

Agile Mobile Apps for the Enterprise with IBM Worklight

See how would one "normally" develop a Worklight app?



conclusions

- We can take advantage of the existing infrastructure
- We can reuse existing code
- We can reuse existing components
- We can reuse existing databases
- We can reuse existing security
- We can reuse existing business logic



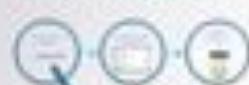
Automated Adapter Testing

- Create test cases for the adapter
- Run the test cases against the adapter
- Identify errors in the adapter
- Fix the errors in the adapter



Agile Mobile Apps for the Enterprise with IBM Worklight

See how would one "normally" develop a Worklight app?



conclusions

- We continue to confirm the importance of continuous integration and automated testing - we are moving to code reuse
- Continuous integration and automated testing can help us achieve better quality and increased delivery speed in general



Automated Adapter Testing

- Integration of the adapter with the mobile application
- Configuration of the adapter
- Configuration of the adapter
- Configuration of the adapter



Continuous Integration

- Continuous integration
- Continuous integration
- Continuous integration

Agile Mobile Apps for the Enterprise with IBM Worklight

Andrew Ferrier

andrew.ferrier@uk.ibm.com

Donal Spring

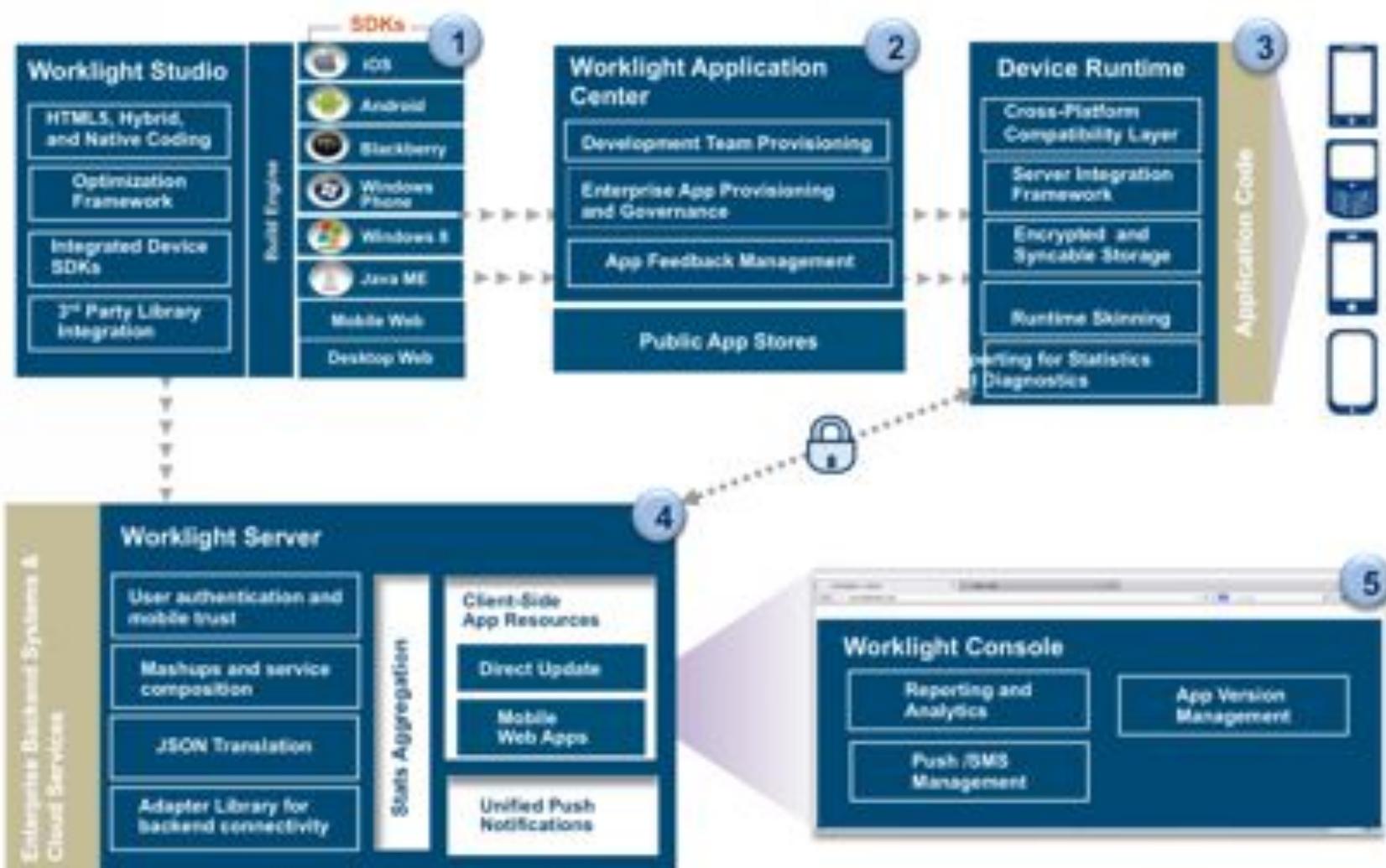
donalspr@uk.ibm.com

UK WebSphere User Group, 25th March 2014

Agenda

- What is IBM Worklight?: *A Review*
- Continuous Integration
- Automated Adapter Testing
- Preparing for Production (versioning adapters, managing authentication, updating apps...)

