Debugging applications

Overview

In this tutorial, various approaches to debugging (the web resources of) a MobileFirst application will be explored - before running the application in a device and and while running in a device. Debugging of MobileFirst adapters will be explored as well, and tools available to the developer's disposal in order to conduct the debugging effort.

The available debugging options are:

- Debugging on a desktop browser
- Debugging with Mobile Browser Simulator
- Debugging with iOS Remote Web Inspector
- Debugging with Chrome Remote Web Inspector
- Debugging with Weinre
- Debugging with IBM MobileFirst Logger
- Testing the adapter procedures
- Debugging with WireShark

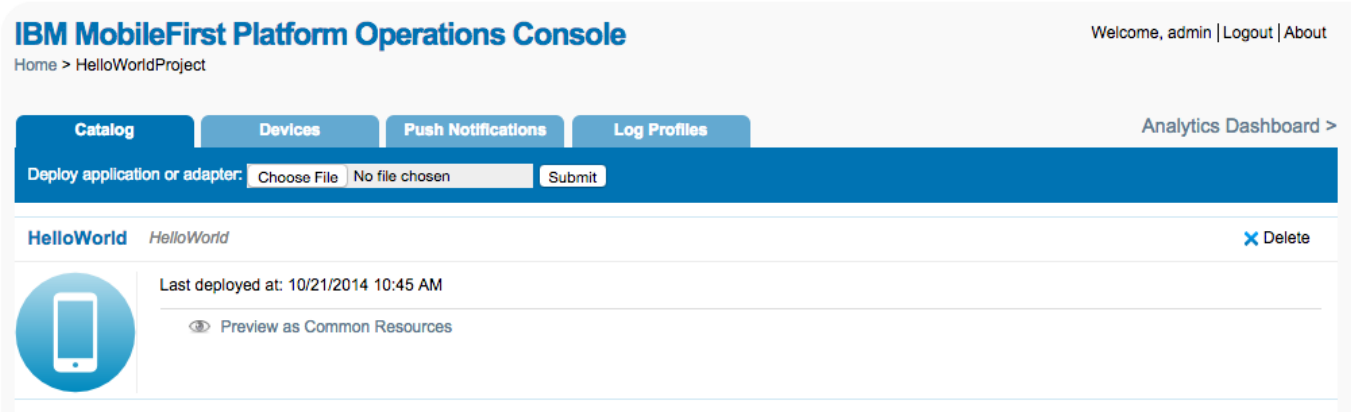
Debugging

Debugging is a process that consists of finding the cause of defects in the application code and UI.

- MobileFirst applications consist of web-based resources and optional native code (such as Java, Objective-C and C#).
- Native code can be debugged by using standard tools that are provided by the platform SDK, such as XCode, Android LogCat/ADB or Microsoft Visual Studio.

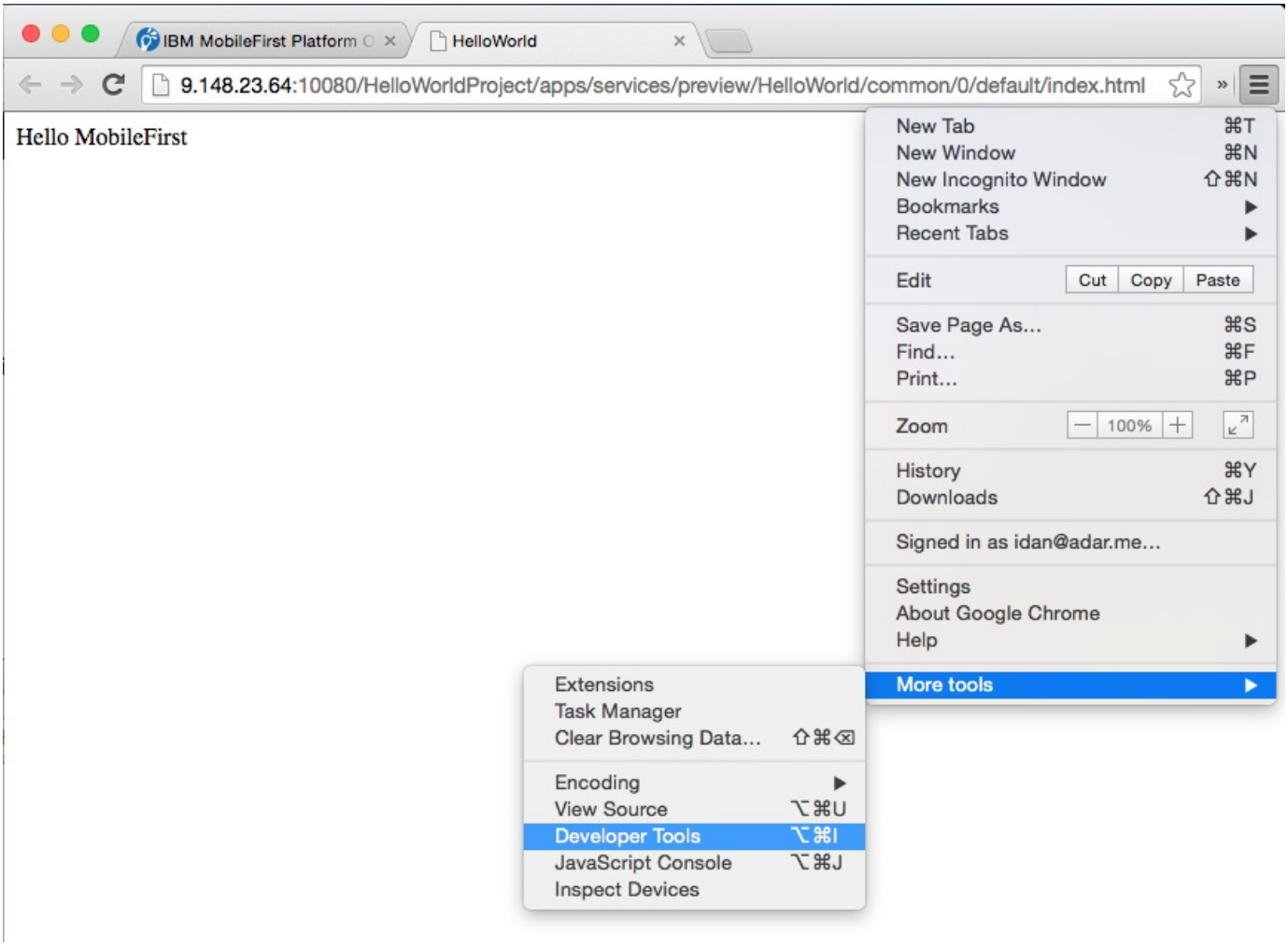
Debugging on a desktop browser

Modern browsers, such as Chrome, Firefox, Safari or Opera, provide an easy and convenient way to debug web apps.
As seen in the previous tutorials, during development you can preview applications in a desktop browser by using the MobileFirst Console.

