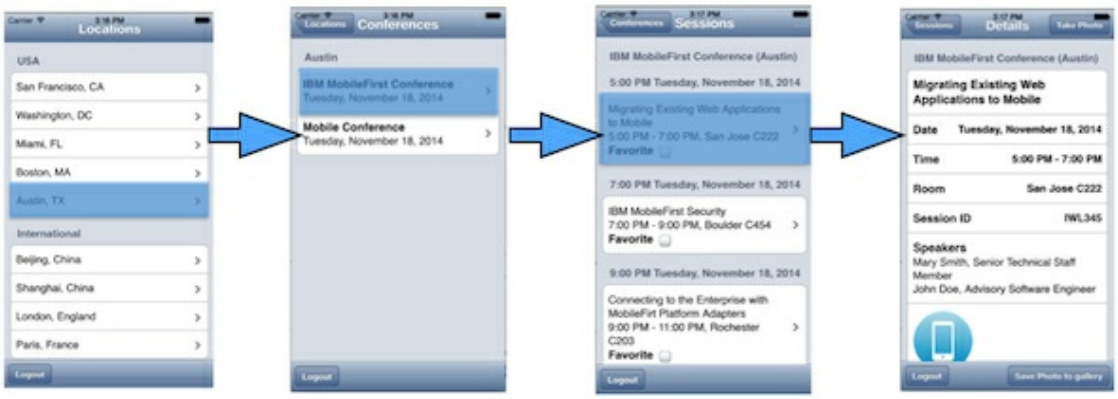
```
-(void)wlInitDidCompleteSuccessfully
{
    [[WL sharedInstance] hideSplashScreen];
    UIViewController* rootViewController = self.window.rootViewController;
    // Create a Cordova View Controller
    CDVViewController* cordovaViewController = [[CDVViewController alloc] init] ;
    cordovaViewController.startPage = [[WL sharedInstance] mainHtmlFilePath];
    // Adjust the Cordova view controller view frame to match its parent view bounds
    cordovaViewController.view.frame = rootViewController.view.bounds;
    // Display the Cordova view
    [rootViewController addChildViewController:cordovaViewController];
    [rootViewController.view addSubview:cordovaViewController.view];
    NSString* remoteURLStr= @"http://localhost:10080/conference/common/location.jsp";
                                        NSURL* remoteURL = [NSURL URLWithString: remoteURLStr];
                                        NSURLRequest* request= [NSURLRequest requestWithURL:remoteU
RL];
                                        [cordovaViewController.webView loadRequest:request];
}
```

If running on Tomcat server, change port to 8080.

1. Right-click the WAR and select **Run As > Run on Server** and choose the MobileFirst Development Server.
2. Right-click the hybrid application and select **Run As > Run on MobileFirst Development Server**.

   > Note: Due to same domain restrictions of the browser (web view), the remotely loaded application and MobileFirst Development Server must either be co-located on same host and port, or have a common proxy host and port.

3. Right-click the iPhone folder and select **Run As > Xcode project**.
4. Run the Xcode project on an iOS simulator or device running iOS 7 or higher.

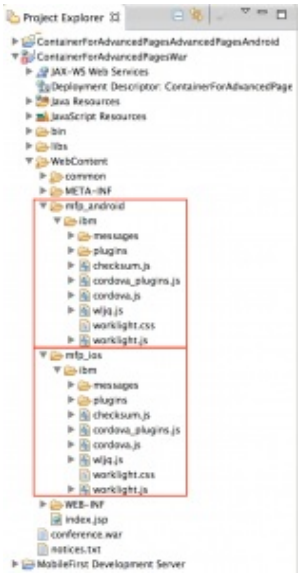


Extending IBM Mobile Conference app to IBM MobileFirst Platform

IBM MobileFirst Platform Integration

IBM MobileFirst Platform provides client libraries for hybrid application development. Although the APIs are platform-neutral, they are often backed by platform-specific implementations. Because the conference app is served remotely, these client-side libraries must be packaged with the conference app (WAR). Each platform-specific implementation is in its own folder structure in the web application.