What is IBM Worklight - The Bits



Worklight Studio

HTML5, Hybrid,
and Native Coding

Optimization
Framework

Integrated Device
SDKs

3rd Party Library
Integration

Build Engine

SDKs

1



iOS



Android



Blackberry



Windows
Phone



Windows 8



Java ME

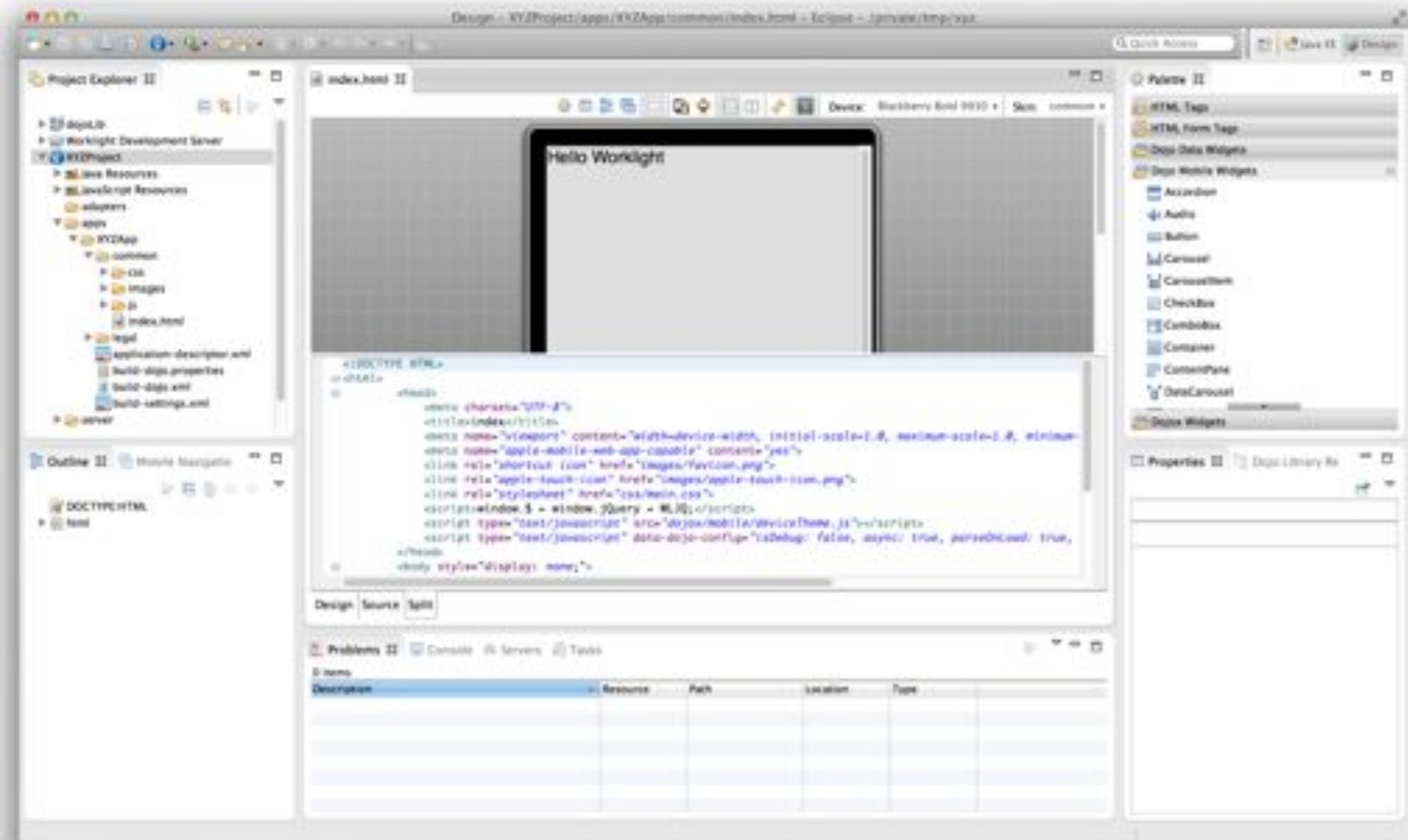


Mobile Web



Desktop Web





Worklight Server

4

User authentication and
mobile trust

Mashups and service
composition

JSON Translation

Adapter Library for
backend connectivity

Stats Aggregation

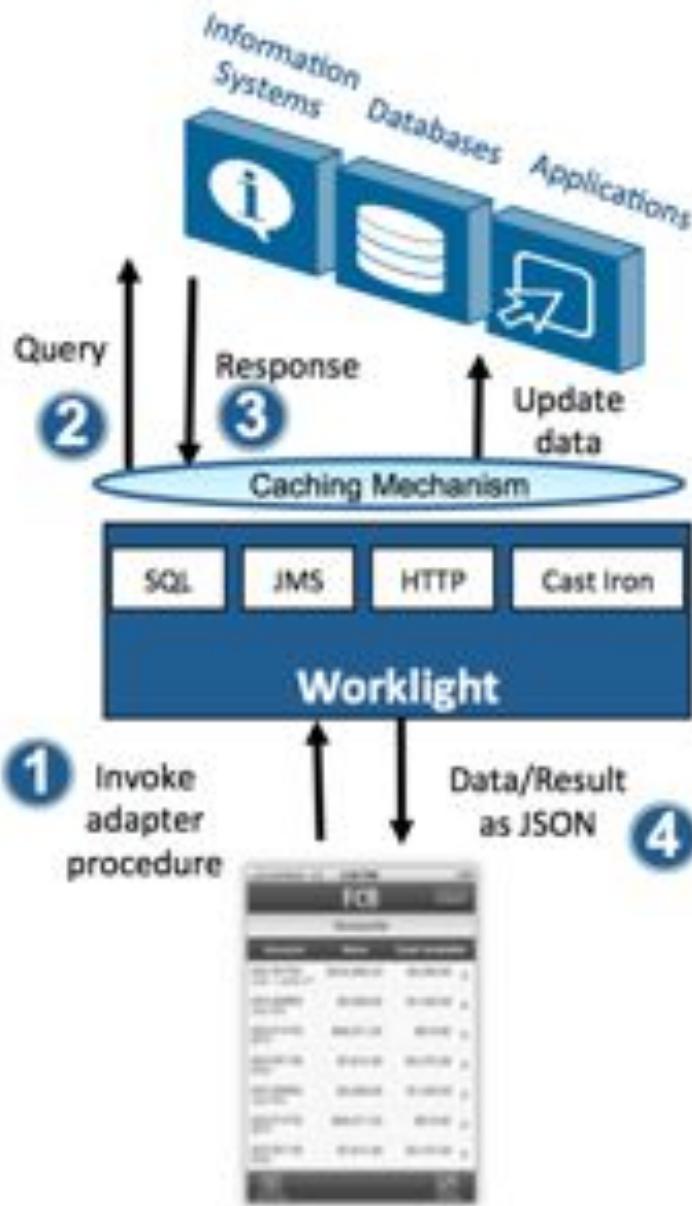
Client-Side
App Resources

Direct Update

Mobile
Web Apps

Unified Push
Notifications

What are Worklight Adapters?



- Standardised way to "mobilise" services
- Essentially a lightweight integration layer
- Much of the real-world use is the HTTP Adapter to connect to existing RESTful Services, i.e. mapping to **Create** (POST), **Read** (GET), **Update** (PUT) and **Delete** (DELETE) verbs.
- There are also other adapter types (SQL, JMS, Cast Iron)

What do they look like?

Client-side (in the app)

```
WL.Client.invokeProcedure({
    adapter: 'Adap1',
    procedure: 'getStories',
    parameters: [],
}, {
    onSuccess: function(object) {
        // .. Do some stuff with the result
    },
    onFailure: function(object) {
        // .. Handle the error situation
    }
});
```

Server-side (on the WL Server)

```
<xsd:version="1.0" encoding="UTF-8">
<wl:adapter name="Adap1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wl="http://www.worklight.com/integration"
    xmlns:http="http://www.worklight.com/integration/http">
    <displayName>Adap1</displayName>
    <description>Adap1</description>
    <connectivity>
        <connectionPolicy xsi:type="http:HTTPConnectionPolicyType">
            <protocol>http</protocol>
            <domain>www4.com.com</domain>
            <port>80</port>
        </connectionPolicy>
        <loadConstraints maxConcurrentConnectionsPerNode="2" />
    </connectivity>
    <procedure name="getStories"/>
</wl:adapter>
```

```
function getStories(param) {
    path = getPath(param);

    var input = {
        method : 'get',
        returnedContentType : 'xml',
        path : path,
        transformation : {
            type : 'xslFile',
            xslFile : 'filtered.xsl'
        }
    };

    return WL.Server.invokeHttp(input);
}
```

```
WL.Client.invokeProcedure({
    adapter: 'Adap1',
    procedure: 'getStories',
    parameters: [],
}, {
    onSuccess: function(object) {
        // .. Do some stuff with the result
    },
    onFailure: function(object) {
        // .. Handle the error situation
    }
});
```

