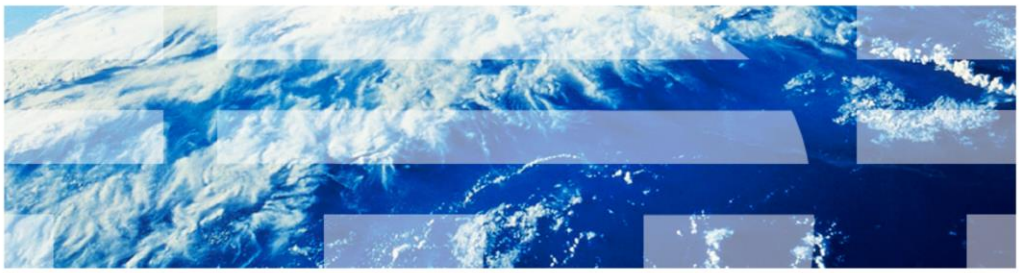
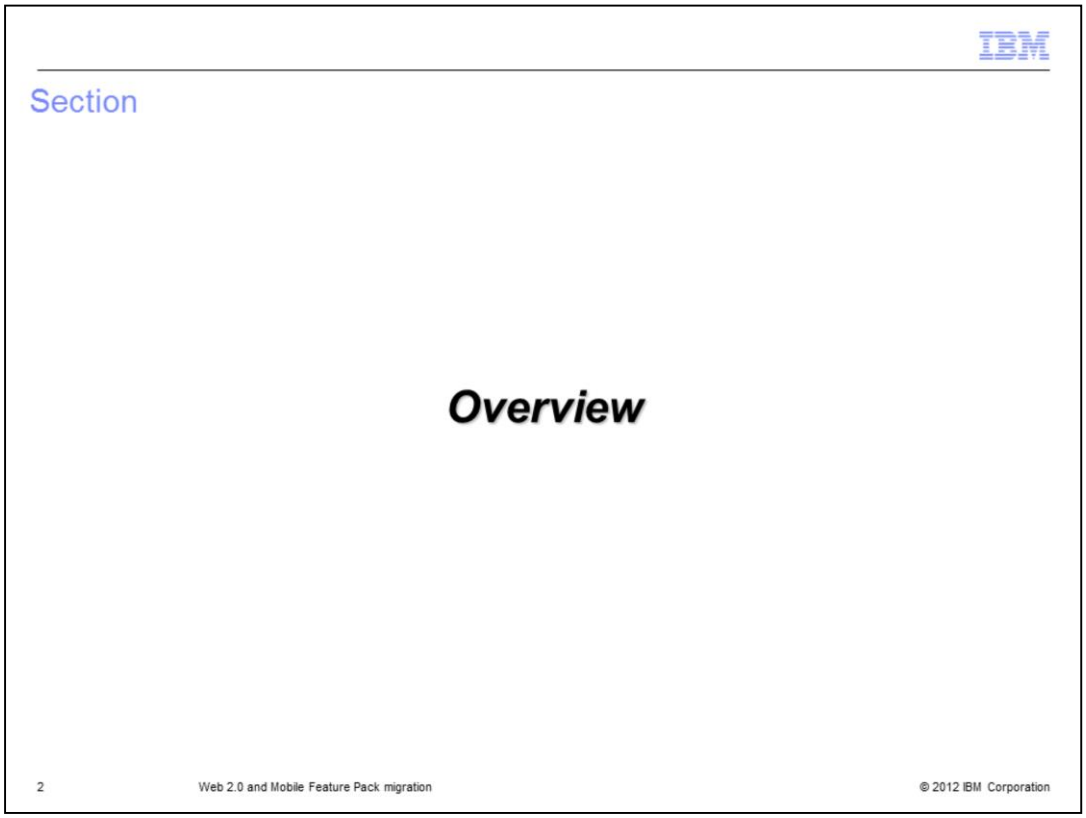
IBM Worklight 5.0

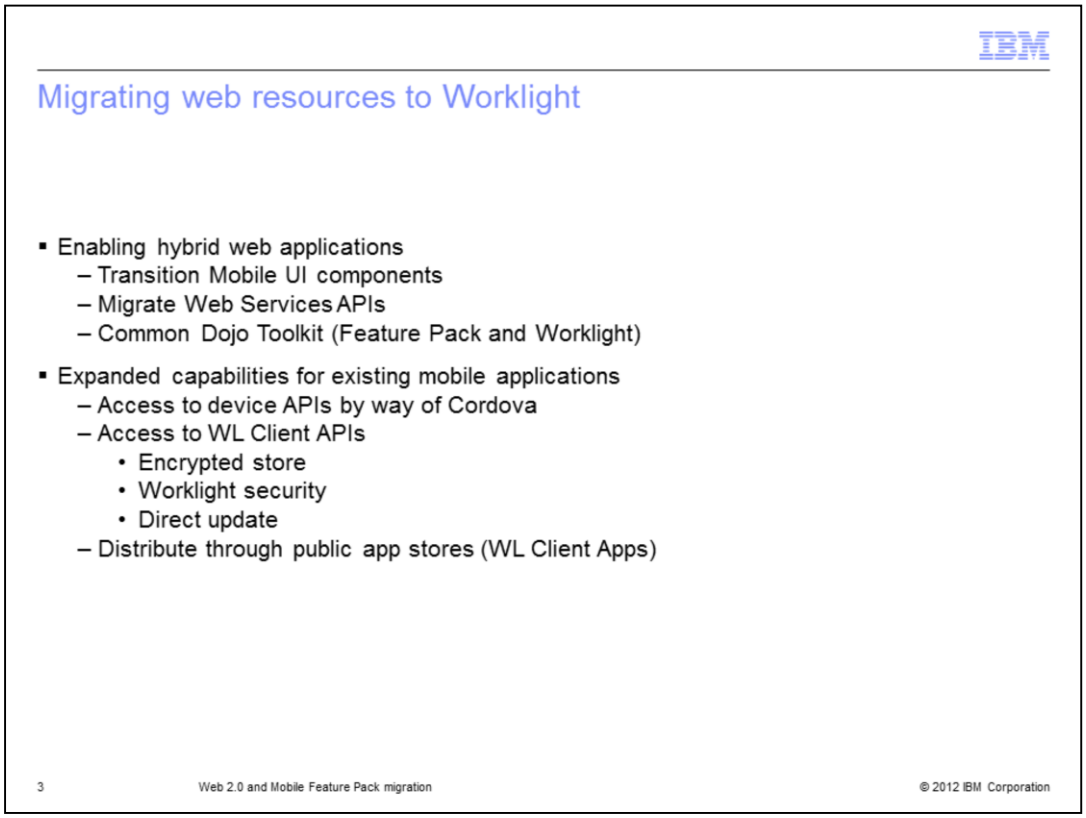
Web 2.0 and Mobile Feature Pack migration



This presentation describes the model for migrating from the web 2.0 and Mobile Feature Pack to IBM Worklight 5.0.