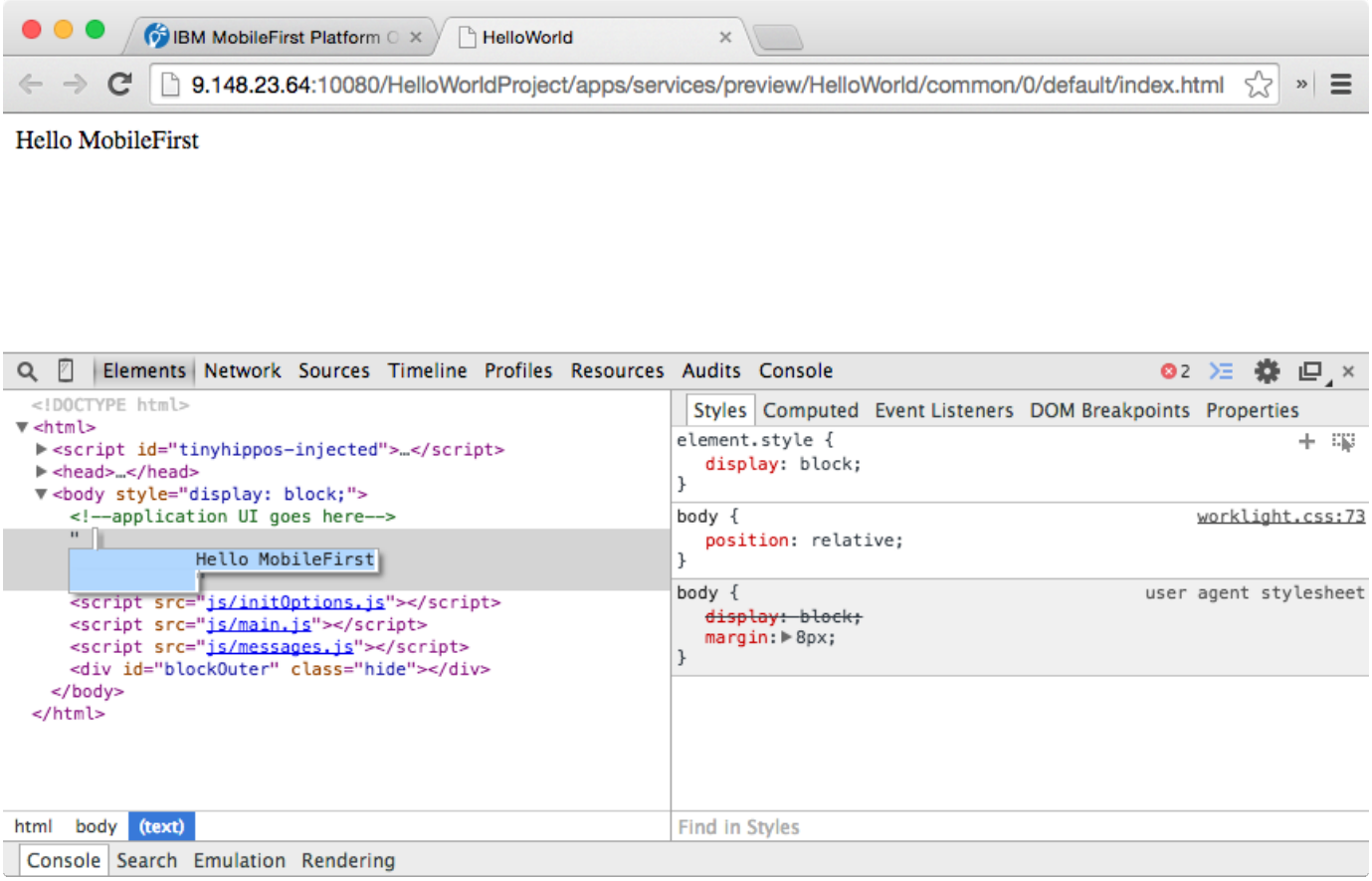


There are many web tools for debugging on various desktop browsers, for example:

- FireBug
- Chrome Developer Tools
- Internet Explorer Developer Tools
- Dragonfly for Opera
- Safari Web Inspector

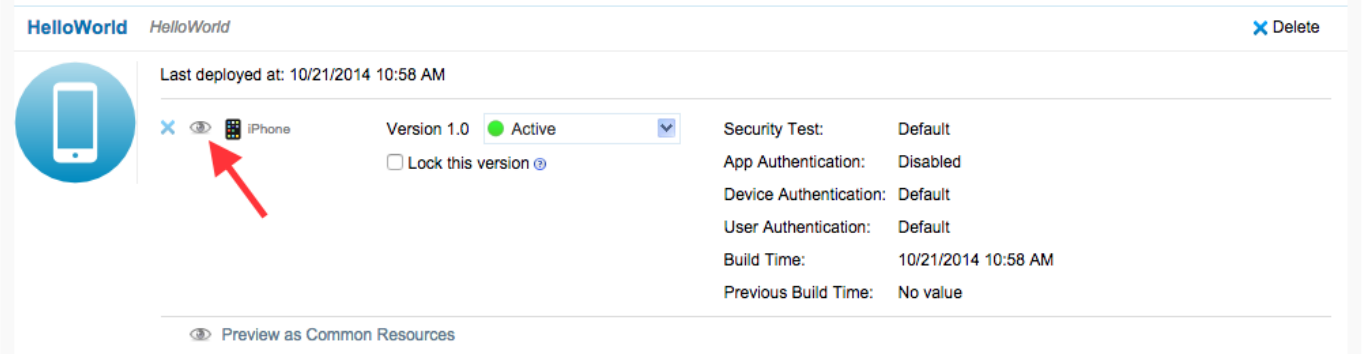


In early application development stages these tools can be used to debug the application just like a regular website. It is not required to install them in a mobile device.
Changes to HTML and CSS can also be previewed in real time by modifying the values in the inspector.