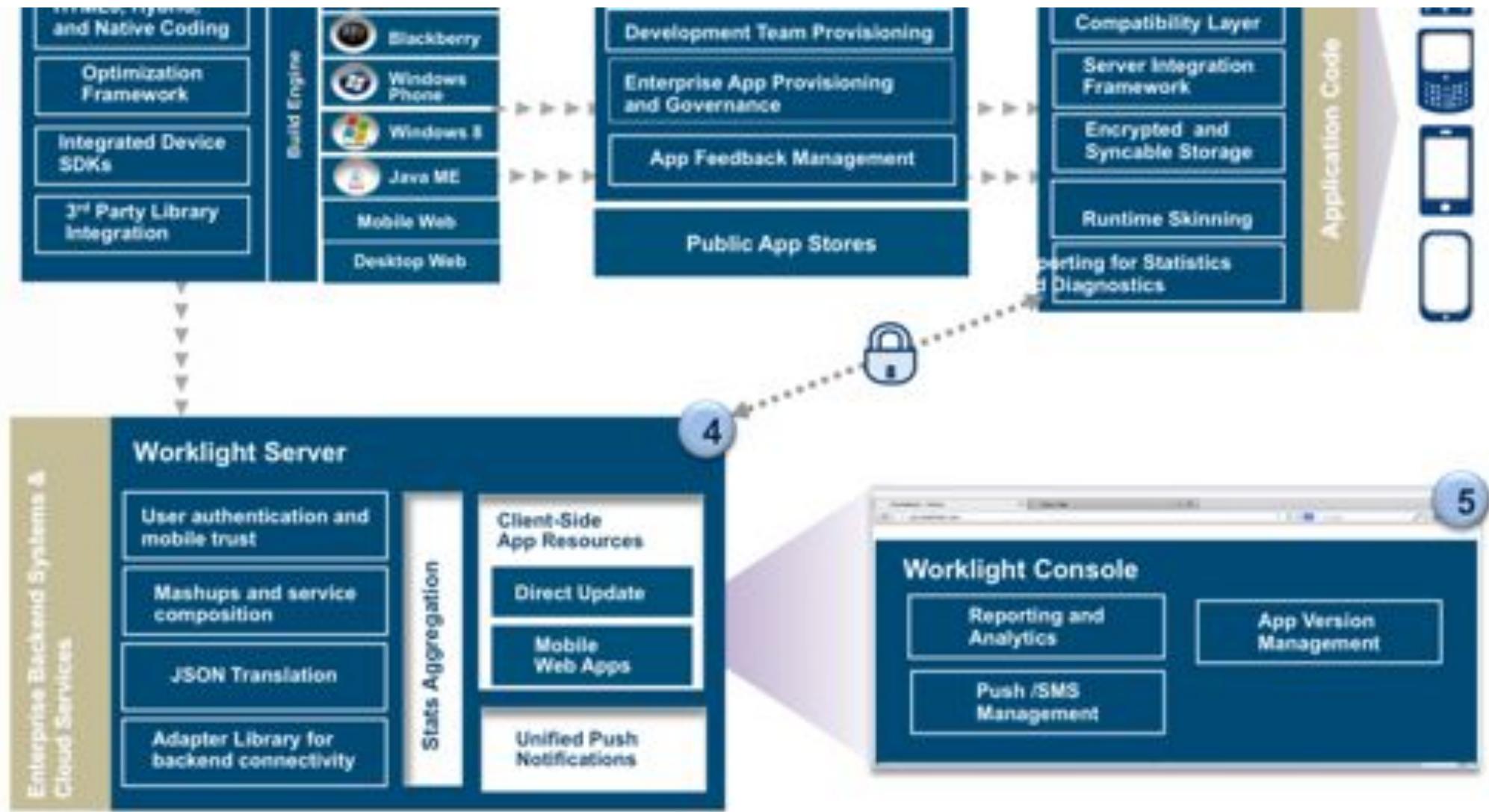
WSDL SERVICE DESCRIPTION

```
<?xml version="1.0" encoding="UTF-8"?>
<wl:adapter name="Adap1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wl="http://www.worklight.com/integration"
  xmlns:http="http://www.worklight.com/integration/http">

  <displayName>Adap1</displayName>
  <description>Adap1</description>
  <connectivity>
    <connectionPolicy xsi:type="http:HTTPConnectionPolicyType">
      <protocol>http</protocol>
      <domain>rss.cnn.com</domain>
      <port>80</port>
    </connectionPolicy>
    <loadConstraints maxConcurrentConnectionsPerNode="2" />
  </connectivity>

  <procedure name="getStories"/>
</wl:adapter>
```

```
function getStories(param) {  
    path = getPath(param);  
  
    var input = {  
        method : 'get',  
        returnedContentType : 'xml',  
        path : path,  
        transformation : {  
            type : 'xslFile',  
            xslFile : 'filtered.xsl'  
        }  
    };  
  
    return WL.Server.invokeHttp(input);  
}
```



IBM Worklight Console

Welcome, Guest | Logout | About

Catalog Services Push Notifications

Display application or adapter

IBM App Center (App Center is the Worklight Enterprise Application Store)

Last deployed at 2014-03-18 16:08

 MobileApp	<input type="button" value="Edit"/> <input type="button" value="Print"/> Version 1.0 Active	<input type="checkbox"/> Lock this version in	Security Test: Default App Authentication: <input checked="" type="radio"/> Access Token <input type="radio"/> Session <input type="radio"/> Device <input type="radio"/> User Device Authentication: Default User Authentication: Default
--	--	---	---

MobileApp (MobileApp)

Last deployed at 2014-03-18 18:33

 MobileApp	<input type="button" value="Edit"/> <input type="button" value="Print"/> Version 1.0 Active	<input type="checkbox"/> Lock this version in	Security Test: Default App Authentication: <input checked="" type="radio"/> Access Token <input type="radio"/> Session <input type="radio"/> Device <input type="radio"/> User Device Authentication: Default User Authentication: Default
	<input type="button" value="Edit"/> <input type="button" value="Print"/> Version 1.0 Active	<input type="checkbox"/> Lock this version in	Security Test: Default App Authentication: <input checked="" type="radio"/> Access Token <input type="radio"/> Session <input type="radio"/> Device <input type="radio"/> User Device Authentication: Default User Authentication: Default

RestServiceStatus (RestServiceStatus)

Worklight Application Center

Development Team Provisioning

Enterprise App Provisioning
and Governance

App Feedback Management

Public App Stores



IBM Worklight Application - X

/fix/appcenterconsole/console.html

Welcome demo Sign out IBM

You are in: Applications

Application Management

Available Applications

Add Application

1 of 1 Page 1 Previous | Next

Sort by: Label... | OS | Update Date Filter: Show all applications

CardMatching
iOS
com.ibm.DemoString.dummyAppForPreview.CardMatching
Access control: unrestricted
1.0 (1.0) 34W16 ★★★★★ 0

Show: 5 | 10 | 20 | 50 | All rows

Jump to page: 1 of 1

Previous | Next

Version: DD-20131126-0830

© Copyright IBM Corporation 2011.. 2013.