IBM Worklight 5.0 is the next step in the evolution of mobile web application development from IBM. In 2011, IBM WebSphere® Application Server released a feature pack that allowed you to extend your web presence to mobile devices. The web 2.0 and Mobile Feature Pack allowed you to create mobile web applications using open standards like HTML, CSS, and JavaScript to connect SOA based services using JAX-RS and JSON-RPC. For the client programming model, the feature pack accelerated time-to-value when building these mobile web applications using the Dojo Toolkit. One of the key libraries in Dojo is Dojo Mobile, an HTML5 mobile JavaScript framework that enables rapid development of mobile web applications with a native look and feel on modern webkit-enabled mobile devices including iPhone, iPod Touch, iPad, Android, and RIM smartphones and tablets. In support of IBM's mission of Service Oriented Architecture (SOA) enabling the enterprise, the Feature Pack for Mobile provided support for building JSON-RPC and REST Services using Apache Wink. This presentation focuses on both the client and the server programming models and how you can migrate existing mobile web applications to IBM Worklight.



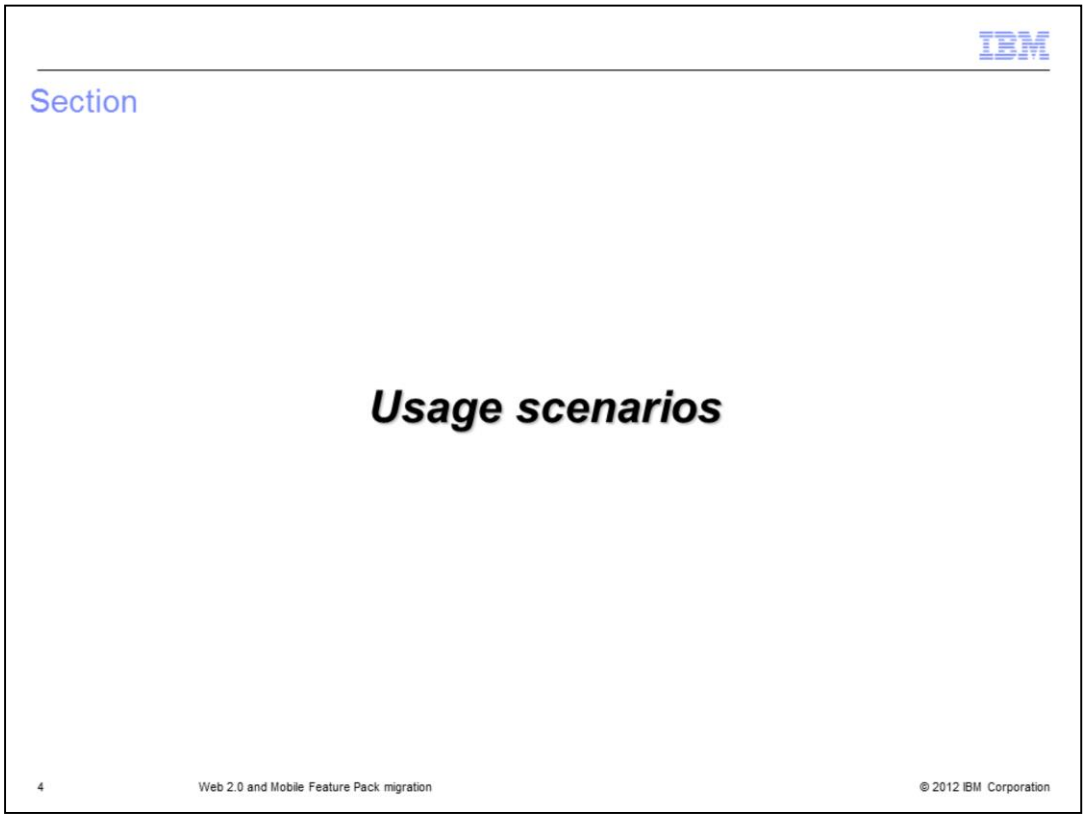
There are multiple ways to build mobile applications, ranging from building mobile web applications that run in a mobile browser to natively packaged applications that are installed and hosted on the device. In the Web 2.0 and Mobile Feature Pack, IBM provided libraries and components that accelerate the building of mobile web applications that can be accessed by several mobile devices and tablets.

Mobile application stores have surfaced for many reasons, including brand recognition and corporate marketing. Users often look for an application in the app store for a particular company before navigating to their site with a mobile browser.

