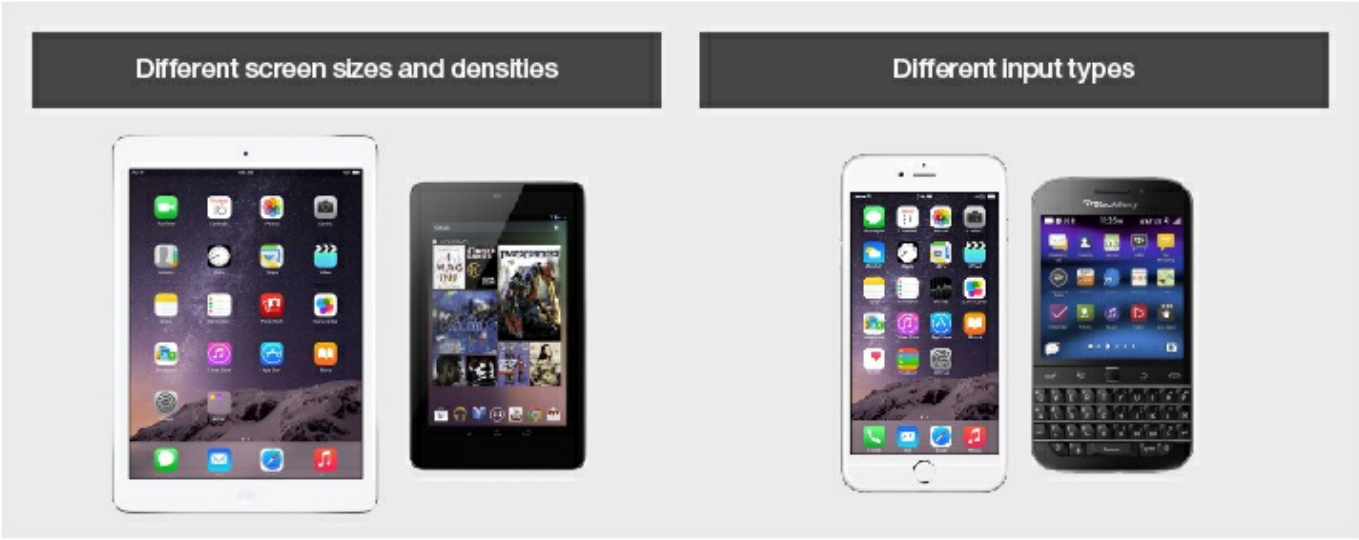
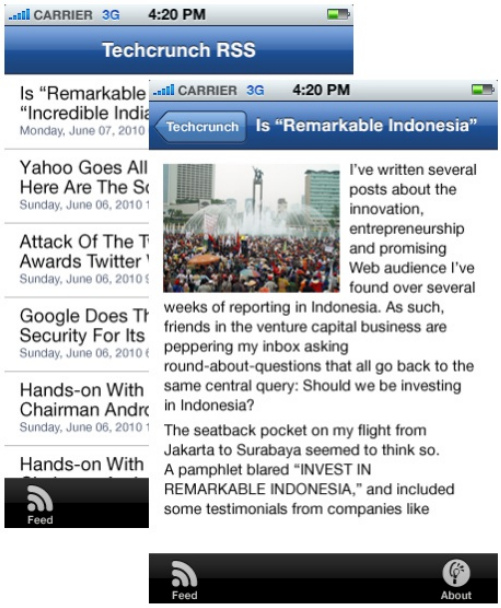
Supporting multiple form-factors using skins

fork and edit tutorial (https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/7.1/advanced-client-side-development/supporting-multiple-form-factors-using-skins.html) | report issue (https://github.ibm.com/MFPSamples/DevCenter/issues/new)

Overview

Skins provide support for multiple form factors in a single executable file for devices of the same OS family. They are a sub-variant of an environment and are packaged together in one application. The decision on which skin to use is made automatically at run time.





This tutorial covers the following topics:

- Skin creation
- Skin packaging
- Applying skins at run time

- Skin registration

Skin creation

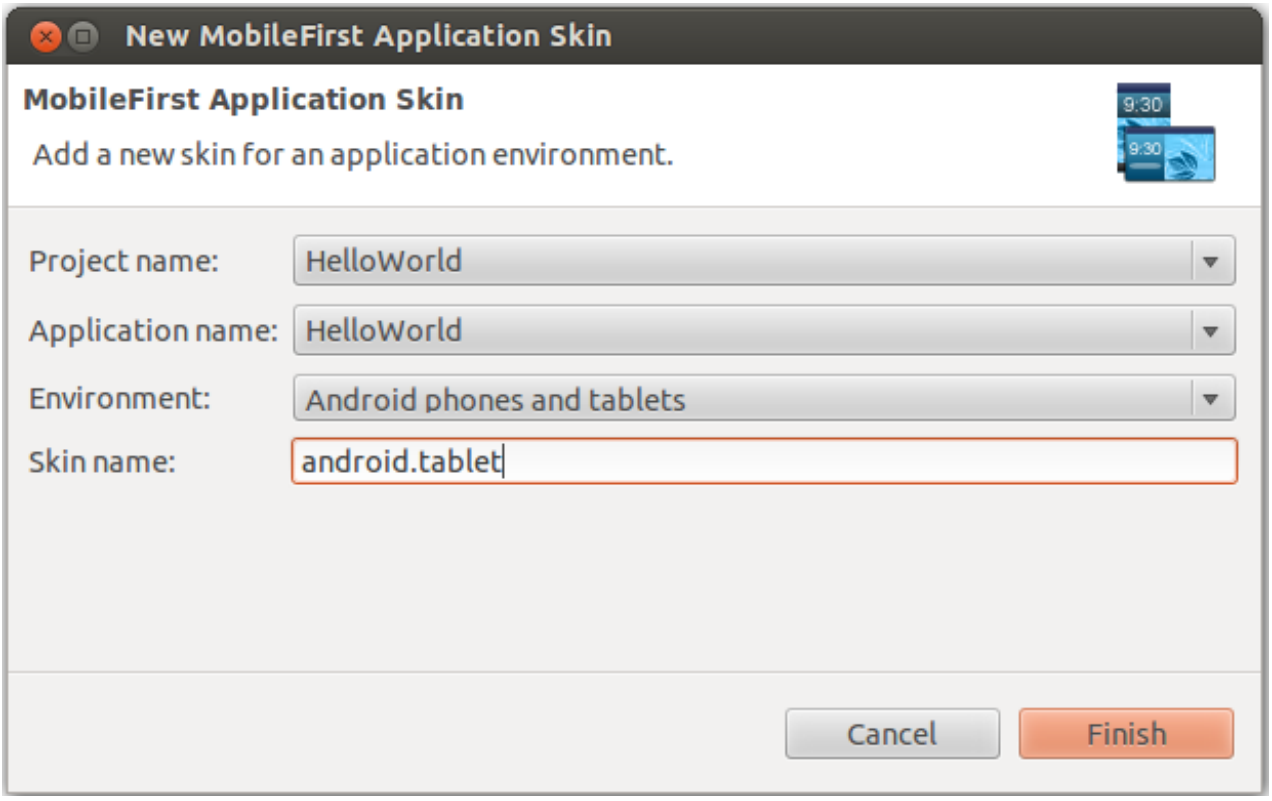
Skins can be created by either using MobileFirst Studio or CLI.
Before you can add a skin, make sure the environment for which you want to create the skin exists for the application. The skins are placed in directories that are next to the corresponding environment directory.

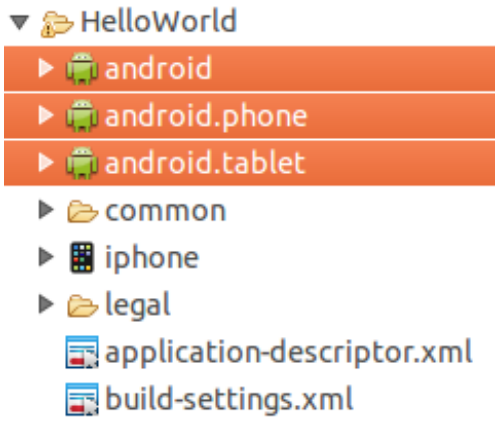
CLI

In a terminal window, navigate to the application folder and use the command: `mfp add skin`.
Follow the on interactive instructions to complete the process.

Studio

Use the Skin wizard in the Studio to add the environment for which you want to create the skin.