2

Device Runtime

Cross-Platform
Compatibility Layer

Server Integration
Framework

Encrypted and
Syncable Storage

Runtime Skinning

Reporting for Statistics
and Diagnostics

3

Application Code







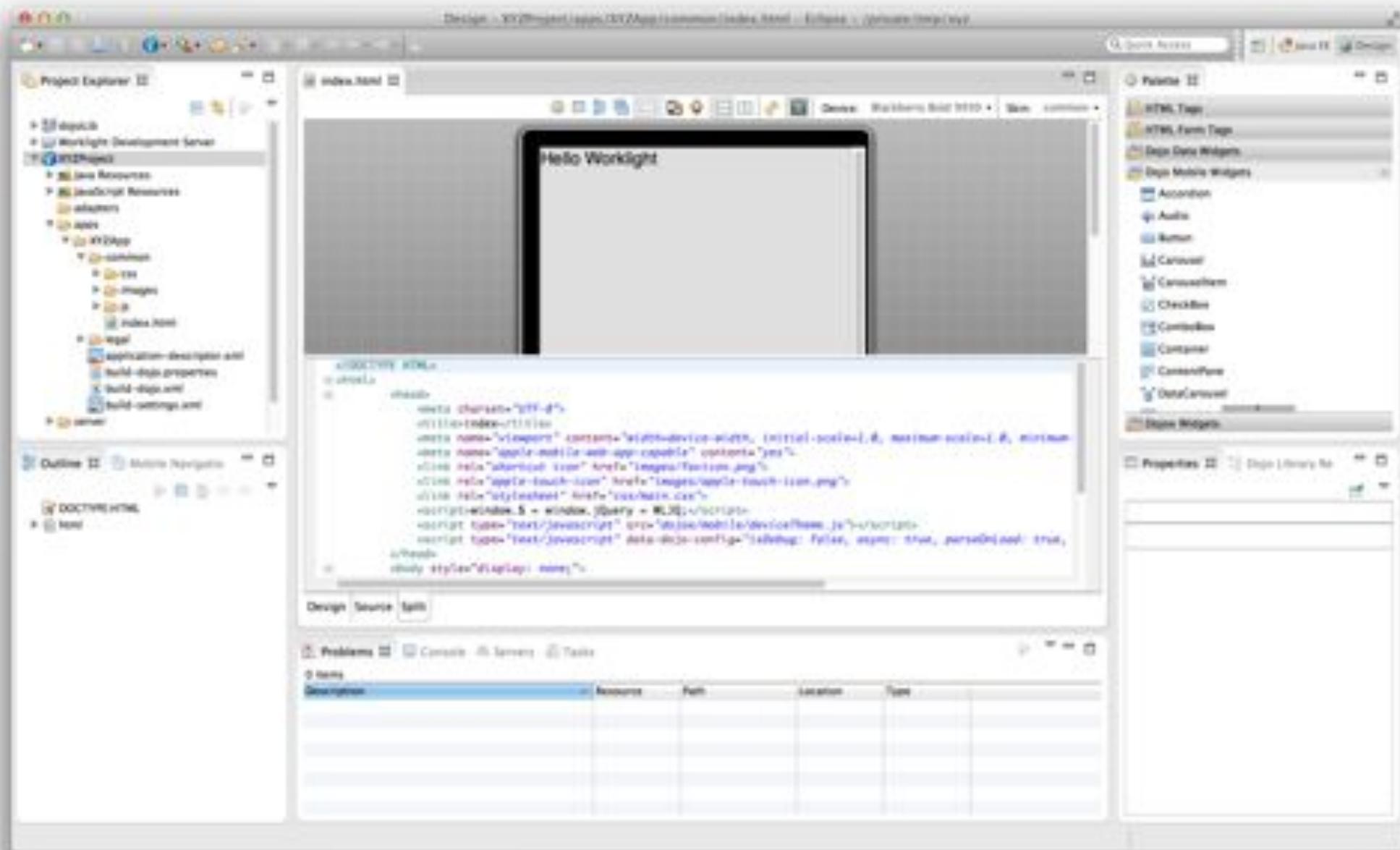
So how would we "normally" develop a Worklight app?



Develop App and
Adapters

... using the IDE





Use Platform-native
Tools to Build



ProjAppiPhone : iPhone - iOS 6.0

ProjAppiPhone : Ready | Today at 09:32

ProjAppiPhone.xcodeproj — CordovaLib.xcodeproj

Build Settings

Architectures

- Setting**: **CordovaLib**
- Debug**: **Any iOS Simulator SDK**, **Any iOS SDK**, **iPhoneOS6.1 SDK**
- Distribution**: **Any iOS Simulator SDK**, **Any iOS SDK**, **iPhoneOS6.1 SDK**
- Release**: **Any iOS Simulator SDK**, **Any iOS SDK**, **iPhoneOS6.1 SDK**

Build Options

- Setting**: **CordovaLib**
- Compiler for C/C++/Objective-C**: **Apple LLVM 5.0**

Deployment

- Setting**: **CordovaLib**
- Installation Build Products Location**: **/tmp/CordovaLib**
- Installation Directory**: **/var/local/198**
- Strip Install**: **Yes**
- Strip Debug Symbols During Copy**

 - Debug**: **No**
 - Distribution**: **Yes**
 - Release**: **Yes**

- String-Linked Product**: **None**
- Targeted Device Family**: **(iPhone/iPad)**
- iOS Deployment Target**: **iOS 5.0**

Packaging

- Setting**: **CordovaLib**
- Info.plist File**: **None**
- Product Name**: **Cordova**
- Public Headers Folder Path**: **include/Cordova**

Identity and Type

- Name**: **CordovaLib.xcodeproj**
- Location**: **Relative to Group**
- CordovaLib**
- CordovaLib.xcodeproj**
- Full Path**: **(temp/tempproj/ProjAppiPhone/ProjAppiPhone/CordovaLib/CordovaLib.xcodeproj)**

Project Document

- Project Format**: **Xcode 5.2-compatible**
- Organization**: **None**
- Class Prefix**: **None**

Test Settings

- Instrument Using**: **Spikes**
- Width**: **4**
- Height**: **8**
- # Wrap Lines**: **Yes**