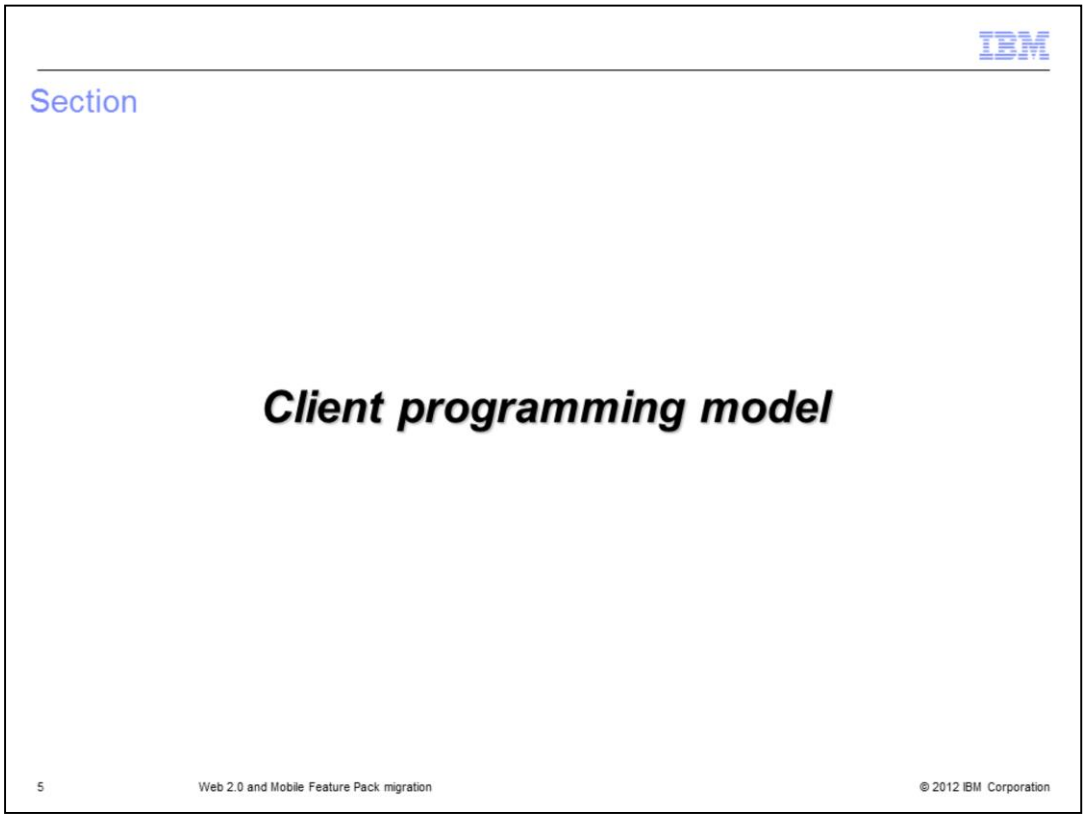
When the content is hosted locally on the device, the user navigation experience and application load time are improved.

With the release of Worklight 5.0, mobile web applications can now be packaged as native applications (referred to as hybrid applications) and deployed through an application store. You can transition these mobile components from mobile web to hybrid with limited packaging changes using technologies that are common across both web and hybrid applications - such as the Dojo Toolkit.

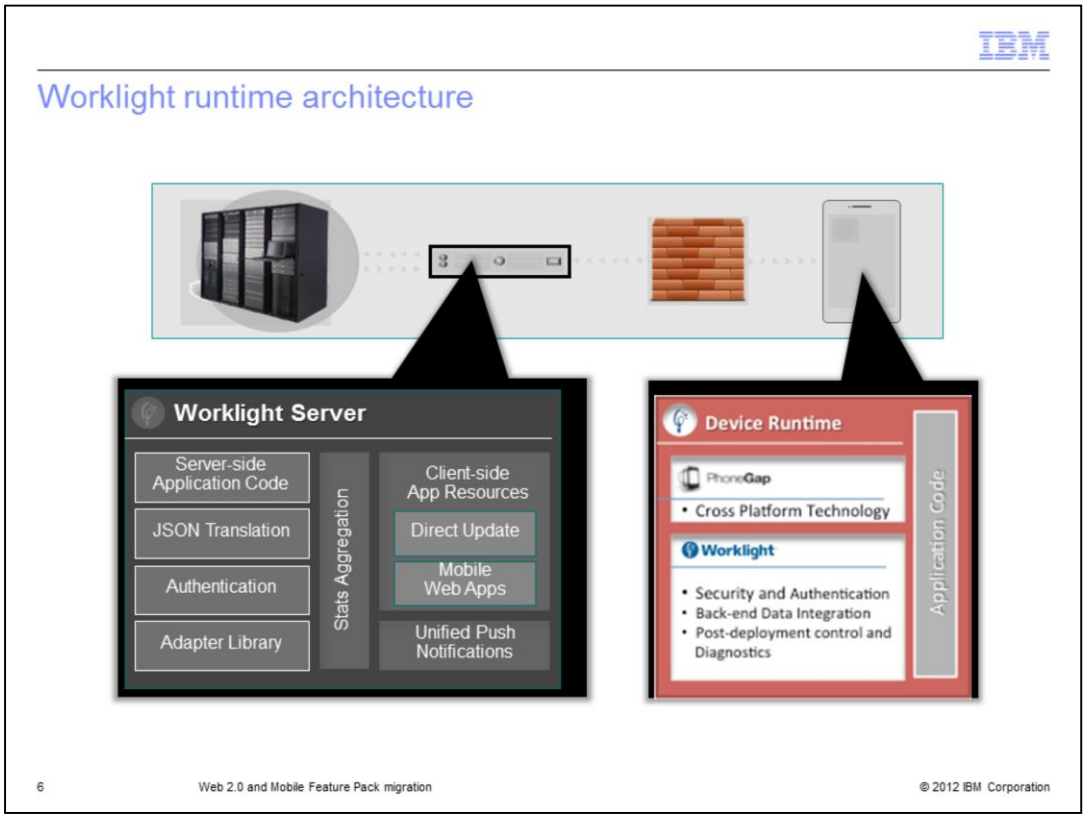
You can use device APIs and Worklight client APIs for features ranging from accessing the device's camera to having an encrypted data store for persisting user information securely. With these additional features of the Worklight Studio, time-to-market for these enhanced applications is unparalleled.



The following sections cover scenarios for migrating mobile web applications to Worklight. The first scenario is the client programming model, where you can take an existing mobile web application using Dojo and port the application to Worklight. The second scenario shows the server side programming model and how you can extend existing server applications to Worklight.



With the industry movement towards Web 2.0 interaction patterns where the client user interface templated content and the dynamically generated content is accessed with RESTful service, the client programming model has become the standard for most mobile web applications. This interaction pattern lets you choose the most appropriate user interface model (native or web) based on the requirements for the application and whether there needs to be a complimentary desktop version of the application. Because this presentation is focused on migrating existing mobile web applications to Worklight, it will focus on the hybrid model where you package the mobile web application as a mobile web hybrid application.