When the IBM Mobile Conference application completes the authentication process, the mobile device client initializes the device with the server by using `WL.Client.init()`. This provides a flexible model for application integrity validation, application updates, and other features.

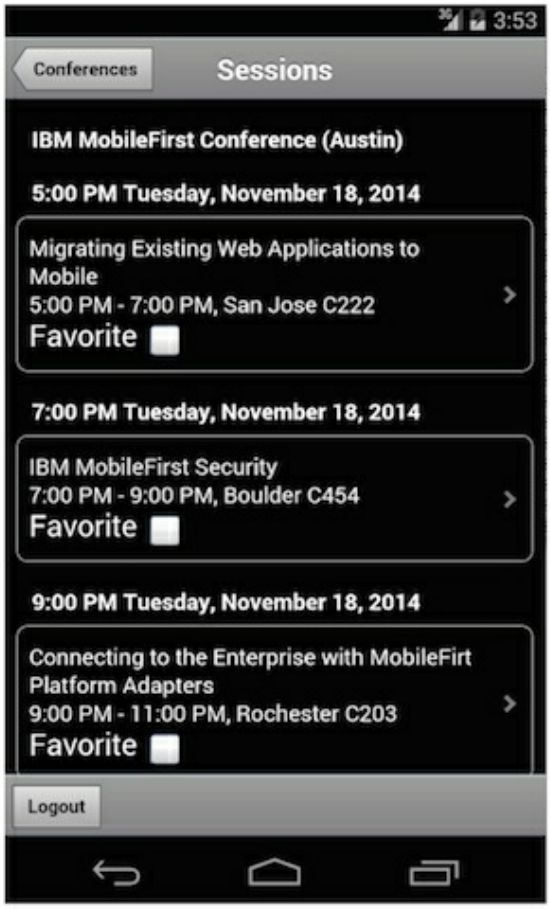



```
<c:choose>
   <c:when test="${fn:contains(iOSClient, 'Android')}">
      <link rel="stylesheet" type="text/css"
         href="dojox/mobile/themes/android/android.css"></link>
   </c:when>
   <c:otherwise>
      <link rel="stylesheet" type="text/css"
         href="dojox/mobile/themes/iphone/iphone.css"></link>
   </c:otherwise>
</c:choose>
</head></p>
<body id="content" onload="WL.Client.init({onSuccess:function success(){ console.info('WL init success...'); }, conn
ectOnStartup:false});">
   <c:set var="locationMap" value='${applicationScope.locationRegistry}' scope="session"/>
...
```

```
<c:choose>
    <c:when test="${fn:contains(iOSClient, 'Android')}">
        <script>
        // Define WL namespace.
        var WL = WL ? WL : {};
        /**
         * WLClient configuration variables.
         * Values are injected by the deployer that packs the gadget.
         */
         WL.StaticAppProps = {
   "APP_DISPLAY_NAME": "AdvancedPages",
   "APP_ID": "AdvancedPages",
   "APP_SERVICES_URL": "\/apps\/services\/",
   "APP_VERSION": "1.0",
   "ENVIRONMENT": "android",
   "LOGIN_DISPLAY_TYPE": "embedded",
   "WORKLIGHT_NATIVE_VERSION": "3826723549",
   "WORKLIGHT_PLATFORM_VERSION": "6.3.0.0",
   "WORKLIGHT_ROOT_URL": "\/apps\/services\/api\/AdvancedPages\/android\/
"
};
</script>
...
    </c:when>
    <c:otherwise>
    <script>
        // Define WL namespace.
        var WL = WL ? WL : {};
        /**
         * WLClient configuration variables.
         * Values are injected by the deployer that packs the gadget.
         */
         WL.StaticAppProps = {
   "APP_DISPLAY_NAME": "AdvancedPages",
   "APP_ID": "AdvancedPages",
   "APP_SERVICES_URL": "\/apps\/services\/",
   "APP_VERSION": "1.0",
   "ENVIRONMENT": "iphone",
   "LOGIN_DISPLAY_TYPE": "embedded",
   "WORKLIGHT_NATIVE_VERSION": "270878231",
   "WORKLIGHT_PLATFORM_VERSION": "6.3.0.0",
   "WORKLIGHT_ROOT_URL": "\/apps\/services\/api\/AdvancedPages\/iphone\/"
};
</script>
...
    </c:otherwise>
</c:choose>
```