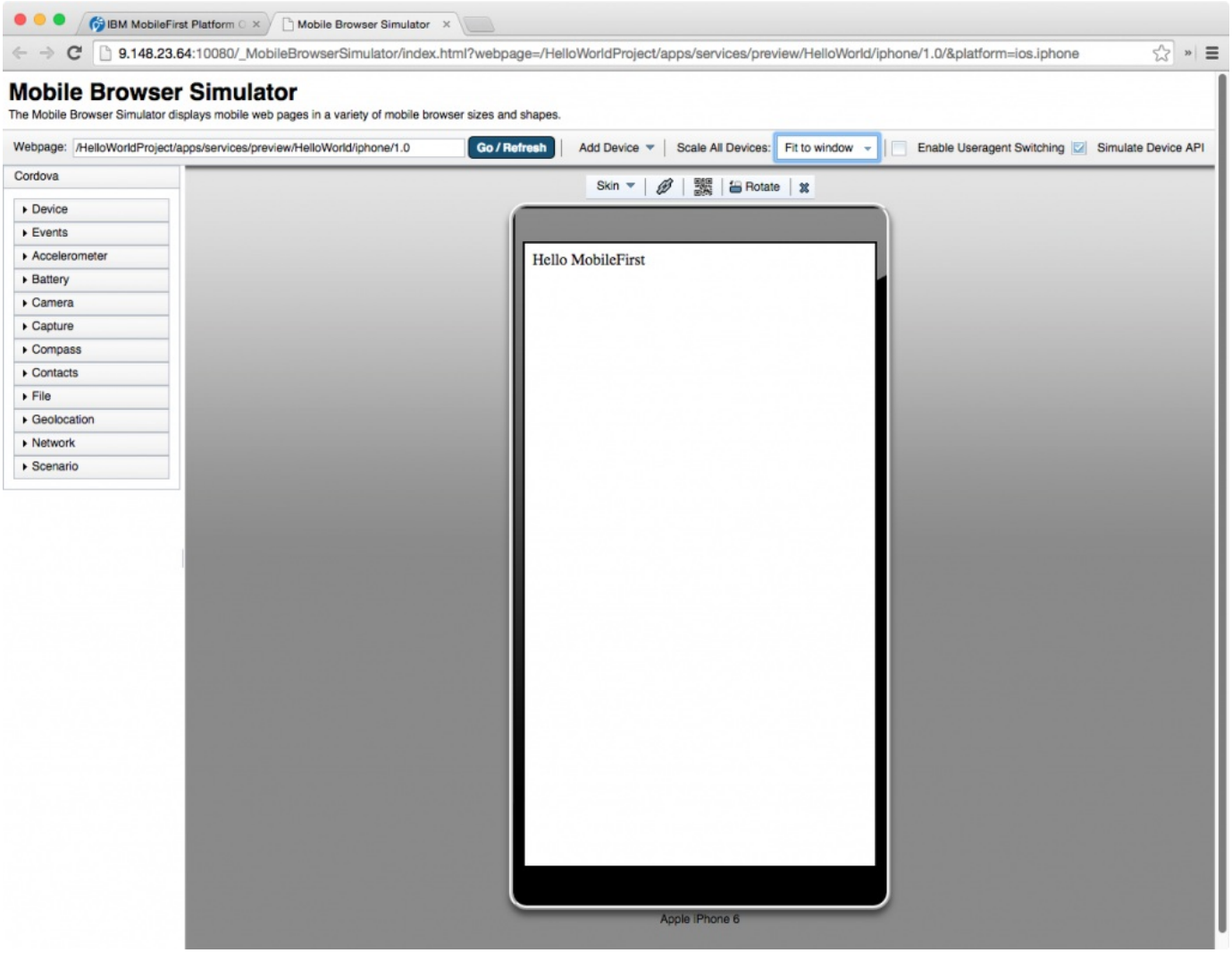
Debugging with the Mobile Browser Simulator

The Mobile Browser Simulator can also be used to preview and debug MobileFirst applications.
To access it, click on the 'eye' icon beside an environment row in MobileFirst Console.



The Mobile Browser Simulator has several added values over Preview as Common Resources, for example:

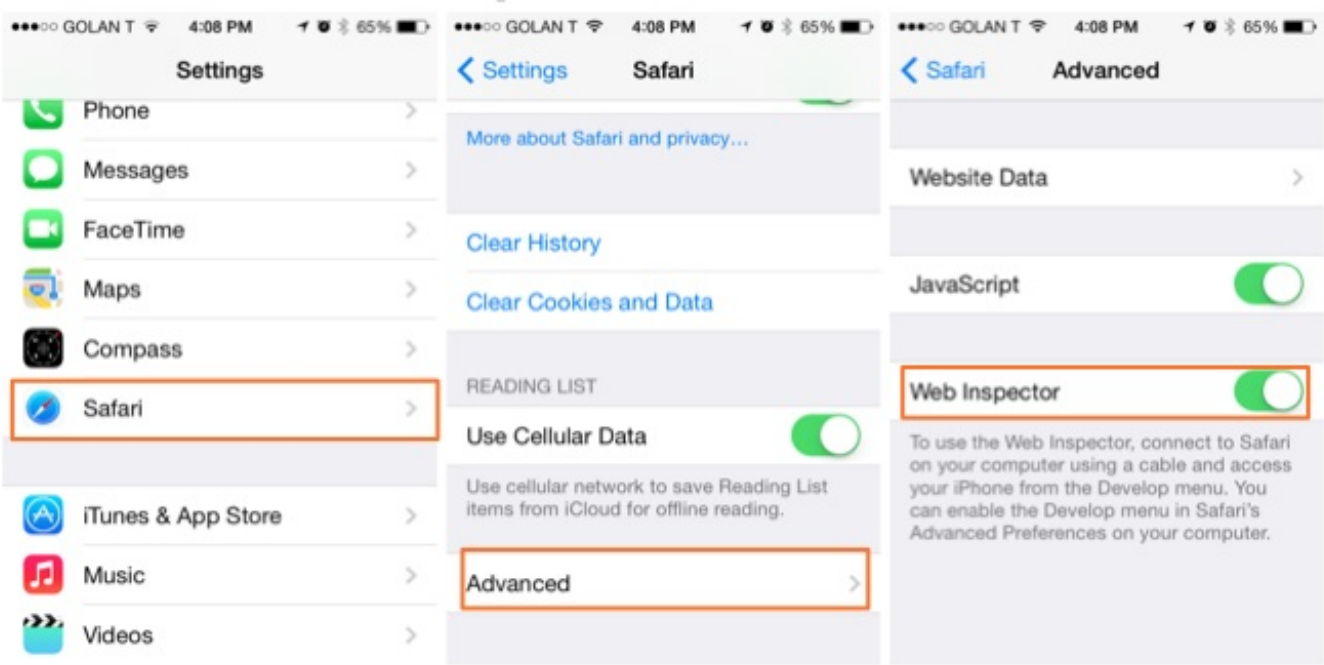
- Preview environment-specific resources
- Emulate different devices and skins
- Emulate some Cordova features such as access to sensors and other hardware



