Creating your first hybrid application

Overview

This tutorial describes all the steps from creating a new MobileFirst project and application to building it. The tutorial explains the structure of the new project and all its components, and the concept of "Single-Page Application" (SPA) that MobileFirst applications are based on. Finally, the tutorial shows how to preview the newly created application. This aspect is covered in more detail in specific tutorials.

This tutorial covers the following topics:

- Creating a hybrid MobileFirst application
- Structure of the new application
- Building an application
- Previewing an application
- Sample application
- Tutorials to follow next

Creating a hybrid MobileFirst application

Using the CLI

1. If you have installed the CLI (../../advanced-client-side-development/using-cli-create-build-manage-project-artifacts/), in the terminal, create a project named `HelloWorldProject`.

   ```
   $ mfp create HelloWorldProject
   A MobileFirst Project was successfully created at /Users/MyUser/Dev/HelloW
   orldProject
   ```

2. Go to the context of your newly created project.

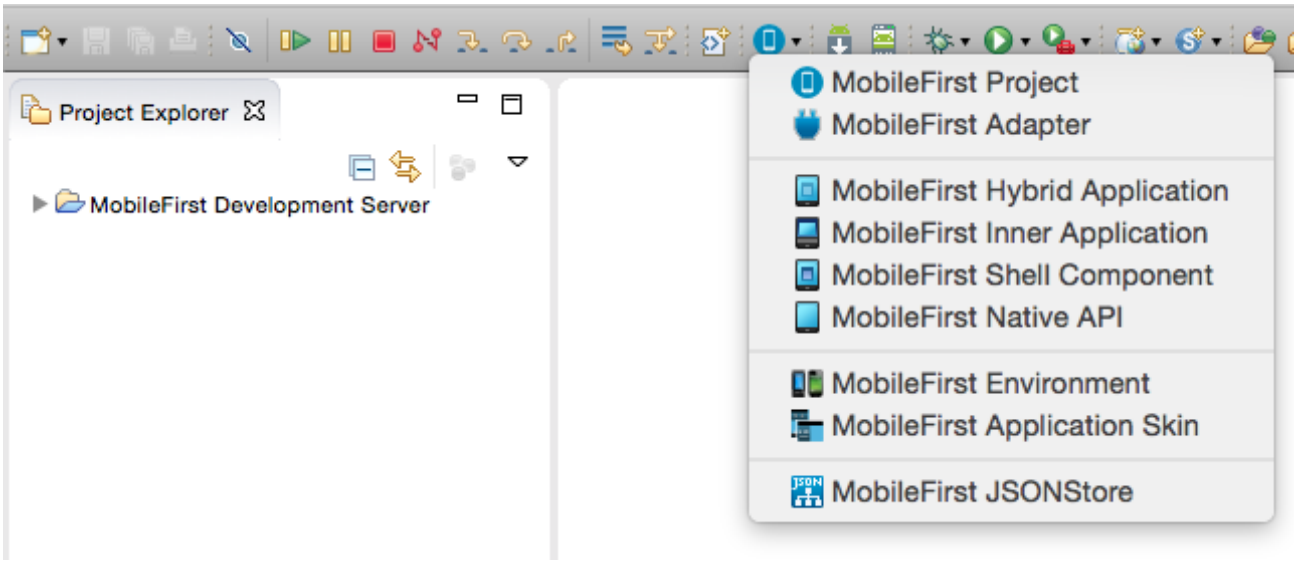
   ```
   $ cd HelloWorldProject
   ```

3. Define a hybrid application named `HelloWorld`.

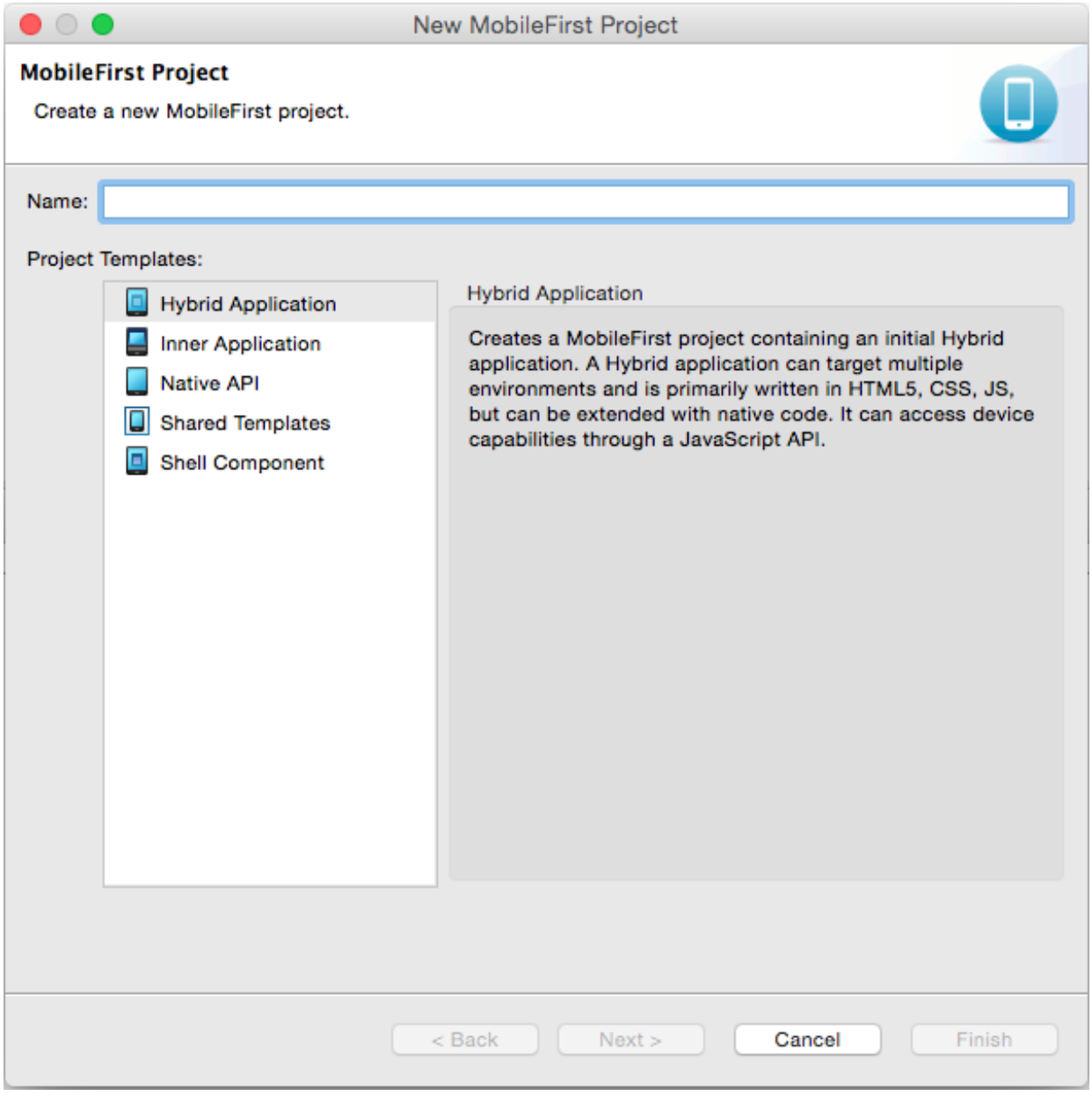
   ```
   $ mfp add hybrid HelloWorld
   A new Hybrid App was added at /Users/MyUser/Dev/HelloWorldProject/apps/Hel
   loWorld<br />
   ```

Using the Studio

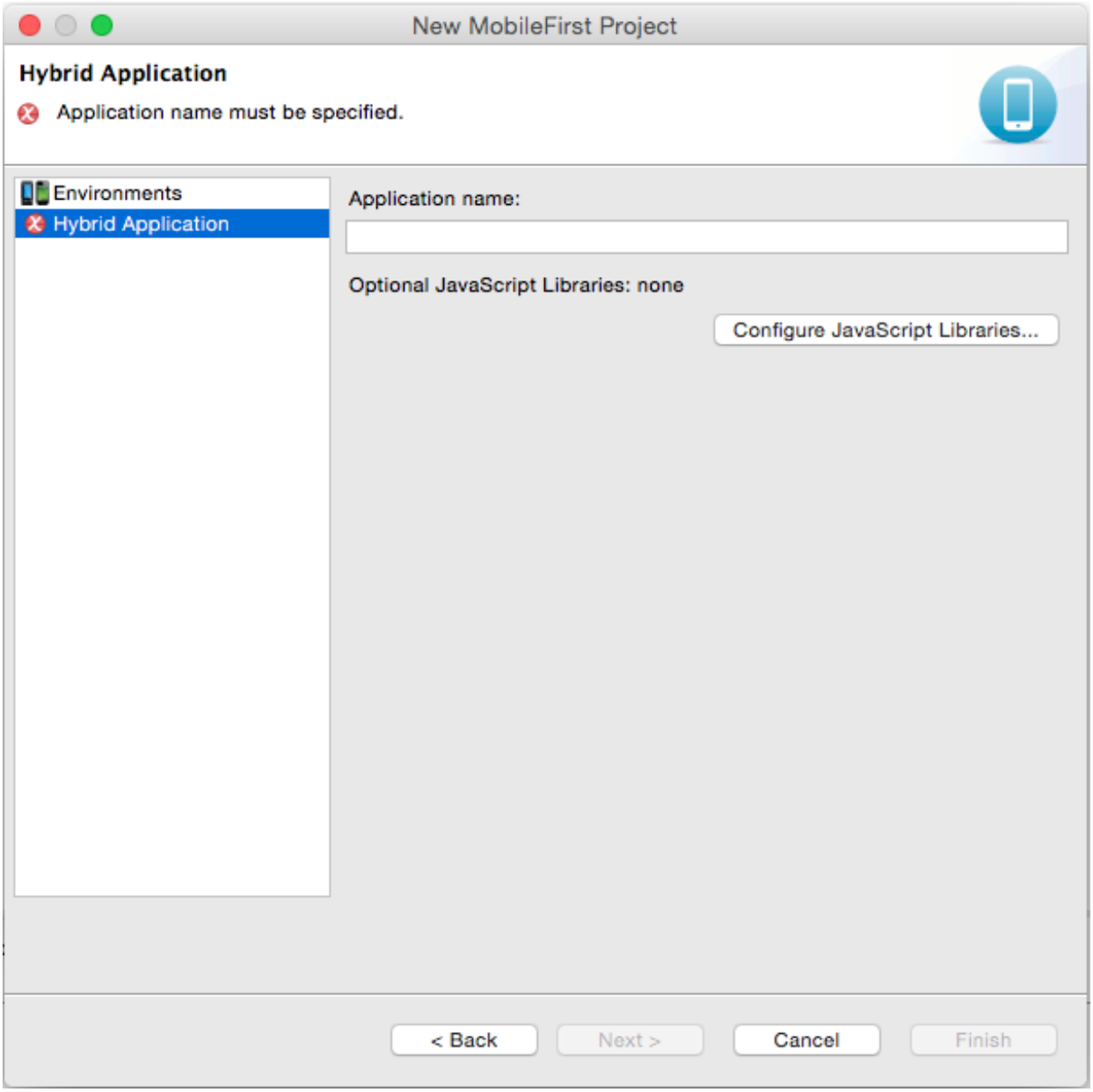
1. In MobileFirst Studio, select **File > New > MobileFirst Project** to create a new MobileFirst Project from the top toolbar.