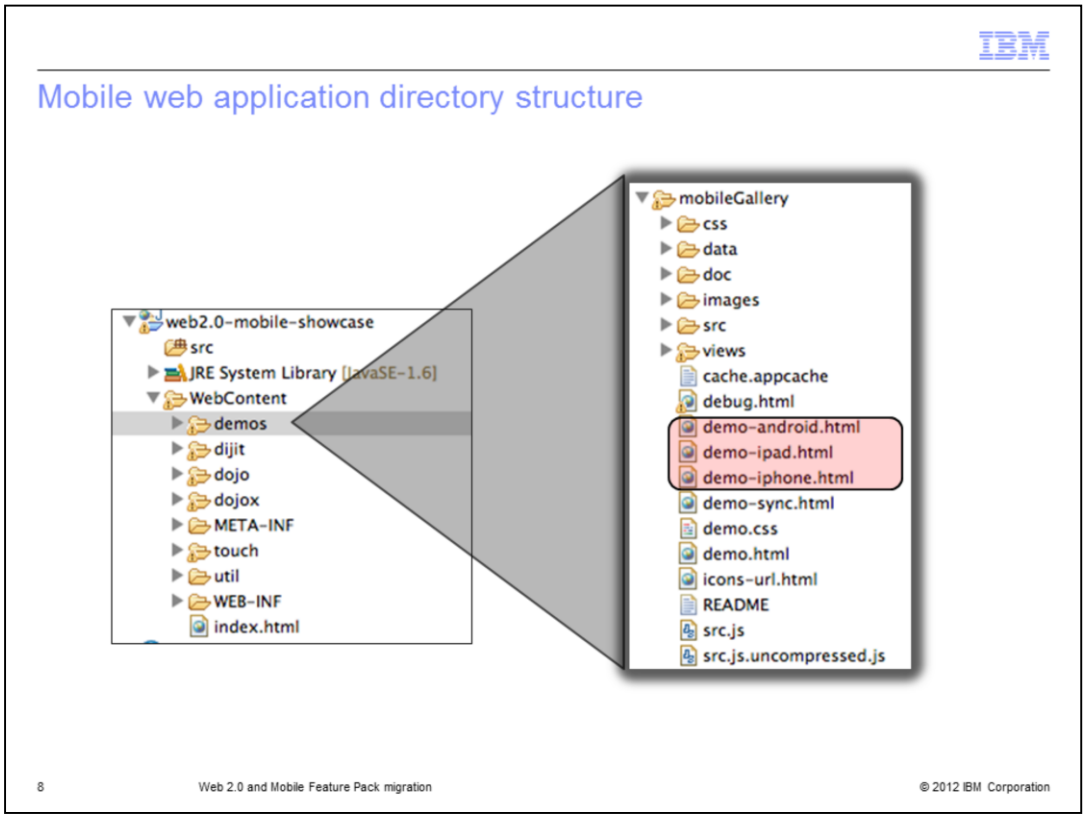
First, a high level overview of how a typical Worklight topology might be designed. This diagram shows a standard multiple tier topology consisting of the mobile device, the enterprise firewall protecting the Worklight Server (hosting the WL Security Model and Console) and then the enterprise services that interact with the Worklight server. One of the key features for Worklight 5.0 is the notion of direct update. For direct update, Worklight 5.0 assumes the basic application has HTML, Javascript, and CSS packaged WITH the application and can be updated (in static form) to the native shell. This model lets you update web resources without having to repackage the application and update the application store. This is a significant time saver and provides some semblance of compatibility with the traditional mobile web development model where updating the application means updating the web archive with the latest version of the application.

The Dojo Mobile Showcase Application is a sample application that was written when Dojo introduced its mobile widget library. In this showcase application, developers can easily navigate the various mobile widgets provided by Dojo with examples and source code available for each of them. The reason why this sample was chosen is that it was designed with reuse in mind using device detection APIs to determine the appropriate rendering based on whether the device was Android or iOS or tablet.

This slide shows the initial landing page of the application displayed on iOS and Android tablet devices. Notice the application uses the available screen area for split pane viewing of the various widgets. As each widget is selected in the right side list view, the left side pane loads the content.

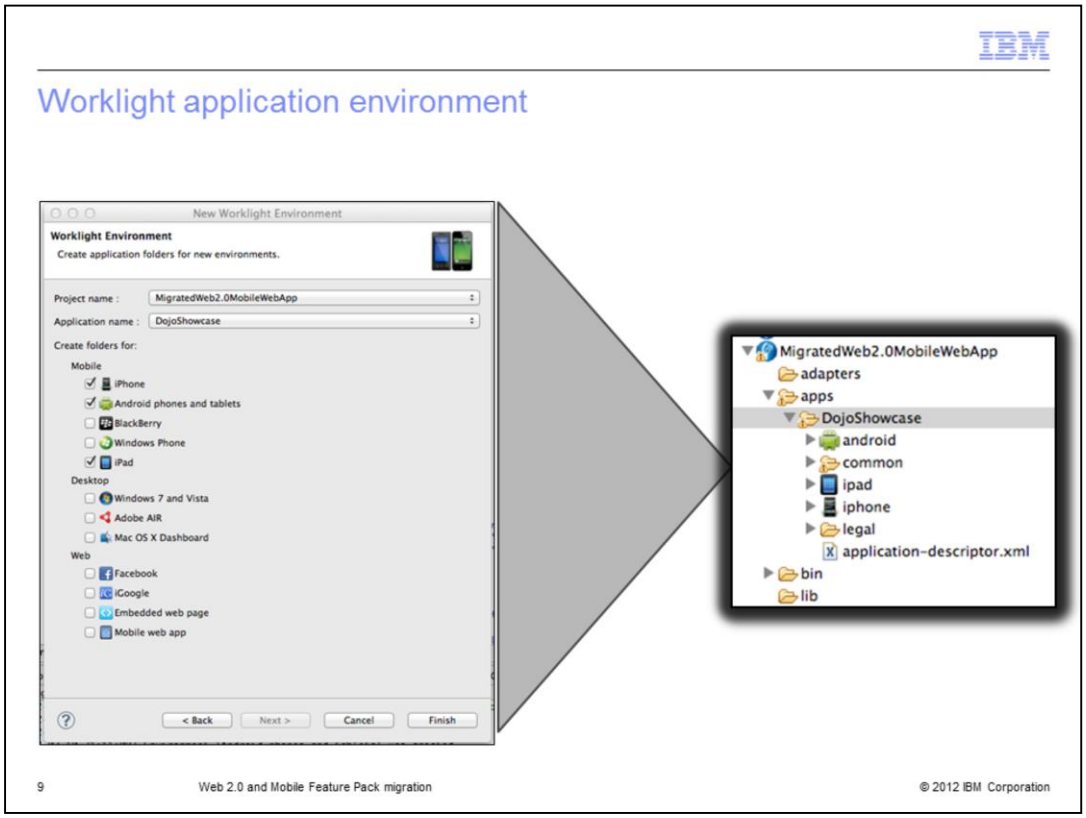
In addition to support for tablets, this same application can be rendered on a mobile phone. The available screen space is substantially smaller on a smartphone than on a tablet. As a result, the navigation also changes to transition the current list view to a detailed view of the selected widget.