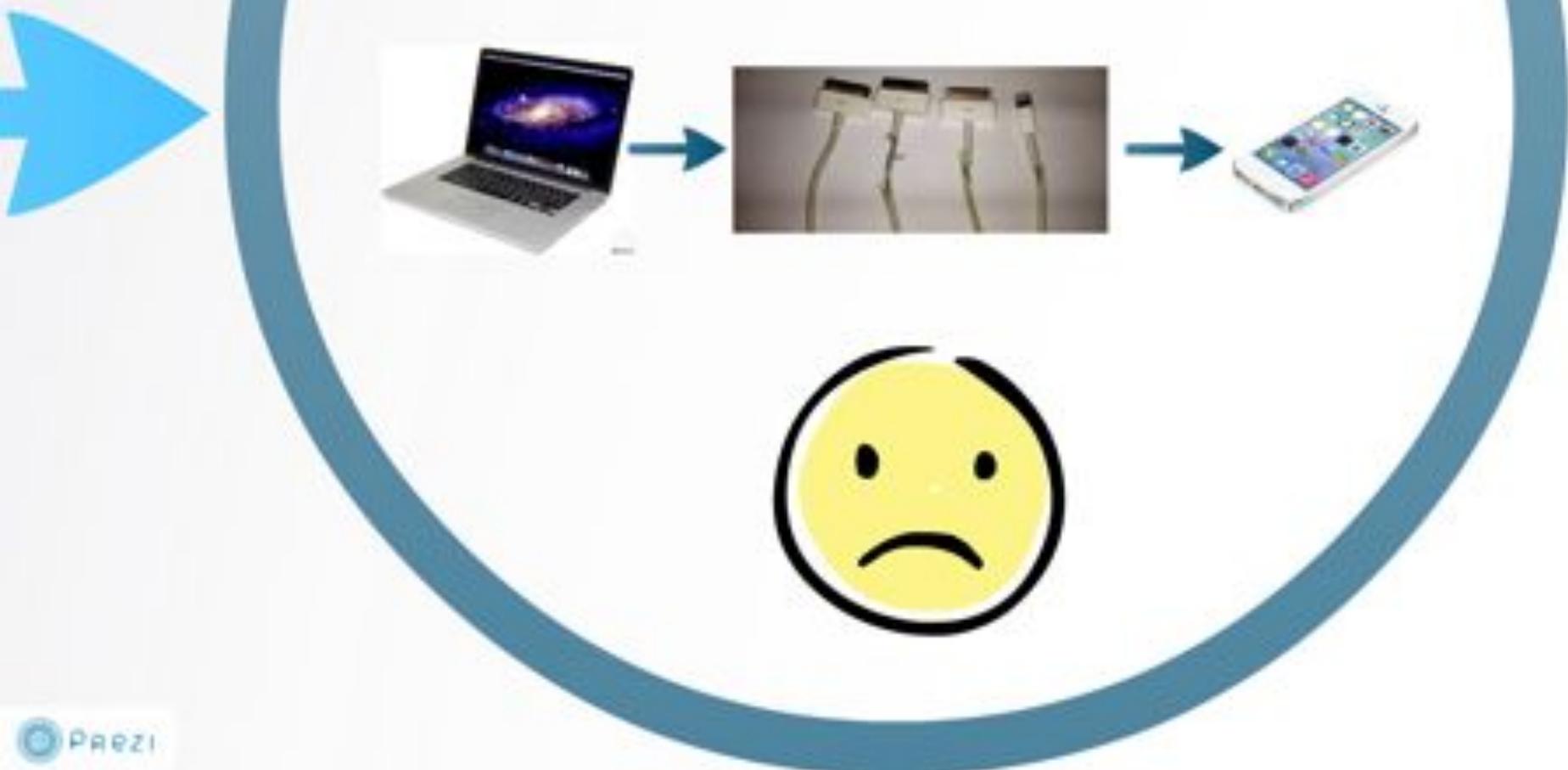
Source Control

- Repository**: **None**
- Type**: **None**
- Current Branch**: **None**
- Version**: **None**
- Status**: **No changes**
- Location**: **None**

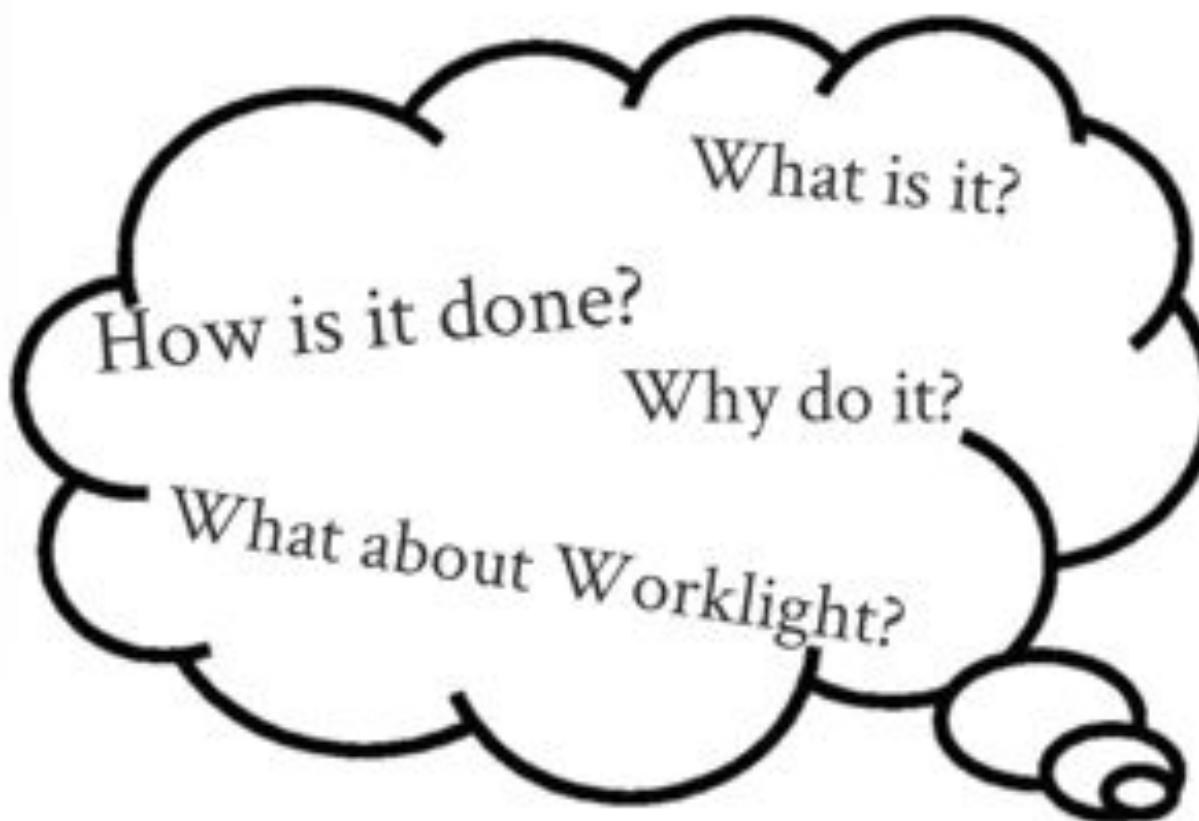
Objectives

- Objective-C class**: **An Objective-C class with a header for Cocoa Touch**
- Objective-C protocol**: **An Objective-C protocol for Cocoa Touch**
- Objective-C test case class**: **An Objective-C class implementing a unit test**