2. Give your project a name, for example *HelloWorldProject*, and select the **Hybrid Application** template.

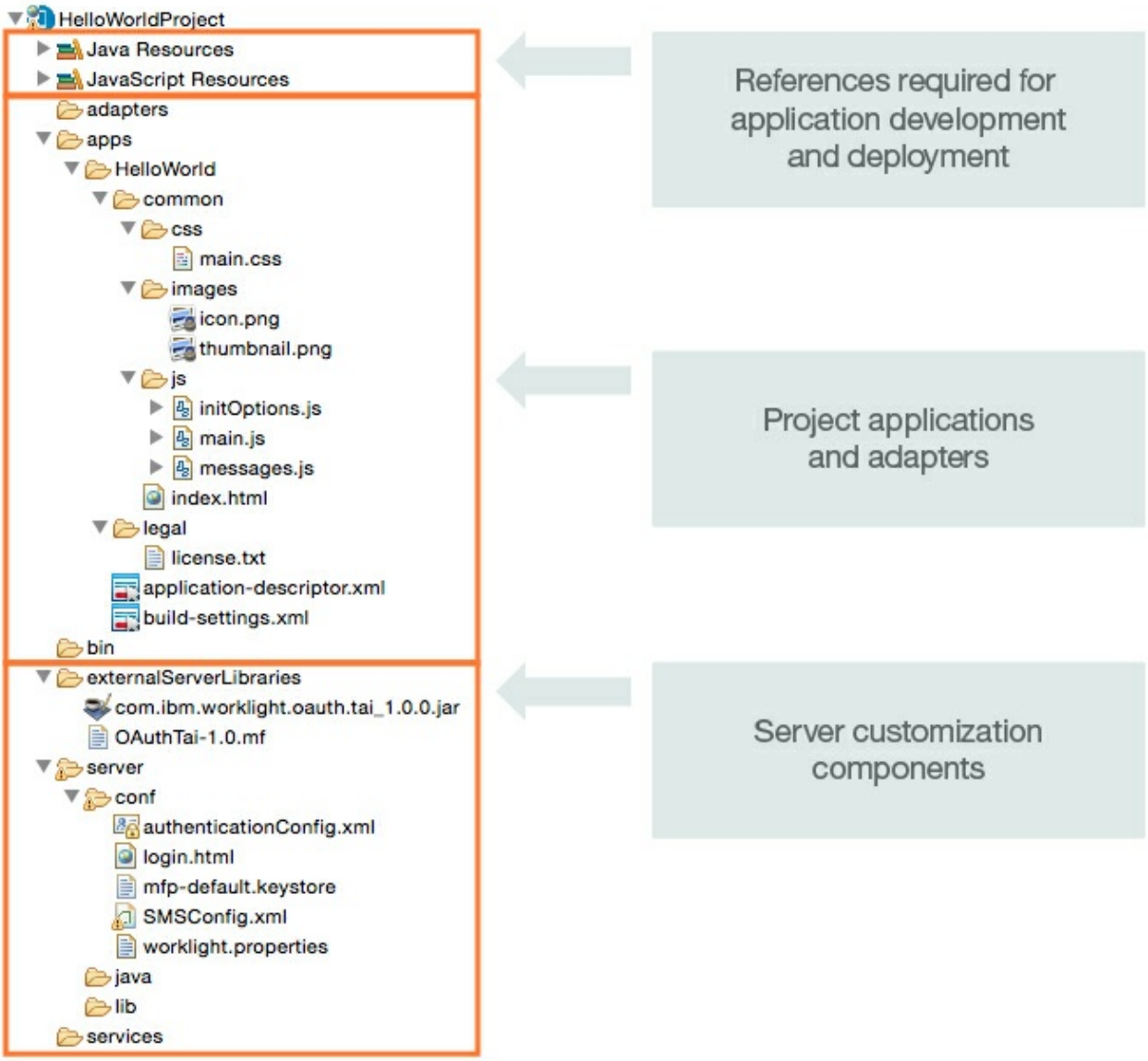


3. Give the application a name, for example *HelloWorld*. You can add JavaScript frameworks to your project in this screen. Click **Finish** when done.



To learn more about adding frameworks to applications, read the Client-side development basics > Working with UI frameworks (../../client-side-development-basics/working-ui-frameworks/) tutorial.

Structure of the new application



Environment files

Common

The default environment is called **common**. The **common** environment contains all the resources that are shared between environments:

- **index.html**: main HTML file
- **css**
  - main.css: main application CSS file.
- **images**: Default MobileFirst images for the common environment
- **js**
  - initOptions.js: Contains initialization options for the application
  - main.js: The main JavaScript file for the application
  - messages.js: A JSON object that holds all app messages. Can be used as the source for translation.
- **legal**: A folder that holds all the legal docs.
- **application-descriptor.xml**: Contains the application metadata.
- **build-settings.xml**: Contains configuration options for minification and concatenation.

Other environments

To add an environment, right-click the apps folder and select **New > MobileFirst environment**, or use the top toolbar icon.
The resources of the new environment have the following relationship with the common resources:

- **images** – Overrides the common images when both have the same name.
- **css** – Extends, overrides, or both, the common CSS files.
- **js** – Extends the common application instance JS object. The environment class extends the common app class.
- **index.html** – Optional HTML file that overrides the common HTML code when both have the same name.

Server files

- **externalServerLibraries**: Contains the libraries to be placed in external service servers and used for access token validation (by the service).
- **server**: Contains files that are used for server-side customization of a project:
  - **conf**: contains
    - **authenticationConfig.xml**: Defines authentication realm and security tests.
    - **default.keystore**: A default SSL certificate that is provided by the project.
    - **login.html**: Presents a login form for web environments and the MobileFirst Operations Console.
  - **SMSConfig.xml**: Defines SMS Gateways.
  - **worklight.properties**: Defines the properties that are used by MobileFirst Server.
- **java**: Used to hold Java classes that will be compiled and deployed to a MobileFirst Server instance after the application is built. You can place your custom Java code here.
- **lib**: Used for JAR files that are deployed to the server.
- **services**: Contains any back-end services that were discovered.