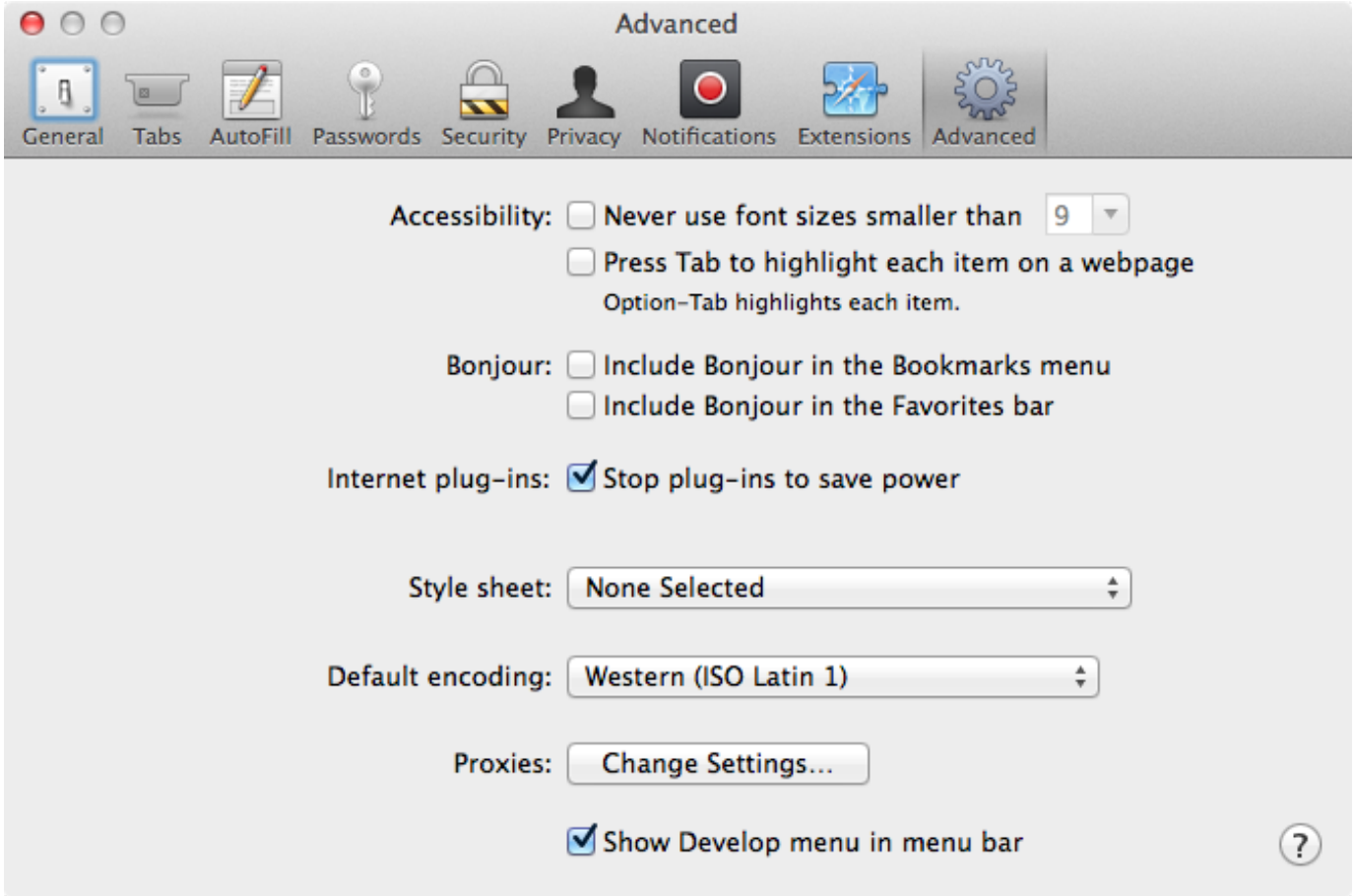
Debugging with iOS Remote Web Inspector

Starting in iOS 6, Apple introduced a remote web inspector for debugging web applications on iOS devices. In order to debug, make sure the device (or simulator) has “Private Browsing” turned off.