Encrypted Cache

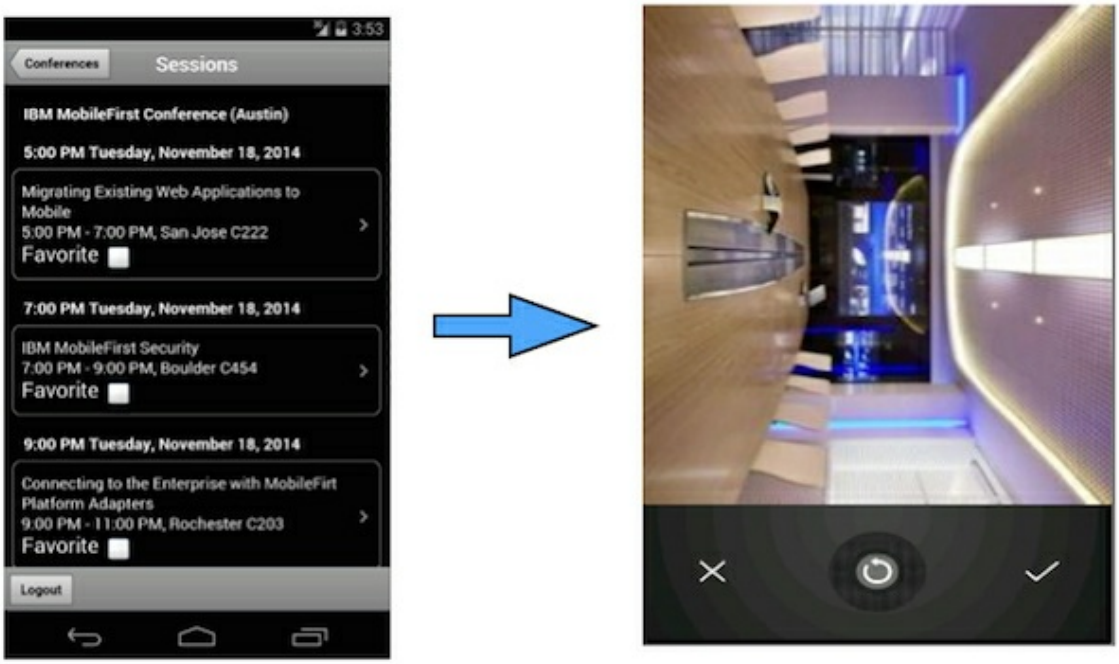
The Encrypted Offline Cache (EOC) API is used for securely storing data by using HTML5 local cache. The Conference app uses EOC to save a list of favorite sessions in a secure way.




```
function __writeEOC(mykey, mydata, callback) {
    WL.EncryptedCache.write(mykey, mydata, function() {
        callback();
    }, function() {
        alert('An error occurred writing to the encrypted cache');
    });
}
...
function saveToCache(key, value) {
    if (isEOCOpen == false) {
        __openEOC(function() {
            __writeEOC(key, value, function() {
            });
        });
    } else {
        __writeEOC(key, value, function() {
        }, function() {
            alert('Unable to write to encrypted cache');
        });
    }
}
```

Camera support

Use the IBM MobileFirst Client API to extend the application to take photos of the conference. The application has a social aspect, where users can view photos from the other conference attendees.



Proxy considerations

IBM MobileFirst Platform is compatible with most known proxy solutions.

WebSphere HTTP Plug-in support: Requires single shared `plugin-cfg.xml` for routing requests to both the MobileFirst Development Server and the WebSphere Application Server that is hosting the IBM Mobile Conference App.

Apache Web Server support: Requires mod_proxy module to be loaded by the web server.

Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/ContainerForAdvancedPagesProject.zip) the Studio project.