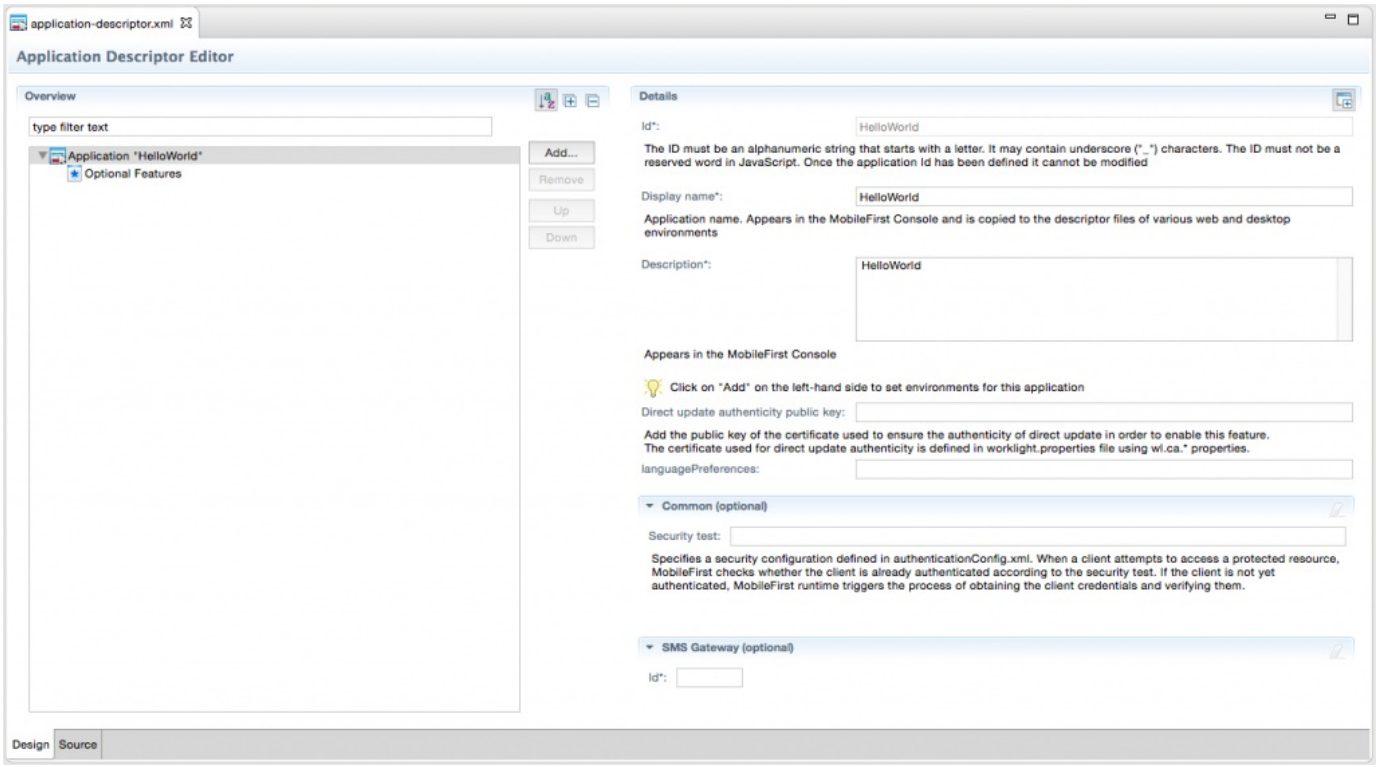
The bin folder

The `bin` folder contains project artifacts that are deployed to MobileFirst Server.
MobileFirst Studio deploys those artifacts to the embedded MobileFirst Development Server automatically as a part of the build process.

- `.wlapp` files are application bundles.
- `.wladapter` files are adapters.
- `.jar` and `.war` files are server customization files that contain `worklight.properties`, `authenticationConfig.xml`, and custom Java code.

The application-descriptor.xml file

The application descriptor is an XML file that stores the metadata for an application.
You can edit this file in the Design or Source editors (in Studio), or with your preferred editor if you use the CLI.



The file is based on the W3C Widget Packaging and Configuration standard and contains application properties that are used at build time.
You can specify the application description, details about the author, and the thumbnail image to be displayed in the MobileFirst Operations Console.

```
    <!-- Attribute "id" must be identical to application folder name -->
    <application id="HelloWorld" platformVersion="6.2.0.00.20140701-1500
"
    xmlns="http://www.worklight.com/application-descriptor"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <displayName>HelloWorld</displayName>
