**Worklight**

**Pankaj P. Babhare Khushbu W. Patre Dipti M. Ughade**

*VI Semester Dept. of I.T., VI Semester Dept. of I.T., VI Semester Dept. of I.T.,*

*VDCET., Mouda VDCET., Mouda VDCET., Mouda*

pankajbabhare010@gmail.com 20khushbu13@gmail.com dipaliuhade@gmail.com

 8551831843 9404082477 7276253547

***Abstract-: IBM Worklight provides an open, comprehensive and advanced mobile application platform for smartphones and tablets, helping organizations of all sizes to efficiently develop, connect, run and manage HTML5, hybrid and native applications Leveraging standards-based technologies and tools, the platform ships with a comprehensive development environment, mobile-optimized middleware, and an integrated management, and analytics console, supported by a variety of security mechanisms.IBM Worklight enables the creation of rich, cross-platform apps without the use of code translation, proprietary interpreters or unpopular scripting languages, while reducing the time to market, cost and complexity of development and enabling a better user experiences across a variety of mobile devices.IBM Worklight is part of the****IBM Mobile Foundation****family of products that provides the essential elements needed for complete mobile development, deployment and management within a business.***

1. **Introduction**

With IBM Worklight, you can simplify and accelerate the development, testing, and delivery of your mobile apps.IBM Worklight offers an Eclipse-based visual development and a server environment for you to create native, hybrid, and standard web mobile applications, and maximizes code reuse across different mobile phone platforms.

In a mobile development platform, cross-platform portability of the application code is critical for mobile device application development. Various methods exist to achieve this portability. With IBM Worklight, you can develop multiplatform applications by using Worklight Studio, which is a mobile development studio, to address the requirements of the organization.

**Work light Studio tasks:**

Develop rich HTML5, hybrid and native applications for all supporting modern devices by using native code, a bidirectional WYSIWYG, and standard web technologies and tools.

* Maximize code sharing by defining custom behavior and styling guidelines that match the target environment.
* Access device APIs by using native code or standard web languages over a uniform Apache Cordova bridge.
* Use both native and standard web languages within the same application to balance development efficiency and a rich user experience.
* Use third-party tools, libraries, and frameworks such as JQuery Mobile, Sencha Touch, and Dojo Mobile.
* Implement runtime skins to build apps that automatically adjust to environment guidelines such as form factor, screen density, HTML support, and UI input method.
1. **Worklight Components**
* **IBM Worklight Studio** - An Eclipse-based IDE, allowing developers to perform all the coding and integration tasks that are required to develop a fully operational application.
* **IBM Worklight Server –** The Java-based Server is a scalable gateway between applications, external services, and the enterprise backend infrastructure. The Server contains security features to enable connectivity, multi-source data extraction and manipulation, authentication, direct update of web and
* hybrid apps, analytics and operational management functions.
* **IBM Worklight Device Runtime Components -**Client-side runtime code that embeds server functionality within the target-environment of deployed apps.
* **IBM Worklight Console** - A web-based UI dedicated for the on-going monitoring and administration of the Worklight Server and its deployed apps, adapters and push notifications
1. **Worklight Studio**

In a mobile development platform, cross-platform portability of the application code is critical for mobile device application development. Various methods exist to achieve this portability. With IBM Worklight, you can develop multiplatform applications by using Worklight Studio, which is a mobile development studio, to address the requirements of the organization.



Fig . components of worklight

1. **Worklight Studio tasks:**
* Develop rich HTML5, hybrid and native applications for all supporting modern devices by using native code, a bidirectional WYSIWYG, and standard web technologies and tools.
* Maximize code sharing by defining custom behavior and styling guidelines that match the target environment.
* Access device APIs by using native code or standard web languages over a uniform Apache Cordova bridge.
* Use both native and standard web languages within the same application to balance development efficiency and a rich user experience.
* Use third-party tools, libraries, and frameworks such as JQuery Mobile, Sencha Touch, and Dojo Mobile.
* Implement runtime skins to build apps that automatically adjust to environment guidelines such as form factor, screen density, HTML support, and UI input method.
1. **Worklight Server**

The Worklight Server is a runtime container for the mobile applications you develop in Worklight Studio. It is not an application server in the Java Platform, Enterprise Edition (JEE) sense. It acts as a container for Worklight application packages, and is in fact a collection of web applications (optionally packaged as an EAR file) that run on top of traditional application servers.