In the next few slides you will see how to migrate this application to Worklight.



In JEE, there is a standard model for deploying web applications called the Web Archive also known as a .WAR file. In a .WAR file, there is a directory denoted for serving web content (the root directory) and a directory for configuration called WEB-INF. When developing in an IDE such as Rational® Application Developer, the root directory appears as a directory labeled WebContent.

In the Dojo Mobile Showcase, there is a subdirectory labeled demos that contains the demos for each of the mobile components you saw on the previous slide. Using device detection APIs such as dojo.has() and dojox/mobile/deviceTheme, the client device is redirected to the most appropriate resource for that device. In this example, there are three resources highlighted for the various supported device types. Dojo provides custom themes to render most appropriately on each device. In the case for Android, a single theme is used for both the tablet and the phone.



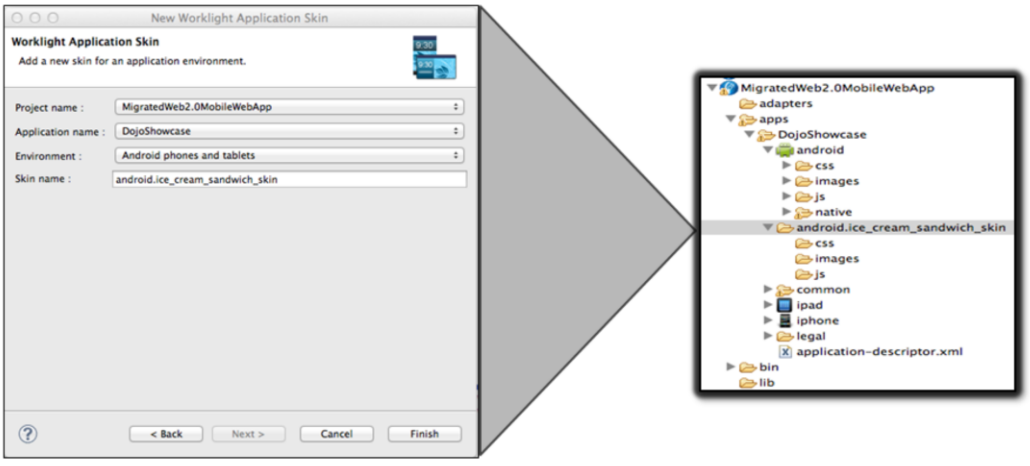
This section shows the transition to the Worklight Application Development model inside of Worklight Studio. In this application, you select the various target platforms (known as environments) where the application is going to be deployed.

The target platforms for this application are Android, iPhone, and iPad for compatibility with the existing sample application. After you create the environments, create a set of directories. The directories can be categorized into two distinct types. The first is the common directory and second is device specific.

The “common” directory is platform-independent for hosting web resource such as images, CSS, JavaScript and HTML. For most web application, much of the code is platform independent and can be reused for each of the target devices.

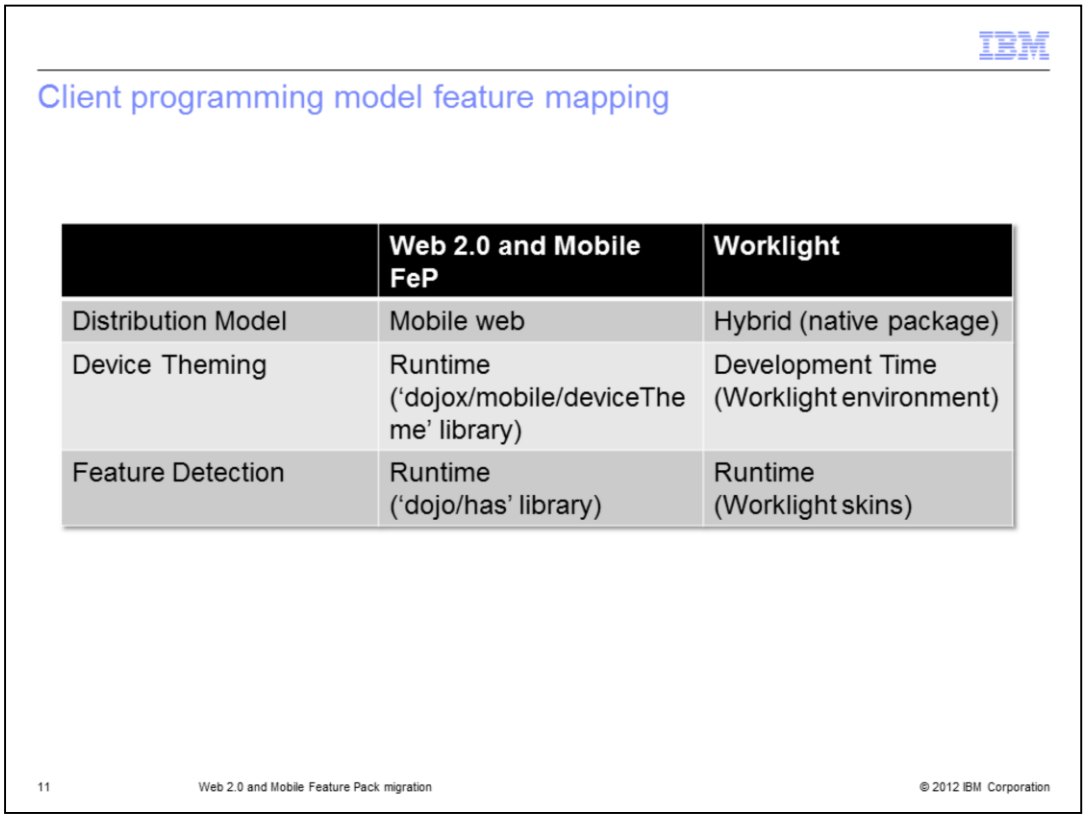
The device specific directories reflect the platform name - android, ipad, and iphone - and contain platform dependent web resources including images, CSS, JavaScript, and HTML. You can also contribute native libraries for the target platform, like Cordova plug-ins. In this sample application, there are no custom native libraries being deployed with the application but the support is there depending upon the application’s usage scenario.

Worklight application skins



Mobile are viewed as a disruptor to the technology industry because of the rate at which these devices are released and the variations of the devices in terms of feature functions. This is common on both platforms but even more so on the Android platform. To provide cross-device support for a variety of mobile devices, Worklight 5.0 provides a finer grained model of the Worklight Environment referred to as Worklight Application Skin. The Worklight Application Skin is a user interface variant of an application that can be applied during run time based on run-time device properties, like screen resolution or form factor.