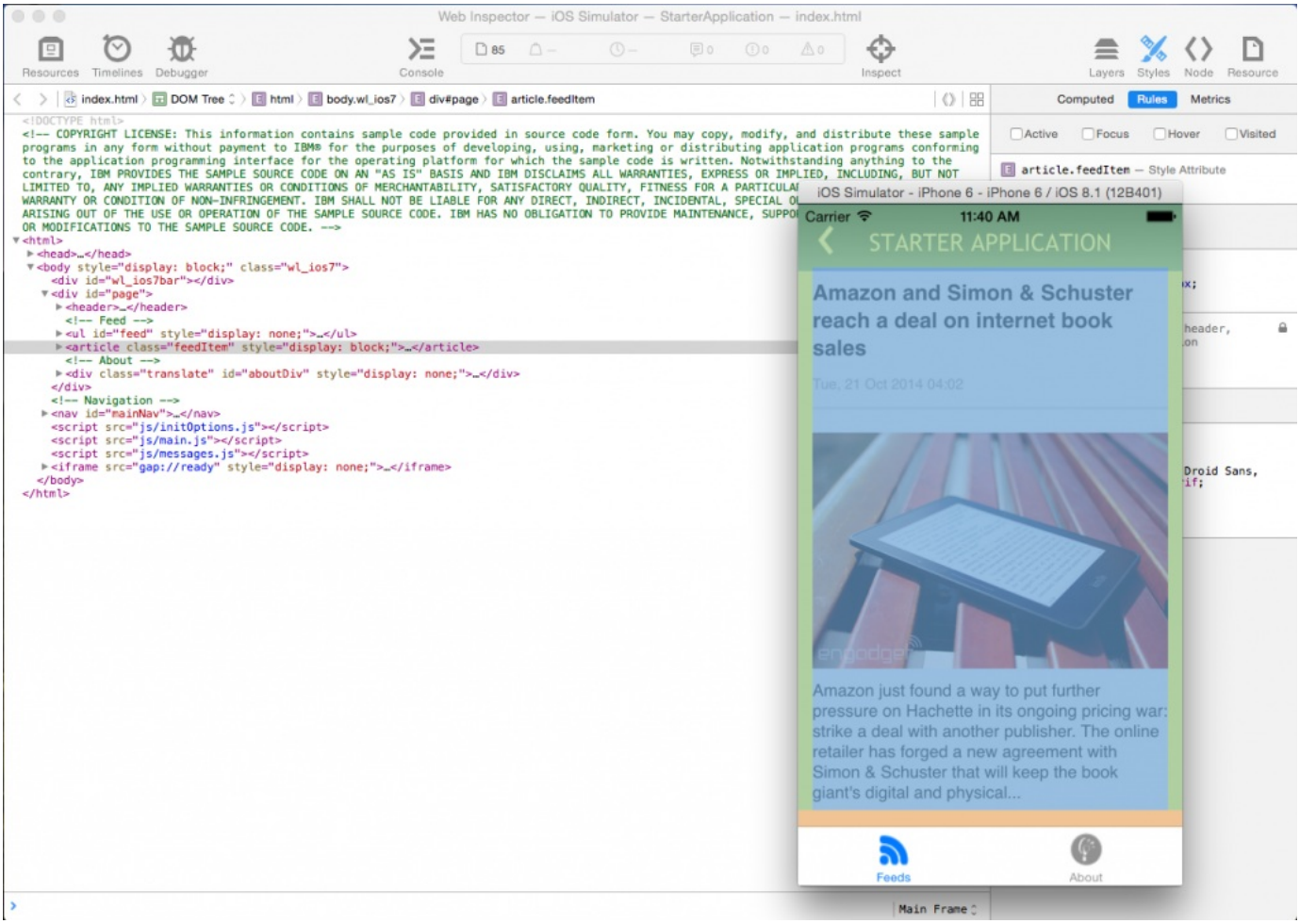
To enable Web Inspector on the device: **Settings > Safari > Advanced > Web Inspector**.



To start debugging, connect the iOS device to a Mac, or start the simulator. Safari 6.0 or higher is required. In Safari, go to **Preferences > Advanced**, and select the **Show Develop menu in menu bar** checkbox.



Now in Safari, select **Develop > [your device ID] > [your application HTML file]**.
The DOM can now be inspected. It is also possible to alter the CSS and run JavaScript commands, just like in the desktop inspector.

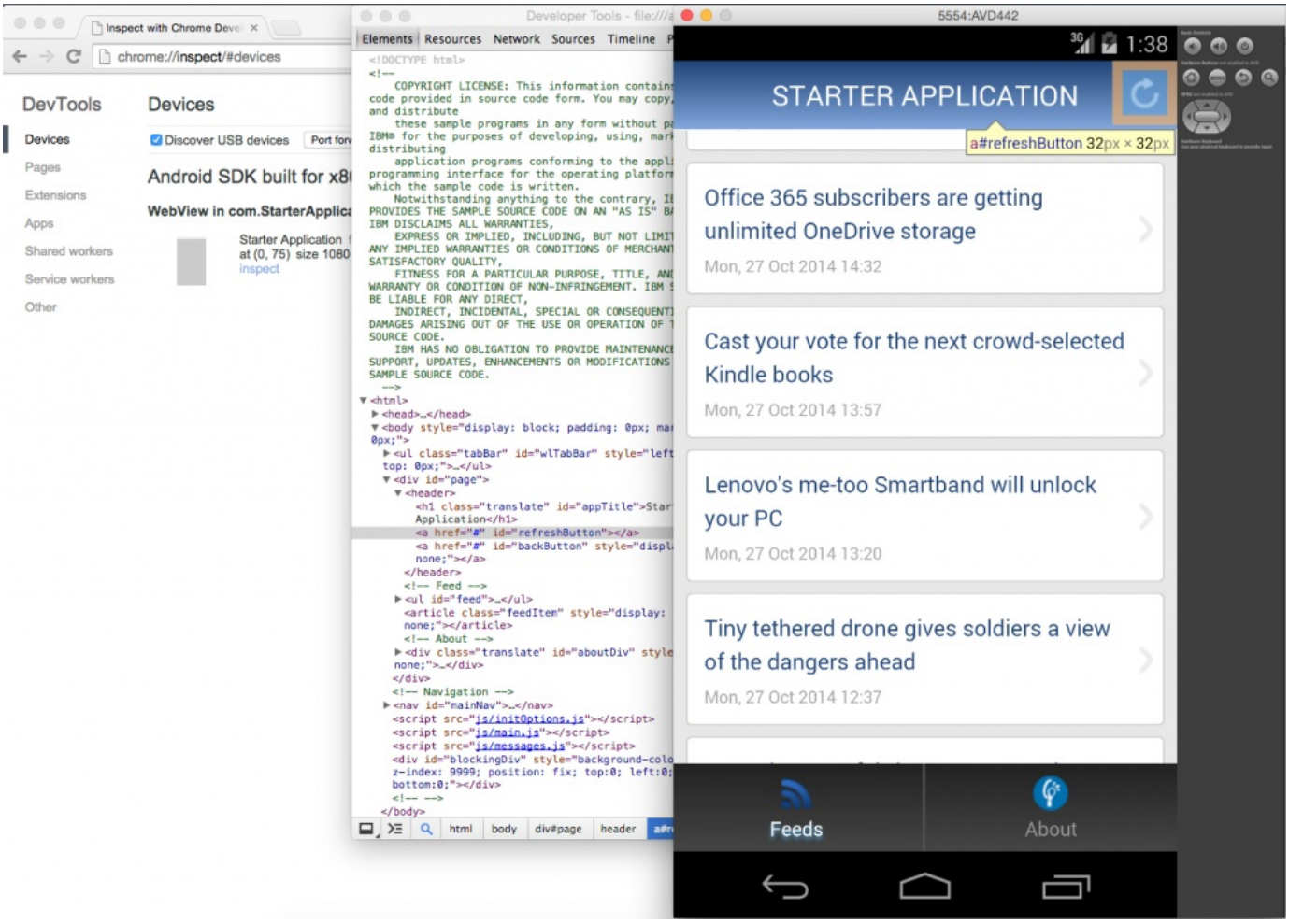


Debugging with Chrome Remote Web Inspector

With Google Chrome it is possible to remotely inspect web applications on Android devices.
This action requires Android 4.4 or later, Chrome 32 or later and IBM MobileFirst Platform Foundation V6.2.0 or later.