Worklight Server is designed to integrate into the enterprise environment and use its existing resources and infrastructure. This integration is based on adapters that are server-side software components responsible for channeling back-end enterprise systems and cloud-based services to the user device. You can use adapters to retrieve and update data from information sources, and to allow users to perform transactions and start other services and applications.

1. **Worklight Server tasks:**
* Empower hundreds of thousands of users with transactional capabilities and enable their direct access to back-end systems and cloud-based services.
* Configure, test, and deploy descriptive XML files to connect to various back-end systems by using standard Worklight Studio tools.
* Directly update deployed hybrid and web applications, without going through the different app stores (subject to the terms of service of the vendor).
* Automatically convert hierarchical data to JSON format for optimal delivery and consumption.
* Enhance users interaction with a uniform push notification architecture.
* Define complex mash up of multiple data sources to reduce overall traffic.
* Integrate with the existing security and authentication mechanisms of the organization.



Fig : worklight server

1. **VII.Worklight Device Runtime Components**

IBM Worklight provides client-side runtime code that embeds server functionality within the target environment of deployed apps. These runtime client APIs are libraries that are integrated into the locally stored app code. They complement the Worklight Server by defining a predefined interface for apps to access native device functions. Among these APIs, IBM Worklight uses the Apache Cordova development framework. This framework delivers a uniform bridge between standard web technologies (HTML5, CSS3, JavaScript) and the native functions that different mobile platforms provide.

The Worklight device runtime components provide the following functions:

* Mobile data integration: connectivity and authentication APIs
* Security features: on-device encryption, offline authentication, and remote disablement of apps
* Cross-platform support: runtime skins, UI abstractions, and HTML5 toolkits compatibility
* Mobile client functionality: hybrid app framework, access to device APIs and push notification registration
* Reports and analytics: built-in reports and event-based custom reporting
* Resource serving: direct update of app web resources and HTML5 caching

## Worklight Console

The Worklight Console is used for the control and management of the mobile organization, from managing deployed applications to collecting and analyzing user statistics.:

* Monitor all deployed applications, adapters, and push notification rules from a centralized, web-based console.
* Assign device-specific identifiers (IDs) to ensure secure application provisioning.
* Remotely disable applications by using preconfigured rules of app version and device type.
* Customize messages that are sent to users on application launch.
* Collect user statistics from all running applications.
* Generate built-in, pre-configured user adoption and usage reports.
* Configure data collection rules for application-specific events.
* Export raw reporting data to be analyzed by the BI systems of the organization.

## Application Center

With the Application Center, you can share mobile applications that are under development within your organization in a single repository of mobile applications. Development team members can use the Application Center to share applications with members of the team. This process facilitates collaboration between all the people who are involved in the development of an application.

Your company can typically use the Application Center as follows:

1. The development team creates a version of an application.
2. The development team uploads the application to the Application Center, enters its description, and asks the extended team to review and test it.
3. When the new version of the application is available, a tester runs the Application Center installer application, which is the mobile client. Then, the tester locates this new version of the application, installs it on their mobile device, and tests it.
4. After the tests, the tester rates the application and submits feedback, which is visible to the developer from the Application Center console.

The Application Center is aimed for private usage within a company, and you can target some mobile applications to specific groups of users. You can use the Application Center as an enterprise application store.

With the Application Center, you can manage native or hybrid applications that are installed on mobile devices. The Application Center supports applications that are built for the Google Android platform and the Apple iOS platform, but does not target mobile web applications.