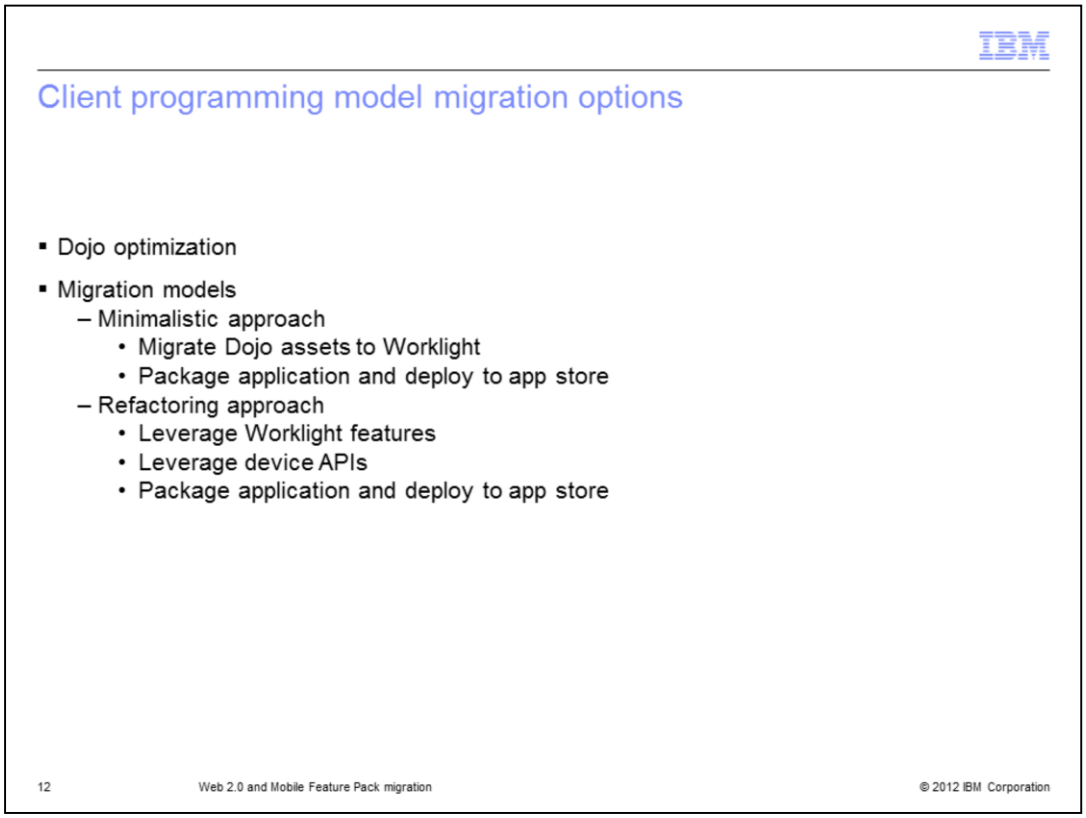
This slide shows how you can further extend the Dojo Mobile Showcase to use custom theming for the Android 4.0 “Ice Cream Sandwich” skin and augment the dojo themes to take advantage of the latest CSS rendering technologies that it supports.



There are some differences between the two models for developing applications.

The mobile web provides a quick and iterative deployment model from development to deployment of the application. Since the mobile web is hosted inside of your enterprise, the ability to deploy, update and maintain the application is simplified from a management perspective. With Worklight, the deployment model is through an application store. While the ability to quickly update the content in a public app store is more involved, Worklight's direct update feature provides a quick and efficient model similar to the mobile web use case to provide application updates. Packaging with Worklight lets you deliver applications through the app store, which provides a higher visibility for the application (marketing presence) and improved speed and offline support, since the application is hosted on the actual device versus loaded remotely.

Both device theming and feature detection models provide device level and finer grained feature detection support. The main difference between the two models is where device theming is applied. In the Worklight model, the various themes are packaged based upon the device whereas the Dojo model packages all versions within the application and then selects the appropriate theme at runtime.



One of the key design elements of Dojo is the modularized packaged structure. This package structure results in an increased number of web resources as you include more and more widgets in your application. However, there is a level of optimization that should be done for all mobile applications regarding custom dojo profiles and layer builds. In earlier releases of Dojo, you had to provide a list of Dojo components used by the application and generate the build with a script. In the most recent release of Dojo, there is a set of default layers that applications can include. The result is a much smaller deployment and increased performance of your application. Depending upon what version of Dojo you are using and whether you want to upgrade your version of Dojo for your application to the latest level, you can pick which option you want to use. This is required for Android since Android cannot load resources containing underscores in name (Dojo has many files with this convention to denote private modules).

Migrating to Worklight can be completed in stages.

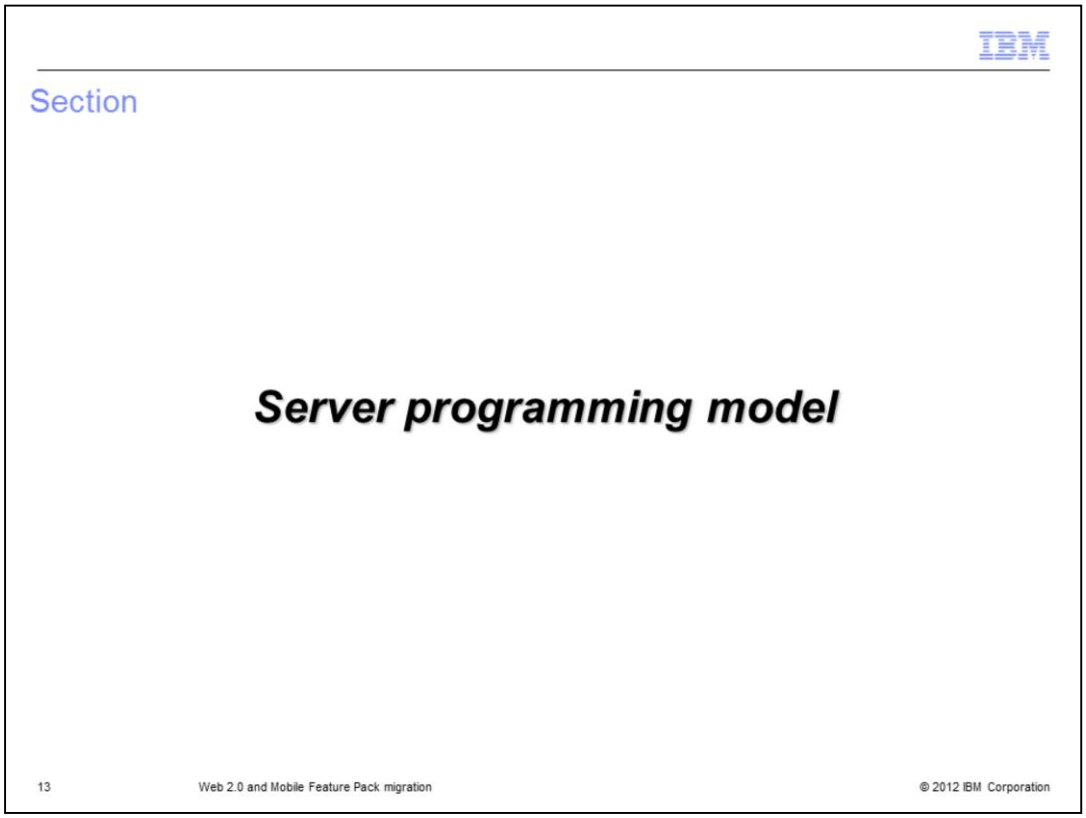
The first stage is “Minimal refactoring to enhance the mobile experience”. In this model, the main process is migrating the project structure to Worklight Environment Model described earlier in the presentation.