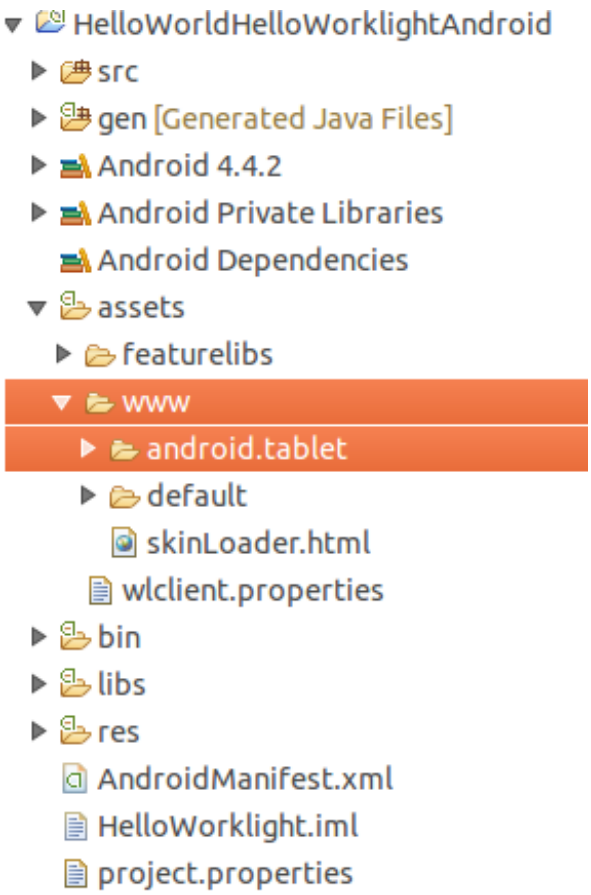


A skin folder contains the following folders: `images`, `CSS` and `js`. In order to alter the visuals and logic of the skin application, create new `.css` and `.js` files in the corresponding folders.

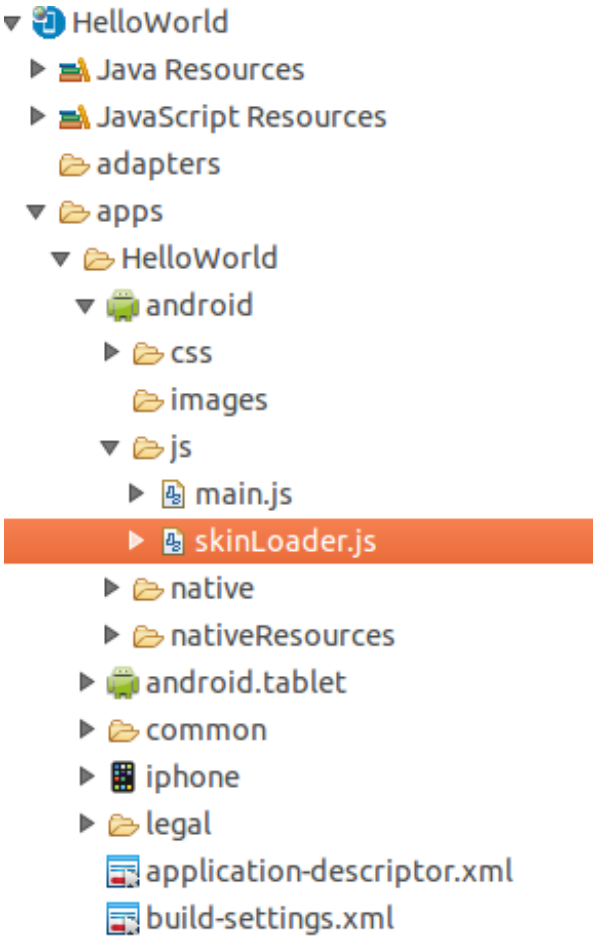
- To extend existing code and styling from the default skin, use the same file names as those of the default skin
- To create new code and styling, use different file names than those of the default skin. When you choose this option, you need to copy the HTML file to the skin folder. You must also change the reference to the `.js` and `.css` files in the `HEAD` element of the HTML file.

Skin packaging

All skins for a specific environment are packaged within the app.



Applying skins at run time



A special, developer-controlled JavaScript file, `skinLoader.js`, is run when the app starts in order to determine which skin to load.
The default `skinLoader.js` file contains examples of code.

```
function getSkinName() {
  var skinName = "default";
  if (device.version == "2.2" || device.version == "2.1")
{
    skinName = "android.HTML5";
  }
  return skinName;
}
```

Skin registration

Skins are automatically registered in the `application-descriptor.xml` file.
Registration determines the hierarchical order between the `common` folder, environment, and skin.
If you remove a skin from the project, you must modify the `application-decsriptor.xml` file manually.

```
<android version="1.0">
   <skins>
      <skin name="default">
         <folder name="common"/>
         <folder name="android"/>
      </skin>
      <skin name="android.tablet">
         <folder name="common"/>
         <folder name="android"/>
         <folder name="android.tablet"/>
      </skin>
      <skin name="android.phone">
         <folder name="common"/>
         <folder name="android"/>
         <folder name="android.phone"/>
      </skin>
   </skins>
   <worklightSettings include="false"/>
   <security>
      <encryptWebResources enabled="false"/>
      <testWebResourcesChecksum enabled="false" ignoreFileExtensions="png, jpg, jpeg, gif, mp4, m
p3"/>
      <publicSigningKey>Replace this text with the actual public signing key of the certificate used to si
gn the APK, available by using the 'Extract public signing key' wizard.</publicSigningKey>
      <packageName>Replace this text with the actual package name of the application, which is the val
ue of the 'package' attribute in the 'manifest' element in the AndroidManifest.xml file.</packageName>
   </security>
</android>
```