**Fig: Different Mobile Approaches**

****

**Fig: Coding Environment of worklight**

1. **Introducing the approaches**

**Native apps**

Native apps have binary executable files that are downloaded directly to the device and stored locally. The installation process can be initiated by the user or, in some cases, by the IT department of the organization. The most popular way to download a native app is by visiting an app store, such as Apple’s App Store, Android’s Marketplace or BlackBerry’s App World, but other methods exist and are sometimes provided by the mobile vendor. Once the app has been installed on the device, the user launches it like any other service the device offers. Upon initialization, the native app interfaces directly with the mobile operating system, without any intermediary or container. The native app is free to access all of the APIs that are made available by the OS vendor and, in many cases, has unique features and functions that are typical of that specific mobile OS.

Table: Difference between various SDK

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Apple iOS** | **Android** | **Blackberry OS** | **Windows Phone** |
| **Languages** | Objective-C, C, C++ | Java (some C, C++) | Java | C#, VB.NET and more |
| **Tools** | Xcode | Android SDK | BB Java Eclipse Plug-in | Visual Studio, Windows Phonedevelopment tools |
| **Packaging format** | .app | .apk | .cod | .xap |
| **App stores** | Apple App Store | Google Play | Blackberry App World | Windows Phone Marketplace |

**Hybrid apps**

The hybrid approach combines native development with web technology. Using this approach, developers write significant portions of their application in cross-platform web technologies, while maintaining direct access to native APIs when required. The native portion of the application uses the operating system

APIs to create an embedded HTML rendering engine that

serves as a bridge between the browser and the device APIs. This bridge enables the hybrid app to take full advantage of all the features that modern devices have to offer.

App developers can choose between coding their own bridge ortaking advantage of ready-made solutions such as PhoneGap—open-source library that provides a uniform JavaScript interface to selected device capabilities that is consistent across operating systems.

1. **Setting up IBM Worklight**

Setting up an IBM Worklight development environment is not a complex task. Since IBM Worklight is a plug-in for Eclipse, you need two pieces of software, Eclipse, and the IBM Worklight plug-in for Eclipse. Eclipse is a free and open-source IDE that is popular for developing Java applications, however it is highly flexible and can be used for a number of other development languages and frameworks. In addition to installing IBM Worklight Studio, you can install other plug-ins onto Eclipse and use the same installation for other kinds of projects as well.

It is important to note that while —as of writing—is Eclipse Juno, or Eclipse 4.2 IBM Worklight is designed to work with Eclipse Indigo, or Eclipse 3.7 which was the last of the 3.x series of Eclipse. Also, there are many different editions of Eclipse, each catering to different kinds of developers, and they come fitted with different sets of plugins. What we need is Eclipse Classic, which should be approximately 170MB download.



Fig: Worklight Adapters

1. **Creating a new Worklight Project**

Now that you have a fresh installation of Worklight let’s start an actual Worklight project. You can start a new project in Eclipse by clicking on one of the project types listed in File>New submenu. If “Worklight Project” is listed there, click on it, else click on “Other…” and select “Worklight Project” in the dialog that pops up. You will be asked to name the project. Since we are building a To-Do application, name your project “Todo Project”.

1. **Worklight Project Structure**

For our purposes, the most important folder is the apps folder. This is the folder in which we will be placing all the

code we write for our application. However each of these folders and files have an explicit purpose, and will be useful while developing Worklight apps. We will go over all the important ones in turn, starting from the top. Right at the top of the project explorer you will see entries such as “WL Server Library”, “JRE System Library”, “server/ java” and “JavaScript Resources”, these are all references needing by your application to function. What follows is a folder called “adapters”. Adapters are what allow your client side application to connect to your server-side services. They define procedures that can be performed on the server side, and translate data the server and the client.



smartphones are still in a rather nascent stage. They might seem to be like they have been around for long enough, but you just need to look at how long we have had computing in general, and you will realise that there are still

ways to go. The first computers go as far back as the 1940s, but they have less in common with the personal computers of today than smartphones do. Still computers have been around for a long time, and smartphones as a form factor are still only a small blip in that history.

There are still some things that are being figured out in the field of mobile devices that have been taken for granted in more traditional form factors.

Things such as which operating systems will really survive, which will be forgotten in a few years’ time.