        <description>HelloWorld</description>
        <author>
            <name>application's author</name>
            <email>application author's e-mail</email>
            <copyright>Copyright My Company</copyright>
            <homepage>http://mycompany.com</homepage>
        </author>
        <mainFile>index.html</mainFile>
        <thumbnailImage>common/images/thumbnail.png</thumbnailImage>
        <features>
        </features>
    </application>
```

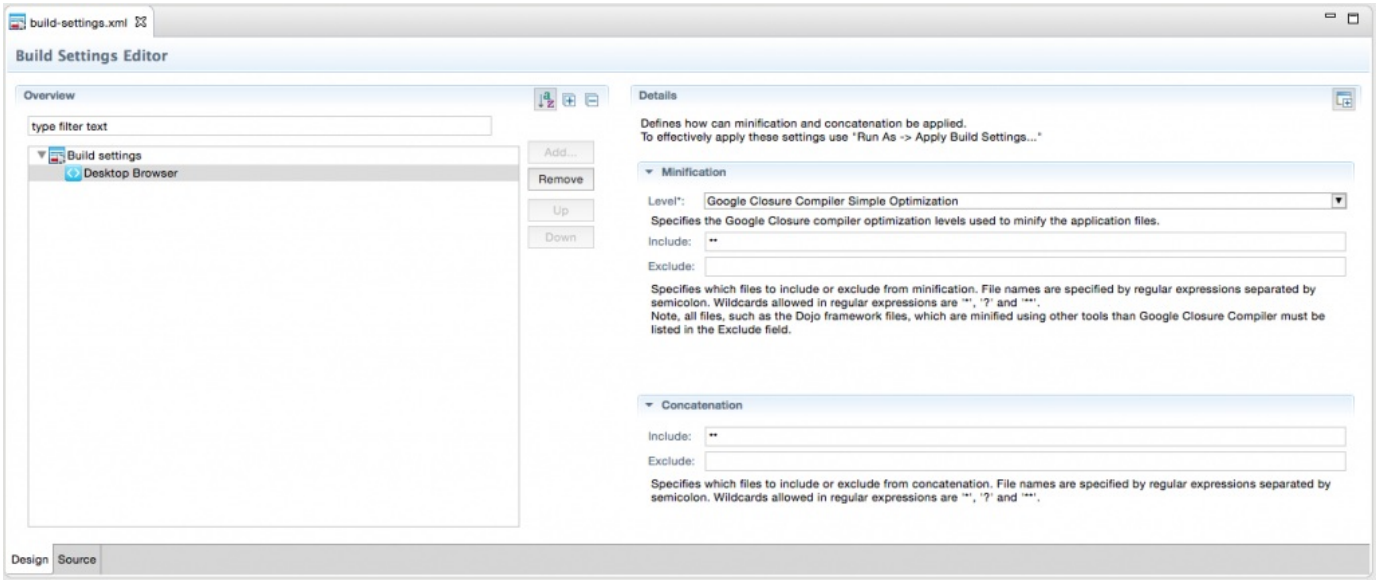
Environment-specific information is inserted automatically as new environments are added to the MobileFirst project:

```
    <iphone bundleId="com.HelloWorld" version="1.0">
        <worklightSettings include="false"/>
        <security>
            <encryptWebResources enabled="false"/>
            <testWebResourcesChecksum enabled="false" ignoreFileExtensions="png, jpg, jpeg, gif, mp4,
mp3"/>
        </security>
    </iphone>
    <android version="1.0">
        <worklightSettings include="false"/>
        <security>
            <encryptWebResources enabled="false"/>
            <testWebResourcesChecksum enabled="false" ignoreFileExtensions="png, jpg, jpeg, gif, mp4,
mp3"/>
            <publicSigningKey/>
            <packageName/>
        </security>
    </android>
```