Deploy Artifacts
Manually for
Development and
Testing



Continuous Integration



Build in the morning have your fix on devices by the end of the day



Why would we do it? This is why:



Client transparency
Time to market
Encourages developers
Mitigates integration issues

Ideal Steps to Production



CI Environment

- Development:
 - built very frequently the latest level of code checked into SCM
 - possibly unstable, provides immediate feedback
- (Future Proofing)
- Bug Fixing - Once version 1.0 of the app is in app store and v1.1 is being created. Fixes can be made to the app in parallel with future development



CI provides immediate feedback to developers



app in parallel with future development

Jenkins

Build #3171 (Mar 21, 2016)

Build #3171 (Mar 21, 2016)

No changes

Started by anonymous user

Build #133 (Mar 19, 2016)

No changes

Started by anonymous user



Jenkins

Build #133 (Mar 19, 2016)

No changes

Started by anonymous user



CI provides immediate feedback to developers

[Back to Project](#)

[Status](#)

[Changes](#)

[Console Output](#)

[Edit Build Information](#)

[Delete Build](#)

[Polling Log](#)

[Build Graph](#)

[Previous Build](#)



Build #3171 (Mar 21, 2015)



Build Artifacts

- [Basket.adapter](#)
- [Category.adapter](#)
- [CommonUtilities.adapter](#)
- [Content.adapter](#)
- [Customer.adapter](#)
- [REDACTED] .war
- [Location.adapter](#)
- [Product.adapter](#)
- [RestServiceStubs-mobilewebapp-1.0.wlapp](#)
- [Reviews.adapter](#)
- [Search.adapter](#)
- [SecurityTestApp-mobilewebapp-1.0.wlapp](#)
- [Stock.adapter](#)



No changes.



Started by GitHub push by MarcoSergo Url Triggered
[app/commit/0258d63ace7dab7b46d7c5b43e5c23c](#)

Jenkins

Jenkins > SI-test1-Deploy > #133

 [Back to Project](#)

 [Status](#)

 [Changes](#)

 [Console Output \[raw\]](#)

 [Edit Build Information](#)

 [Delete Build](#)

 [Parameters](#)

 [Build Graph](#)

 [Previous Build](#)



Build #133 (Mar 19, 2



No changes.



Started by anonymous user

Continuous Integration server

Automated ways of building each component

What tools do we need?

A few bits of glue in the form of scripts etc.

Automated Testing?





*Liberty profile and Jenkins is a free,
open source and highly customisable
solution*



Continuous Integration server

Automated ways of building each component

What tools do we need?

Mature source-code management is vital e.g. git.

A few bits of glue in the form of scripts etc.

Automated Testing?



The Components of Worklight

- Worklight server-side components
 - .war - The Worklight server
 - .wlapp - The Worklight application
 - .adapter - The adapter

The screenshot shows the IBM Worklight Console interface. At the top, there are three tabs: Catalog, Devices, and Push Notifications. Below the tabs, a blue header bar contains the text "Deploy application or adapter" and a "Choose file" button with the message "No file chosen". To the right of the button is a "Submit" button. Underneath the header, the "Catalog" tab is active, showing a list of deployed applications. The first item in the list is "CommonUtilities", which has a blue plug icon. To its right, the text "Last deployed at 2014-03-20 16:58" and a "Show details" link are displayed. Further down the list, another entry for "CommonUtilities" is shown with a similar icon and deployment details. The bottom right corner of the interface has a checkbox labeled "Lock this version".

Worklight supplies Ant Tasks to build & deploy its components

Continuous Integration server

Automated ways of building each component

What tools do we need?

Mature source-code management is vital e.g. git.

A few bits of glue in the form of scripts etc.

Automated Testing?



Powerful Source-code Management

- GitHub Pull requests or similar features enable easy code peer review prior to integration



- GitHub Webhooks can trigger builds on code commit allowing CI to "blame" code mergers that break the build

Powerful SCM can boost development time and simplify integration

All Requests

1

Open Closed

Sort: Newest ▾

New pull request

Yours

Cleanup whitespace.

I would like to clean up the whitespace in worklight.properties, please...

#171

by andrewt23 3 minutes ago feature/cleanup-whitespace

Find a user...

andrewt23

1

Keyboard shortcuts available ▾

Cleanup whitespace. #171

Open

New pull request

Conversation 0 Comments 0 File Changes 0

Show Diff Stats

Showing 1 changed file with 7 additions and 7 deletions.

File	Line	Change Type	Text
worklightServer/conf/worklight.properties	18	Add	# In worklight.properties you defined:
	19	Remove	#
	20	Remove	#
	21	Remove	# -- My adapter .db .name=jdbc/Wkldgtw123
	22	Remove	#
	23	Add	# In adpater .db .
	24	Remove	#
	25	Remove	# <adapters>
	26	Add	DB <DB> <DB> <DB>
	27	Remove	#<adapters> <DB> <DB> <DB> <DB> <DB> <DB> <DB>
	28	Remove	#<adapters> <DB>
	29	Add	# Such properties will be exposed as JNDI resources for the project configuration XML file.

Powerful Source-code Management

- GitHub Pull requests or similar features enable easy code peer review prior to integration



- GitHub Webhooks can trigger builds on code commit allowing CI to "blame" code mergers that break the build

Powerful SCM can boost development time and simplify integration

Continuous Integration server

Automated ways of building each component

What tools do we need?

Mature source-code management is vital e.g. git.

A few bits of glue in the form of scripts etc.

Automated Testing?





The glue

- Scripts and Ant tasks to
 - Source Code Checkout
 - Build Worklight artefacts
 - Deploy .wlapp & .adapters
 - Build native applications (.ipa, etc)
 - Execute tests
 - Deploy Apps to app center
- CI server to drive & manage these



Ideal Steps to Production



SI Environment

Systems Integration:

- More stable code base
- Promoted at end of Sprints
- End to end process tests
- QA and UI testing can be expanded
- Integration tests
- Another App Center

UAT Environment

User Acceptance Testing

- The final check before code is pushed to production
- Business sign off
- Final QA
- "Production like" environment (clustered, load balanced etc)