The current state is one of flux. A must-have platform that is so popular that it must be supported might not even be around in a few years time. Since smartphones started becoming more mainstream—rather than just something owned by a few and coveted by the many—a lot of platforms have come and gone. Symbian and Windows Mobile are particularly good examples. Symbian has been abandoned by Nokia in favour of Windows Phone, and while Windows Phone is a successor to Windows Mobile, it is different enough to be its own OS.

1. **Develop rich cross-platfrom apps**

Worklight supports the following platforms:

* iOS 4 and higher (tablets and mobiles)
* Android 2.1 and higher (tablets and mobiles)
* Blackberry OS 5 and higher
* Windows Phone 7 and higher
* Any web browser that supports at least HTML4, CSS2.1, and JavaScript1.5 (IE7+ and all recent versions of Chrome, Firefox and Safari)
* Facebook apps
* iGoogle Gadgets
* Web widget for embedding in iframe
* Windows gadgets (XP, Vista, and 7)
* Mac OSX Widgets
1. **Enabling mobile security with IBM Worklight**
* On-device data protection
* Encrypted on-device storage
* Offline authentication
* Authenticity testing
* Direct update
1. **Integrating IBM Worklight with enterprise security**

IBM Worklight integrates with the existing security infrastruc-ture of an enterprise in a number of ways. All of the integration with various enterprise security components are made possible by using the powerful framework provided by IBM Worklight. This framework is part of the core IBM Worklight runtime used for handling authentication and data protection.Typical enterprise security requirements include integrating with traditional Lightweight Directory Access Protocol (LDAP) servers for user authentication, working behind application firewalls and reverse proxies and integrating with security gateways that protect backend resources.

1. **Steps to IBM Worklight mobile application development**
2. **Discovery process**

Step one begins with understanding your mobile business requirements and defining your enterprise mobile strategy. The discovery phase plays an important role in the overall mobile application development lifecycle. It sets the mobile visions and strategy by analyzing your mobile challenges, goals, and constraints.For example, you need to identify the **types** of mobile applications you need to develop, which could be mobile web, native, hybrid, or a combination of these. You also have to decide which mobile platforms to support; for example, iOS, Android, Blackberry, Windows® Phone, and so on.

**2. Become a registered developer on the supported mobile platforms**

In order to be able to deploy or publish your application via the various mobile app stores (such as Apple’s App Store or Google Play), all mobile platforms require an organization or individual to be a registered developer. Only registered developers can submit applications to a platform’s associated app store. Therefore, you need to register to these platform programs in the early stage of your development lifecycle.

**iOS:** To test an application on a real iOS device or to prepare your app for app store release, you need to enroll in **Apple's**[**iOS** Developer Program](https://developer.apple.com/programs/ios/) as either an individual developer or as a company. There is a fee associated with this registration.

**Android:** Before you can publish an Android application to Google Play, you or your organization needs to set up a [Google Play](https://play.google.com/apps/publish/signup) account. This is a three-step process, which includes creating a developer profile, agreeing to the Developer Distribution Agreement, and paying a registration fee using Google Checkout.

**Blackberry:** To publish a Blackberry application to Blackberry App World, you first need to have a [vendor account](https://appworld.blackberry.com/isvportal/home/login.seam?pageIndex=1&cid=26861). Developers without a vendor account are not be able to submit applications for publication to BlackBerry App World.

**Windows Phone:** Similarly, you must be a registered Windows Phone developer to submit your applications to the Windows Phone Marketplace. Registration can be completed at the [Microsoft App Hub](http://create.msdn.com/en-US/).

**3. Prepare Worklight mobile application development environment**

**Proper setup of your Worklight development environment is a vital step in the develoing and testing of mobile applications. In a nutshell, you will install Eclipse and IBM Worklight Studio as your development IDE for Worklight mobile applications. Supported mobile platform SDKs will have to be** installed as well. Optionally, you might also wish to set up a team development environment for source control management.