The build-settings.xml file

The build settings file is an XML file that contains configuration options for minification and concatenation of the Desktop Browser and Mobile Web environment web resources.
You can edit this file in the Design or Source editors (in Studio), or with your preferred editor if you use the CLI.

By using minification on specific web resources, you can reduce the size of JavaScript and CSS files in the application.
In addition, you can use concatenation of the web resources to improve the start time of the application.



The index.html file

At application run time, the main HTML document cannot be replaced by another HTML document.
The default application HTML template complies with HTML5 standard markup, but any other DOCTYPE can be specified.

The MobileFirst client-side framework uses the jQuery library for internal functions. By default, the `$` char is assigned to the internal jQuery in the main HTML file (see below). If a different jQuery version is required or if jQuery is not required in the application, this line (#12) can be removed.

The MobileFirst client framework initialization is bound to the `onload` event specified in the `initOptions.js` file. For more information about initialization options, see the user documentation.

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="UTF-8">
        <title>HelloWorld</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, mini
mum-scale=1.0, user-scalable=0">
        <!--
            <link rel="shortcut icon" href="images/favicon.png">
            <link rel="apple-touch-icon" href="images/apple-touch-icon.png">
        -->
        <link rel="stylesheet" href="css/main.css">
        <script>window.$ = window.jQuery = WLJQ;</script>
    </head>
    <body style="display: none;">
        <!--application UI goes here-->
        Hello World
        <script src="js/initOptions.js"></script>
        <script src="js/main.js"></script>
        <script src="js/messages.js"></script>
    </body>
</html>
```

The initOptions.js file

The initialization options file contains MobileFirst framework initialization settings.
It is also responsible for initializing the MobileFirst framework after the element finishes loading.

By default, the MobileFirst application starts in offline mode (the application does not attempt to connect to MobileFirst Server).
To connect to MobileFirst Server, use `WL.Client.connect()`.

Some default initialization options are documented in the file itself. The entire set of options is documented in the Reference topic for the API method "`WL.Client.init`", in the user documentation.

The main.js file

When you create an application, a `main.js` file is created and holds its JavaScript portion.
It contains a `wlCommonInit()` function which is invoked automatically after the MobileFirst framework initialization finishes. Application initialization code can be implemented herein.

This function is used in environment-specific JavaScript files to have a common initialization starting point. Additional details are provided in subsequent tutorials.

As discussed previously, the MobileFirst application starts in offline mode by default.
To begin communicating with MobileFirst Server, follow the instructions provided in the default `wlCommonInit()` function:

```
function wlCommonInit(){
    /*
     * Use of WL.Client.connect() API before any connectivity to MobileFirst Server is required.
     * Call this API only once, before any other WL.Client methods that communicate with MobileFirst Server.
     * Remember to specify and implement the onSuccess and onFailure callback functions for WL.Client.connect(), e.g:
     *
     *    WL.Client.connect({
     *      onSuccess: onConnectSuccess,
     *      onFailure: onConnectFailure
     *    });
     *
     */
    // Common initialization code goes here
}
```

Single DOM model

MobileFirst hybrid applications use a single DOM model.

The single DOM model means that navigation between various HTML files must not be implemented by using hyperlinks or by changing the `window.location` property.
Instead, you can implement a multipage interface by loading an external HTML file content through Ajax requests and by injecting it into an existing DOM.

Such implementation is required because the main application HTML file loads the MobileFirst client-side JavaScript framework files. If the webview navigates away from one HTML file to another, the JavaScript context and loaded scripts are lost.

Most JavaScript UI frameworks available today (for example, jQuery Mobile, Dojo Mobile, Sencha Touch) provide an extensive range of APIs to achieve the required multi-page navigation.

This module explains the principles of a single-page application.
Principles of multipage applications that are built with a single DOM model are explained in other tutorials.

Building an application

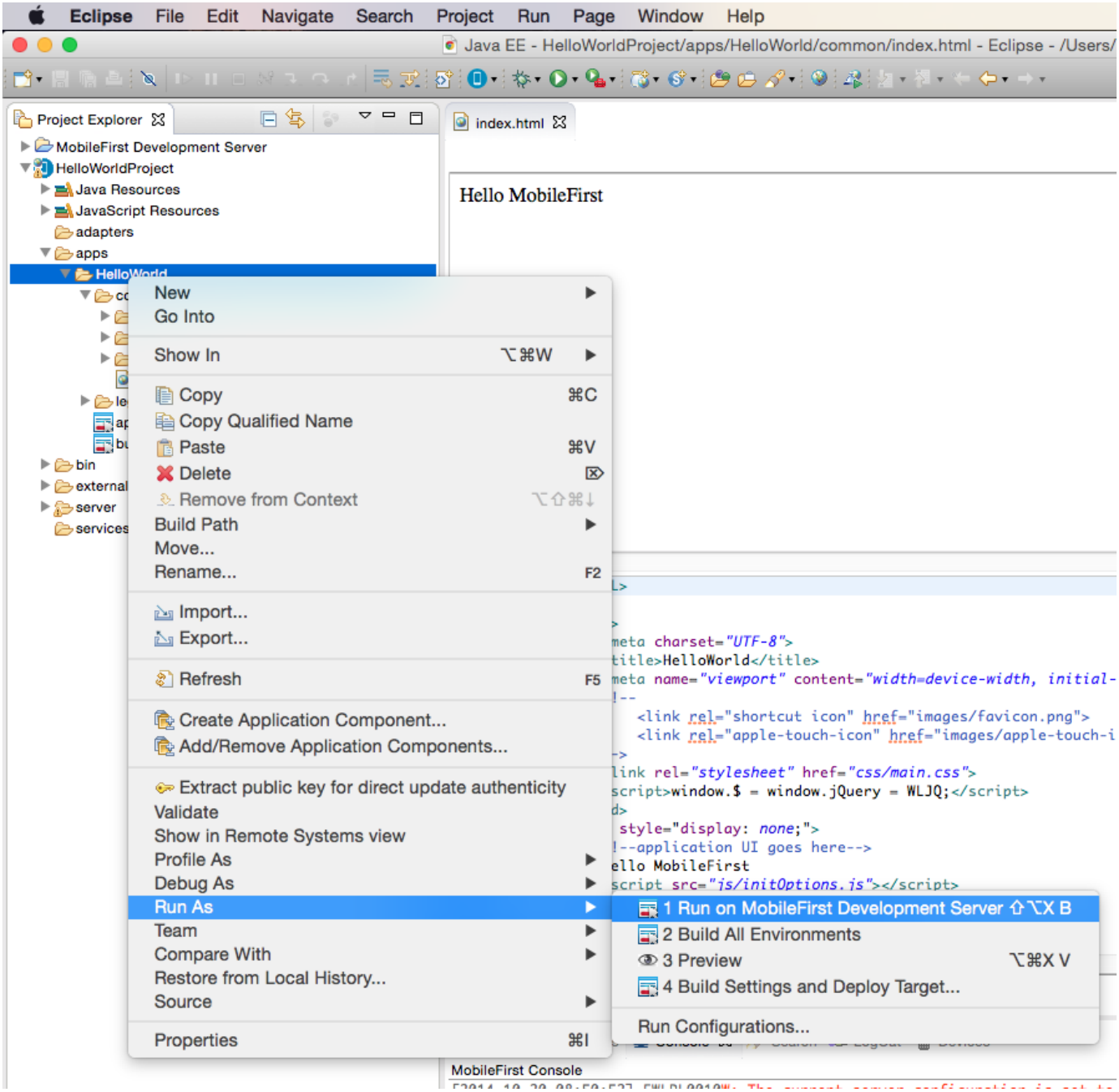
CLI

From the CLI, use `mfp build` to build and `mfp deploy` to deploy the application on the test server. You can also use `mfp bd` to combine both build and deploy in the same command.