Additionally, in the **AndroidManifest.xml** file, **targetSdkVersion** = 19 or above is required.
In **project.properties**, **target** = 19 or above is required.

Start the application in the Android Emulator or a connected device. Then, in Chrome, enter the following URL: **about:inspect** and then press on "Inspect" for the relevant application. All the features of the Chrome Inspector can now be used to inspect the Android application.

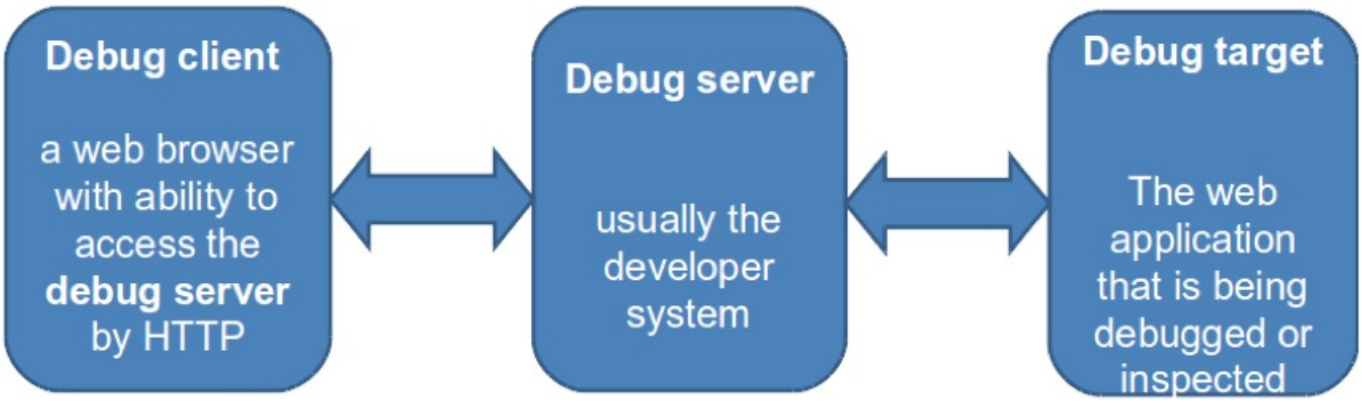


Debugging with Weinre

Weinre (http://people.apache.org/~pmuellr/weinre/) stands for **We**b **In**spector **Re**mote.

Weinre is a debugger for web pages, like Firebug or other Web Inspectors, except that Weinre is designed to work remotely.
Weinre can be used to inspect and debug web resources such as HTML, JavaScript, CSS, and network traffic on mobile handsets.

The Weinre architecture includes the following components:



The Weinre debug server requires a node.js runtime.
Instructions to install Weinre can be found at:
http://people.apache.org/~pmuellr/weinre/docs/latest/Installing.html
(http://people.apache.org/~pmuellr/weinre/docs/latest/Installing.html)

Debug server
When the Weinre server is installed, the following command will run it:
`weinre --httpPort 8888 --boundHost -all-`

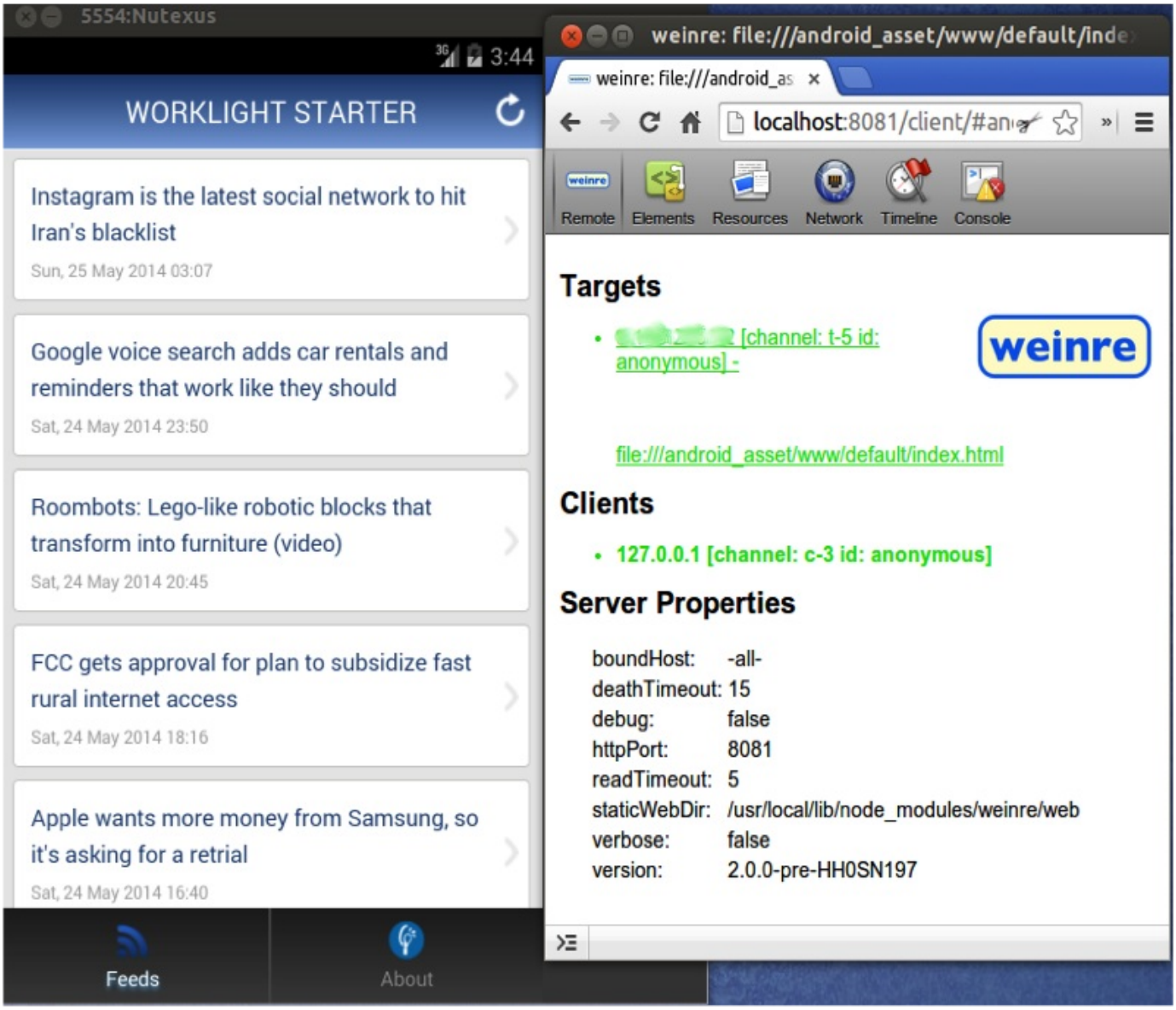
This command starts a Weinre server on a default (changeable) port 8888.