This requires a few steps: Copy web resources to the “common” directory, use “has” feature detection (dojo/has), and use the device theme feature (dojox/mobile/theme)

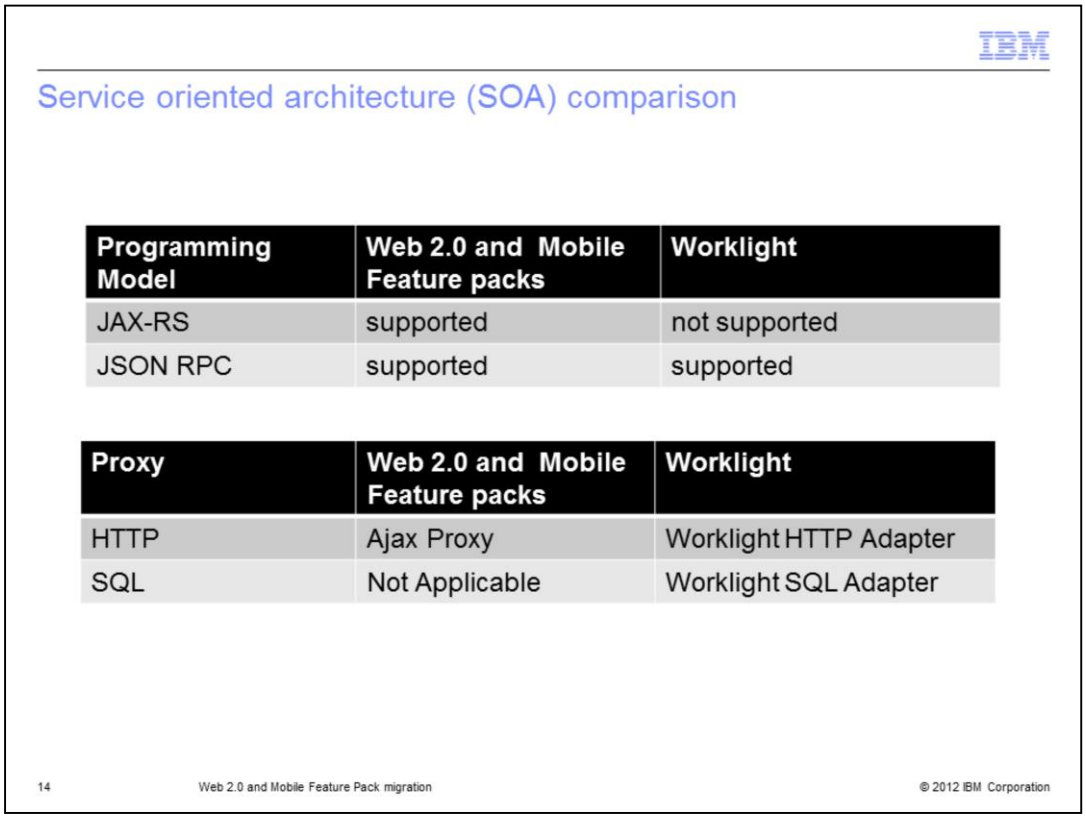
As this point, you have completed the first stage.

The second stage is “Maximal reuse of services and user interface code with Worklight”. In this model, continue refactoring and take advantage of Worklight’s environment and support for skins. Create platform-based directories for web resource overrides for each platform and transition to use “has” feature detection (dojo/has) for skin-specific behaviors.

The third stage uses the advanced features of Worklight. This phase is “Extending the mobile application to Worklight”. In this model, you enable security for features like application authentication and device features like camera interaction using Cordova.