After the build completes, the application is available for preview in the MobileFirst Operations Console.

Studio

To build an application from the Studio, right-click the application name and select **Run As > Run on MobileFirst Development Server**.

While the application is being built and deployed, you can monitor the progress in the Eclipse Console view. After the build completes, the application is available for preview in the MobileFirst Operations Console.



MobileFirst Operations Console

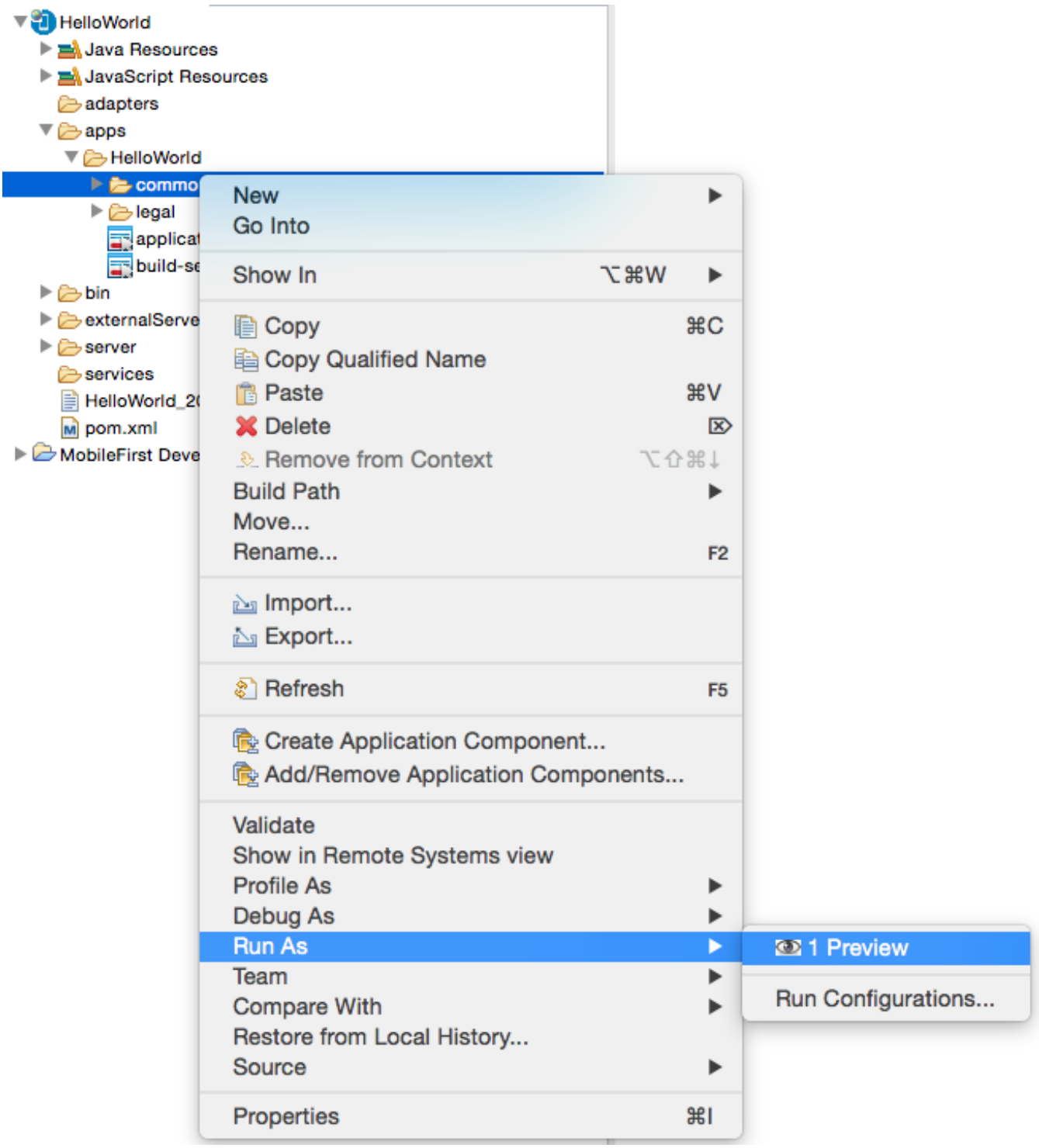
Learn about the MobileFirst Operations Console from the Introduction to MobileFirst Operations Console (../../hello-world/mobilefirst-console/) tutorial.

Previewing an Application

- To preview the application's common resources from the CLI, go to the common folder: `$ cd apps/HelloWorld/common/` and use `$ mfp preview` to start the preview.
- To preview the application's common resources from the Studio, right-click the `common` folder and choose **Run As > Preview**.

If you are presented with a login screen, use admin/admin as the username/password.

Learn more about previewing from the Previewing the web resources of your application (../../hello-world/previewing-applications-web-resources/) tutorial.



Sample application

Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/HelloWorldHybridProject.zip) the Studio project.
The sample application at this time is intended only for you to review the structure of an application; you are not required at all to use it.

Tutorials to follow next

Now that you created your first hybrid application, you can follow the tutorials in the Hybrid Development (../../hybrid-tutorials/) section to learn more about authentication and security, server-side development, advanced client-side development, notifications, and more.