Worklight Studio is an Eclipse-based IDE; you need to install Eclipse on your workstation first before installing Worklight Studio. Worklight Studio is supported on Windows, Mac OS, and Linux® operating systems. If you plan to develop Worklight applications for iOS environments, you can do so on any of these operating systems. However, due to restrictions set by Apple, you can only compile an iOS project on a Mac OS machine.

1. **Installing Eclipse**

The current version of Worklight Studio requires Eclipse Indigo (3.7.2). You can [download Eclipse IDE](http://www.eclipse.org/downloads/packages/release/indigo/sr2) at no cost.

1. **Installing IBM Worklight Studio**

You can install IBM Worklight Studio after Eclipse has been installed. There are three editions of Worklight Studio available: Developer (which is available for download at no cost to get you started), Consumer, and Enterprise. Worklight Studio can be installed from Eclipse Marketplace, or using IBM Installation Manager, or as an Eclipse plugin. For installation details, see the [IBM Worklight product library](http://www-01.ibm.com/software/mobile-solutions/worklight-library/).

1. **Installing mobile SDKs**

If you plan to develop hybrid or native mobile applications with Worklight, you need to install and configure the SDKs (and development IDEs) for the associated supported mobile platforms. The procedures might vary, depending on the platform, but here are steps you are likely to encounter with some of the popular mobile platforms.

* + **iOS:** You must [download Xcode](https://developer.apple.com/xcode/), which is an Apple IDE for developing iOS and Mac applications. You can use Xcode to manage your testing devices, use an iPhone or iPad simulator, and install applications on an iPhone or iPad device.
	+ **Android:** You must install the [Android SDK](http://developer.android.com/sdk/) and [Android Development Tools (ADT) plug-in for Eclipse](http://developer.android.com/sdk/eclipse-adt.html). The Android SDK provides the tools and APIs that are required to develop applications on the Android platform by using the Java™ programming language. The ADT plug-in for Eclipse is an integrated environment in which you can build rich Android applications.
	+ **BlackBerry:** WebWorks application development requires that you download and install Blackberry Ripple Emulator, Blackberry WebWorks SDK, and Blackberry Simulator.
	+ **Windows Phone:** You must download and install [Microsoft Visual Studio 2010/2012 and the Windows Phone SDK](http://dev.windowsphone.com/en-us/downloadsdk). You must also download and install [Zune software](http://www.zune.net/en-US/products/software/download/default.htm) in your development environment to run your applications on a Windows Phone handset.
1. Prepare team development environment

Preparing the team development environment using, for example, IBM Rational® Team Concert, is an excellent idea for managing highly dynamic mobile application development. Keep in mind, however, that when managing Worklight applications with a source control system, some files generated by Worklight projects should not be added to the source control system.



Fig: steps of Application Development

1. **Advantages**
* Application security
* Direct updates
* Enterprise back end connectivity
* Administration console to manage all applications

**Conclusion**

As mobile apps continue to take a central role in the business landscape, organizations around the world are mobilizing a growing number of mission-critical services. Many companies are striving to find the optimal development approach to achieve their goals, but what many are quickly realizing is that each approach carries inherent limitations and no single approach can address all the growing needs and complexity of the modern mobile enterprise. As this paper attempts to show, the answer lies not in one development approach, but rather in a flexible solution—one that can harness the benefits that each provides and support not only the development of a first mobile app, but of all future apps, regardless of their development approach. Worklight Developer Edition 5.0 helps organizations of all sizes to efficiently develop, run, and manage HTML5, hybrid, and native applications using a powerful and flexible mobile IDE. Next-generation mobile middleware, end-to-end security, and integrated management and analytics capabilities are built in. With this release, enterprise architects and application developers can meet the challenges of mobile team development, enterprise integration and application management.

**Bibliography:**

* [www.ibm.com/software/in/**worklight**](http://www.ibm.com/software/in/worklight)
* [www.ibm.com/software/mobile-solutions/**worklight**/features](http://www.ibm.com/software/mobile-solutions/worklight/features)
* <http://www.ibm.com/developerworks/downloads/ws/worklight/index.html>
* <https://www.ibm.com/developerworks/mobile/mobileforum.html>