Target
The Weinre server must be accessible from the device that will be used for debugging. To make it accessible, add the following code line to the web application:

```
<script src="http://a.b.c:8888/target/target-script-min.js"></script>
```

Where a.b.c is the hostname or IP of the Weinre server.

Client
Before you can start debugging, make sure that the application is open and loaded on the browser with this URL:



Debugging with IBM MobileFirst Logger

IBM MobileFirst Platform Foundation provides a `WL.Logger` object that can be used to print log messages to the log for the environment used.

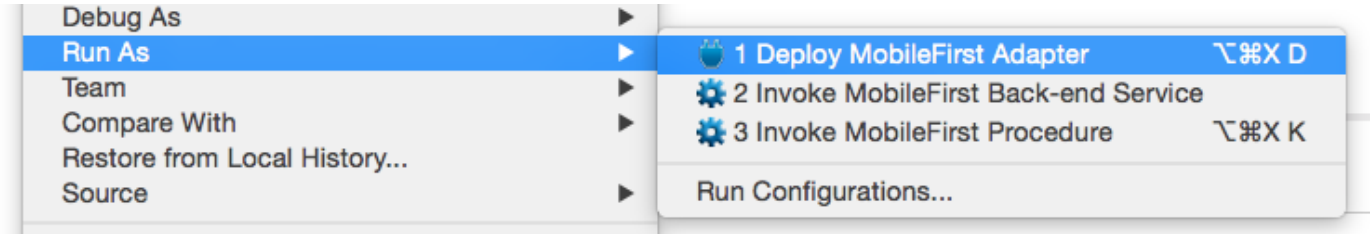
Two of its methods are `WL.Logger.debug()` and `WL.Logger.error()`.
These APIs are multi-platform. The output destination changes according to the platform on which that application runs on:

- **Developer console** when it is running on a desktop browser
- **LogCat** when it is running on Android device
- **Visual Studio Output** when it is running on a Windows Phone 8 device and Windows 8 App
- **XCode Console** when it is running on an iOS device

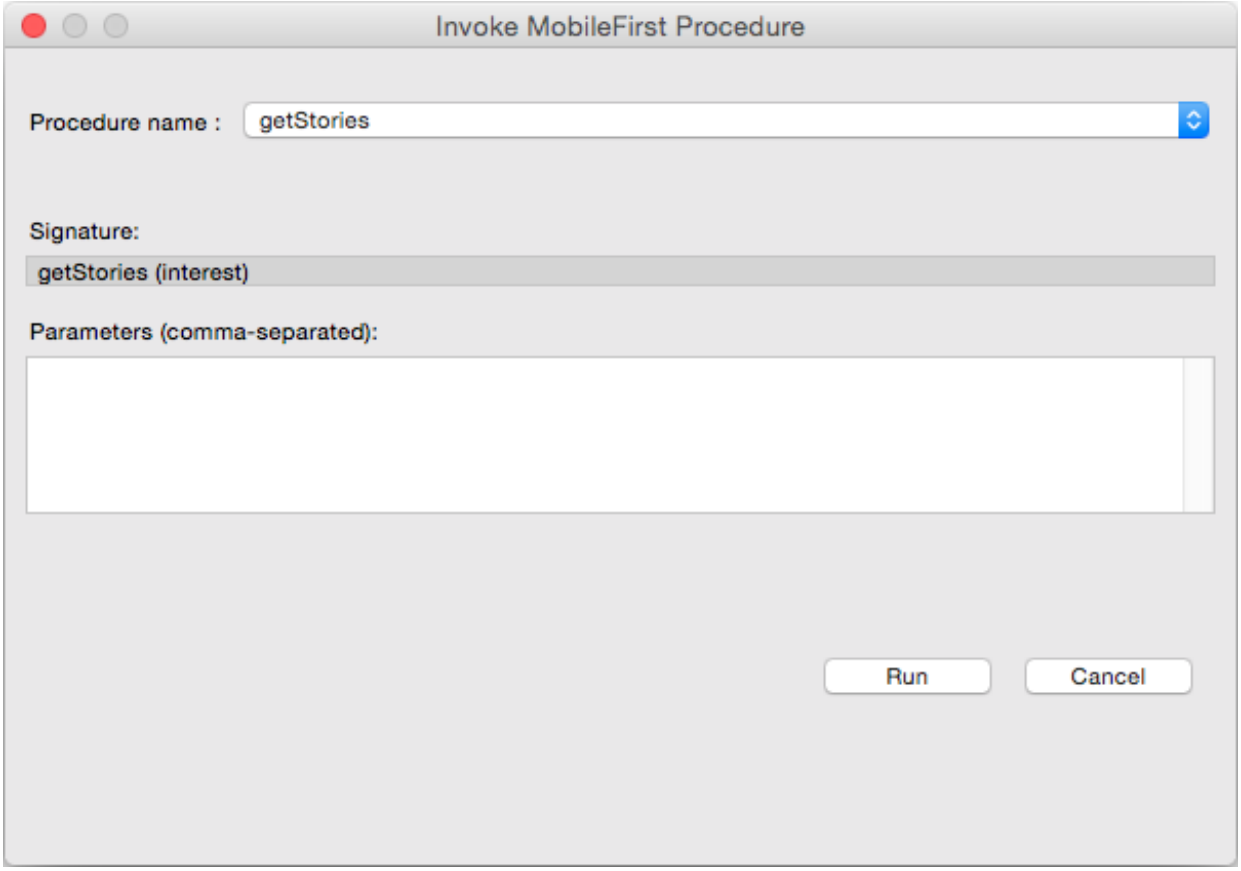
`WL.Logger` contains more methods.
More information is available in the IBM MobileFirst Platform Foundation user documentation topic for `WL.Logger`.