Ideal Steps to Production

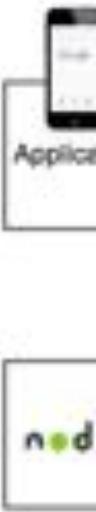


Production ready app in App Stores

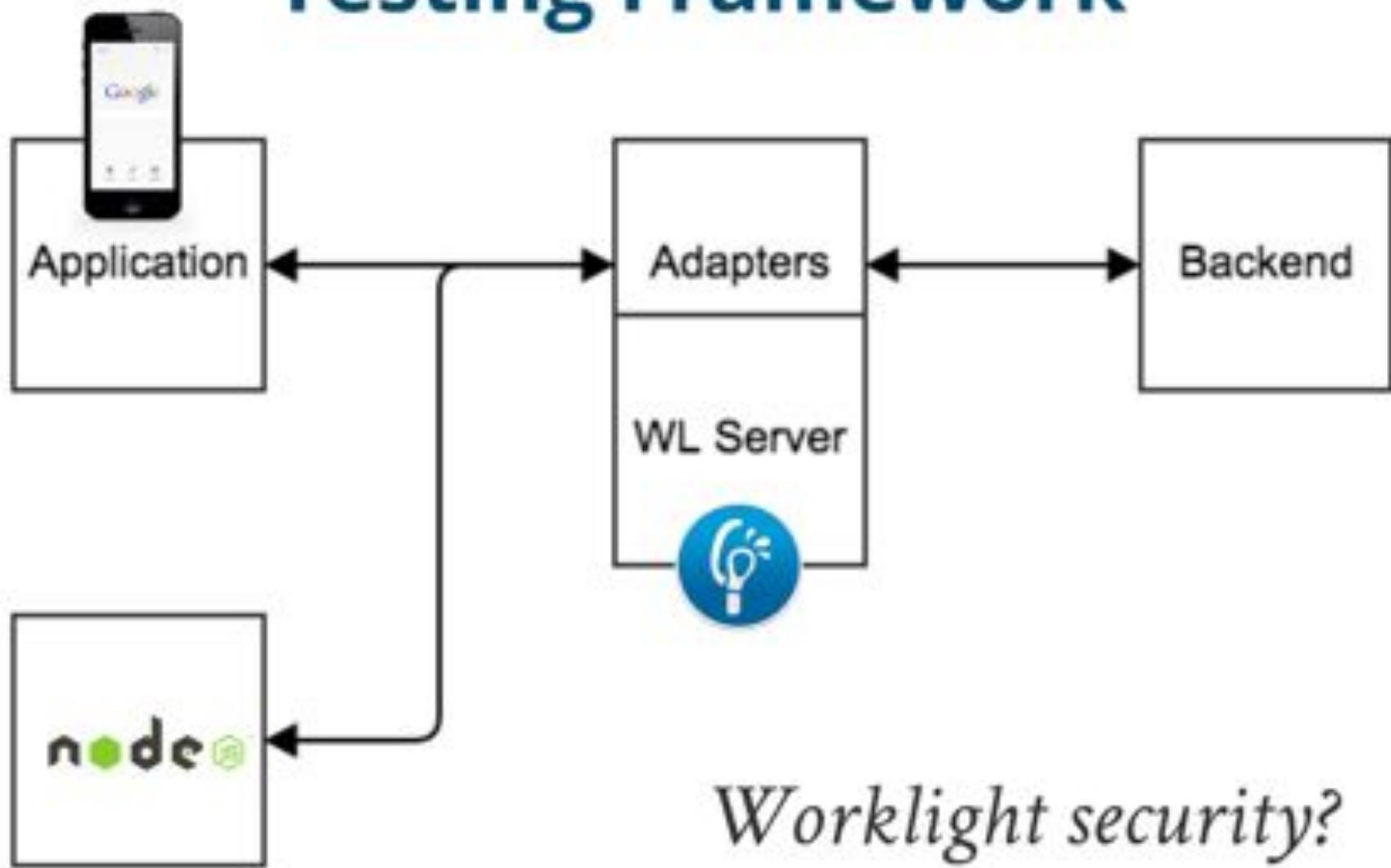


Automated Adapter Testing

- Testing server-side components with well-defined inputs and outputs.
- Provides feedback on code quality and data manipulation
- Integration testing of adapters supported
- Integrated into CI - this gives a powerful metric
- We have successfully combined open-source components with the Worklight API to construct such a framework.



Testing Framework



Automated Adapter Testing

- Testing server-side components with well-defined inputs and outputs.
- Provides feedback on code quality and data manipulation
- Integration testing of adapters supported
- Integrated into CI - this gives a powerful metric
- We have successfully combined open-source components with the Worklight API to construct such a framework.



What tools do we need?



simple, flexible, fun

- Customised Worklight node script to invoke the adapters
- Stubbing solution - Use another Worklight Mobile Web Project
 - Worklight can serve up text in response to GET requests



```

1 var ip = require('wlAdapter');
2 var expect = require('chai').expect;
3
4 var param = "{}";
5
6 describe("Category Adapter Test Suite", function(){
7   describe ("#getTopLevelCategories()", function(){
8     it("should have data field containing something", function(done){
9       ip.invokeProcedure("Category", "getTopLevelCategories", param, function(response){
10         expect(response.data).not.to.be.null;
11       },done); // end callback
12     }); // end it
13   }); // end describe
14}); // end describe

```

```

+ ----- git:(master) > mocha -R spec tests/getTopLevelCategories.spec.js

Category Adapter Test Suite
  #getTopLevelCategories()
    ✓ should have data field containing something (0ms)

  1 failing (10ms)
+ ----- git:(master) > mocha -R spec tests/getTopLevelCategories.spec.js

Category Adapter Test Suite
  #getTopLevelCategories()
    ✘ should not have data field

  0 passing (10ms)
  1 failing

  1) Category Adapter Test Suite #getTopLevelCategories() should not have data field:
     Error: expected 0 object but got Search, ... (1 ms)
      at Object.exports.assert [as assert] (C:\Users\Mark.Gamble\Documents\Node.js\app\Homelander\lib\utils\assert.js:12:11)
      at process._tickCallback (internal/process/next_tick.js:64:9)

```

```
1 var ip = require('wlAdapter');
2 var expect = require('chai').expect;
3
4 var param = "[]";
5
6 describe("Category Adapter Test Suite", function(){
7   describe ("#getTopLevelCategories()", function(){
8     it("should have data field containing something", function(done){
9       ip.invokeProcedure("Category", "getTopLevelCategories", param, function(response){
10         expect(response.data).not.to.be.null;
11         },done); // end callback
12       }); // end it
13     }); // end describe
14   }); // end describe
```

```
- unitTest git:(develop) ✘ mocha -R spec tests/getTopLevelCategories.test.demo.js
```

```
Category Adapter Test Suite
  #getTopLevelCategories()
    should have data field containing something (10ms)
```

```
+ UnitTest git:(develop) ✘ mocha -R spec tests/getTopLevelCategories.test.demo.js

Category Adapter Test Suite
  #getTopLevelCategories()
    ✓ should have data field containing something (105ms)

1 passing (100ms)
+ UnitTest git:(develop) ✘ mocha -R spec tests/getTopLevelCategories.test.demo.js

Category Adapter Test Suite
  #getTopLevelCategories()
    1) should not have data field

0 passing (123ms)
1 failing

1) Category Adapter Test Suite #getTopLevelCategories() should not have data field:
   Error: expected { Object (statusCode, Search, ...) } to be null
    at reportTestFailure (/Users/Marland/git/co.uk.homebase.strategic-app/Homebase/adapters/UnitTest/node_modules/v1Adapter/invokeWProcedure.js:238:13)
    at /Users/Marland/git/co.uk.homebase.strategic-app/Homebase/adapters/UnitTest/node_modules/v1Adapter/invokeWProcedure.js:35:7
    at tryCatchReject (/Users/Marland/git/co.uk.homebase.strategic-app/Homebase/adapters/UnitTest/node_modules/when/Lib/makePromise.js:862:54)
    at Handler.RejectedHandler.when (/Users/Marland/git/co.uk.homebase.strategic-app/Homebase/adapters/UnitTest/node_modules/when/Lib/makePromise.js:554:7)
    at Handler.DeferredHandler.run (/Users/Marland/git/co.uk.homebase.strategic-app/Homebase/adapters/UnitTest/node_modules/when/Lib/makePromise.js:384:13)
    at Scheduler._drainQueue (/Users/Marland/git/co.uk.homebase.strategic-app/Homebase/adapters/UnitTest/node_modules/when/Lib/scheduler.js:42:14)
    at Scheduler.drainQueue (/Users/Marland/git/co.uk.homebase.strategic-app/Homebase/adapters/UnitTest/node_modules/when/Lib/scheduler.js:19:91)
    at process._tickCallback (node.js:415:13)

+ UnitTest git:(develop) ✘
```

```
→ UnitTest git:(develop) ✘ mocha -R spec tests/getTopLevelCategories.test.demo.js
```