This section covers the server programming model



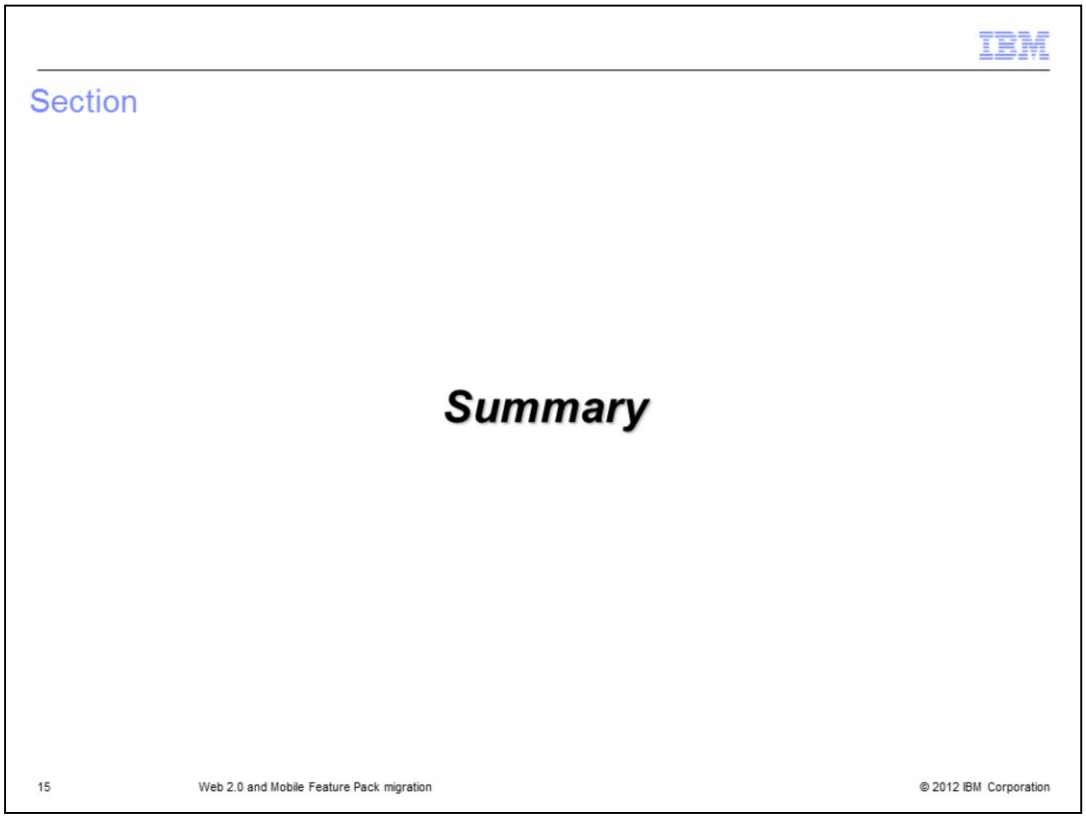
For existing services written using JAX-RS hosted on WebSphere, you can still use the existing REST Services from the mobile client application - no migration is needed. Upgrading to the security integration with Worklight will require proxying of existing REST Services through the HTTP adapter by using the JSON-RPC services. The resultant services will then be hosted in separate .EAR and .WAR files from the Worklight Application. Although hosted in separate files, the same domain sandbox restrictions on host and port may apply.

Security Integration with Worklight really shines when attempting to port over existing services written using JSON-RPC. There is only minimal proxying of existing RPC Services through the Worklight HTTP Adapter using JSON-RPC. These services are also hosted in separate .EAR and .WAR files from the Worklight application, but the same domain sandbox restrictions on host and port may apply.

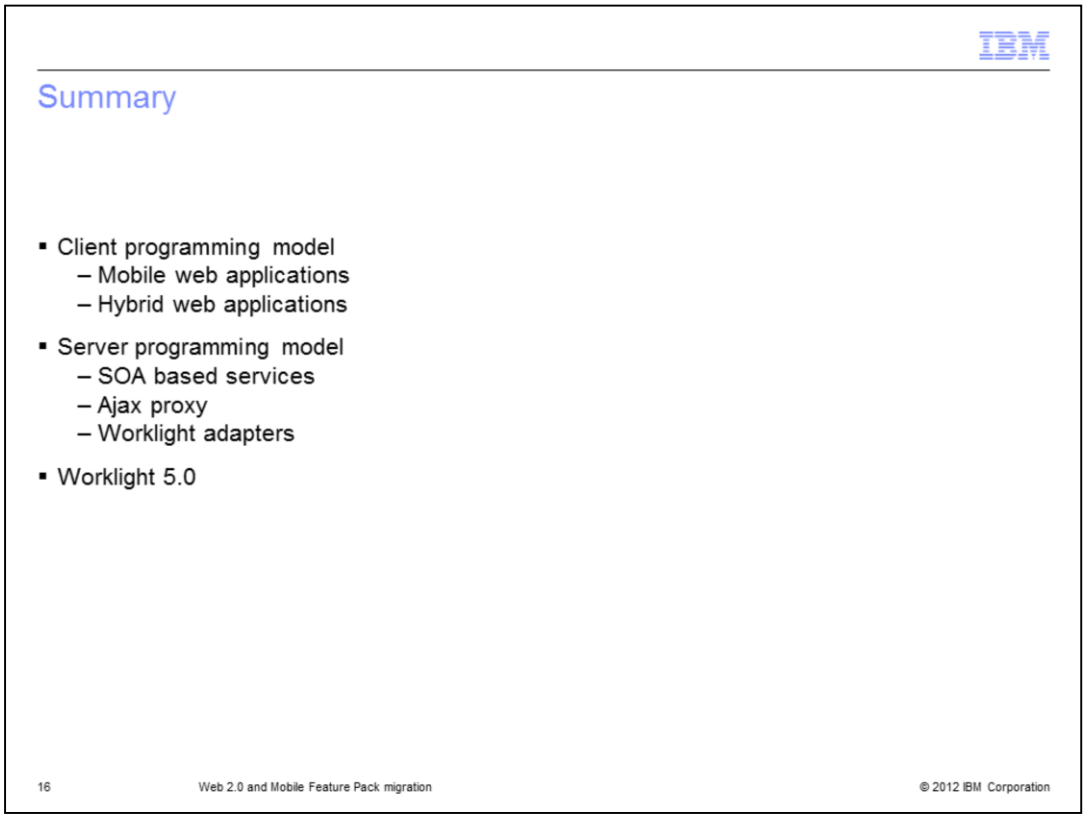
If you prefer to keep the WebSphere Security Model, no migration is needed. You can use the existing RPC services from the mobile client application.

For external services requiring Ajax proxies on WebSphere, there are two options for continuous use. One, if you stay with the current WebSphere Security Model, you do not need to migrate. You can use your existing Ajax proxy from the mobile Worklight client application. Two, if you choose to implement security integration with Worklight, the current services will require proxying of existing HTTP Requests through the Worklight HTTP Adapter using JSON-RPC. These services can be hosted externally from the Worklight application. For more advanced features like whitelisting and header filtering, the current Ajax proxy can be used on top of the implemented HTTP Adapter.

A benefit that comes with the upgrade to the Worklight security integration is the ability to proxy SQL using the Worklight SQL adapter. This functionality does not exist currently in the WebSphere security model.


IBM
Section
Summary
15
Web 2.0 and Mobile Feature Pack migration
© 2012 IBM Corporation

In summary,



This presentation discussed both the client and server programming models for migrating to IBM Worklight 5.0, while detailing how you can extend mobile applications to use the latest features of Worklight 5.0. Whether you are using JSON-RPC or JAX-RS for SOA based programming model, or proxying requests using the Ajax proxy, the new platform provides a rich and innovative model for deploying applications to the public app store.

Trademarks, disclaimer, and copyright information