Testing the adapter procedures

It is possible to test adapter procedures by using MobileFirst Studio.
Testing a procedure is done by right-clicking an adapter folder and selecting **Run As > Invoke MobileFirst Procedure**.



After selecting to invoke a procedure, the adapter and procedure are selected, followed by optionally entering comma-separated parameters.



Adapter invocation result:

Invocation Result of procedure: 'getStories' from the MobileFirst Server:


```
{
  "errors": [
  ],
  "info": [
  ],
  "isSuccessful": true,
  "responseHeaders": {
    "Alternate-Protocol": "80:quic,p=0.01,80:quic,p=0.01",
    "Cache-Control": "private, max-age=0",
    "Content-Type": "text/xml; charset=UTF-8",
    "Date": "Tue, 28 Oct 2014 12:44:22 GMT",
    "ETag": "X8aekjl3CvT45xpcepn6EK2pDJw",
    "Expires": "Tue, 28 Oct 2014 12:44:22 GMT",
    "Last-Modified": "Tue, 28 Oct 2014 12:44:19 GMT",
    "Server": "GSE",
    "Transfer-Encoding": "chunked",
    "X-Content-Type-Options": "nosniff",
    "X-XSS-Protection": "1; mode=block"
  },
  "responseTime": 299,
  "rss": {
    "channel": {
      "copyright": "Copyright 2014 Cable News Network LP, LLLP.",
      "description": "CNN.com delivers up-to-the-minute news and information on the latest top stories, weather, entertainment, politics and more.",
      "image": {
        "description": "CNN.com delivers up-to-the-minute news and information on the latest top stories, weather, entertainment, politics and more.",
        "height": "33",
        "link": "http://edition.cnn.com/index.html?eref=edition",
        "title": "CNN.com - Top Stories",
        "url": "http://i.cdn.turner.com/cnn/.e/img/1.0/logo/cnn.logo.rss.gif",
        "width": "144"
      },
      "info": {
        "feedburner": "http://rssnamespace.org/feedburner/ext/1.0",
        "uri": "rss/edition"
      },
      "item": [
        {
          "description": "The South African state is to appeal both judgment and sentence after athlete Oscar Pistorius was jailed for five years for shooting his girlfriend.",
          "guid": "http://edition.cnn.com/2014/10/27/justice/south-africa-oscar-pistorius-appeal/index.html",
```


Debugging with WireShark

Wireshark is a network protocol analyzer that can be used to see what happens in the network. Filtering is available to follow only what is required.

For more information, see http://www.wireshark.org/ (http://www.wireshark.org/)

Capturing from 2 interfaces [Wireshark 1.8.1 (SVN Rev 43946 from /trunk-1.8)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: http && tcp.port == 10080 Expression... Clear Apply Save Filter Filter Filter Filter

No.	Time	Source	Destination	Protocol	Length	Info
11007	300.268344	9.148.225.225	9.148.23.122	HTTP	518	HTTP/1.1 403 Forbidden (application/json
11228	349.498153	9.148.23.122	9.148.225.225	HTTP	482	GET /zzz/apps/services/api/zzz/android/an
11230	349.507158	9.148.225.225	9.148.23.122	HTTP	71	HTTP/1.1 403 Forbidden (text/html)
11282	358.387361	9.148.23.122	9.148.225.225	HTTP	385	POST /PushNotificationsProject/apps/servi
11288	358.405062	9.148.225.225	9.148.23.122	HTTP	523	HTTP/1.1 200 OK
11806	459.279102	9.148.23.122	9.148.225.225	HTTP	447	GET /PushNotificationsProject/apps/servic
11807	459.281675	9.148.225.225	9.148.23.122	HTTP	497	HTTP/1.1 200 OK (text/html)
11810	459.321210	9.148.23.122	9.148.225.225	HTTP	188	POST /PushNotificationsProject/apps/servi
11814	459.331828	9.148.225.225	9.148.23.122	HTTP	3419	HTTP/1.1 200 OK
12019	467.480230	9.148.23.122	9.148.225.225	HTTP	94	POST /PushNotificationsProject/apps/servi
12021	467.543998	9.148.225.225	9.148.23.122	HTTP	417	HTTP/1.1 200 OK (application/json)
12026	467.625773	9.148.23.122	9.148.225.225	HTTP	188	POST /PushNotificationsProject/apps/servi
12028	467.628632	9.148.225.225	9.148.23.122	HTTP	621	HTTP/1.1 401 Unauthorized (application/j
12031	467.686034	9.148.23.122	9.148.225.225	HTTP	190	POST /PushNotificationsProject/apps/servi
12034	467.715583	9.148.225.225	9.148.23.122	HTTP	2230	HTTP/1.1 200 OK (application/json)

```
⊞ Frame 10706: 385 bytes on wire (3080 bits), 385 bytes captured (3080 bits) on interface 1
⊞ Ethernet II, Src: Cisco_bf:16:3f (00:22:55:bf:16:3f), Dst: WistronI_ce:b2:ca (f0:de:f1:ce:b2:ca)
⊞ Internet Protocol Version 4, Src: 9.148.23.122 (9.148.23.122), Dst: 9.148.225.225 (9.148.225.225)
⊞ Transmission Control Protocol, Src Port: 56781 (56781), Dst Port: amanda (10080), Seq: 751, Ack: 1, Len: 319
⊞ [2 Reassembled TCP Segments (1069 bytes): #10705(750), #10706(319)]
⊞ Hypertext Transfer Protocol
⊞ Line-based text data: application/x-www-form-urlencoded
```