Category Adapter Test Suite

```
#getTopLevelCategories()  
  ✓ should have data field containing something (105ms)
```

```
1 passing (109ms)
```

```
→ UnitTest git:(develop) ✘ mocha -R spec tests/getTopLevelCategories.test.demo.js
```

Category Adapter Test Suite

```
#getTopLevelCategories()  
  1) should not have data field
```

```
0 passing (123ms)
```

```
1 failing
```

```
PAGE 1 Category Adapter Test Suite #getTopLevelCategories() should not have data field:
```

```
1 passing (109ms)
```

```
+ UnitTest git:(develop) ✘ mocha -R spec tests/getTopLevelCategories.test.demo.js
```

Category Adapter Test Suite

```
#getTopLevelCategories()  
  1) should not have data field
```

```
0 passing (123ms)
```

```
1 failing
```

```
1) Category Adapter Test Suite #getTopLevelCategories() should not have data field:
```

```
Error: expected { Object (statusCode, Search, ...) } to be null
```

```
at reportTestFailure (/Users/Marland/git/co.uk.homebase.strategic-app/Homebase/adapters  
at /Users/Marland/git/co.uk.homebase.strategic-app/Homebase/adapters/UnitTest/node_modu  
at tryCatchReject (/Users/Marland/git/co.uk.homebase.strategic-app/Homebase/adapters/U  
at Handler.RejectedHandler.when (/Users/Marland/git/co.uk.homebase.strategic-app/Homeba  
at Handler.DeferredHandler.run (/Users/Marland/git/co.uk.homebase.strategic-app/Homebas  
at Scheduler._drainQueue (/Users/Marland/git/co.uk.homebase.strategic-app/Homebase/adap  
at Scheduler.drainQueue (/Users/Marland/git/co.uk.homebase.strategic-app/Homebase/adapt  
at process._tickCallback (node.js:415:13)
```

```
+ UnitTest git:(develop) ✘ └
```



Jenkins can read test results

Jenkins

Jenkins → CI-Develop-dev1 → #3124 → Test Results

[Back to Project](#)

[Status](#)

[Changes](#)

[Console Output \(raw\)](#)

[Edit Build Information](#)

[History](#)

[Failures](#)

[Test Results](#)

Test Result

111 failures (4111)

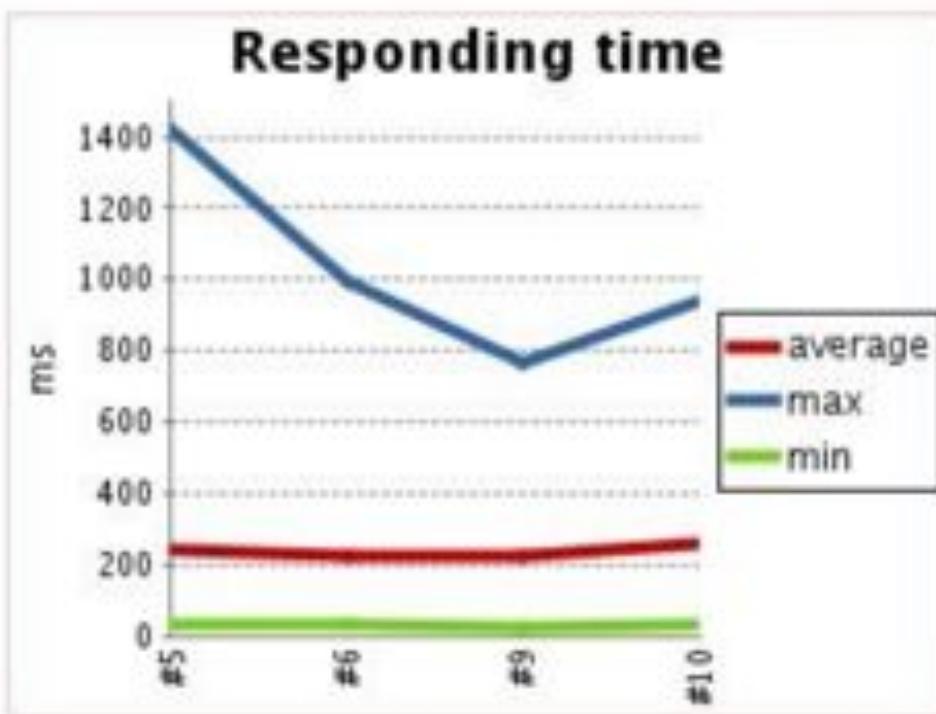
All Failed Tests

Test Name

↳ Consent.Adapter.Test.Suite.#setContent() should work if statusCode is 200 - UAT1.version.should.work if statusCode is 200

Performance Testing

- Performance testing the adapter layer against stubs then against real services.
- Provides instant feedback on performance regressions and improvements



Food for thought...

- How do you handle PUT, POST, and DELETE?
- What do you do with tests that connect to both stubbed data and real services?
- How are unit tests different from integration tests?

Preparing for Production

(or, a few gotchas...)

- Worklight Applications have good management for versioning
- However, currently Worklight Adapters **do not**
- Therefore, you should do it **in code** before you reach v1, to prepare yourself for v2:

```
function getNewVersion() {
    var adapter = WL.Client.createAdapter();
    adapter.getDefinition(function(err, definition) {
        if (err) {
            WL.Logger.error("Error getting adapter definition: " + err);
            return;
        }
        var newVersion = definition.version + 1;
        adapter.setVersion(newVersion, function(err) {
            if (err) {
                WL.Logger.error("Error setting adapter version: " + err);
            } else {
                WL.Logger.info("Adapter version updated to: " + newVersion);
            }
        });
    });
}
```

Understand Updates

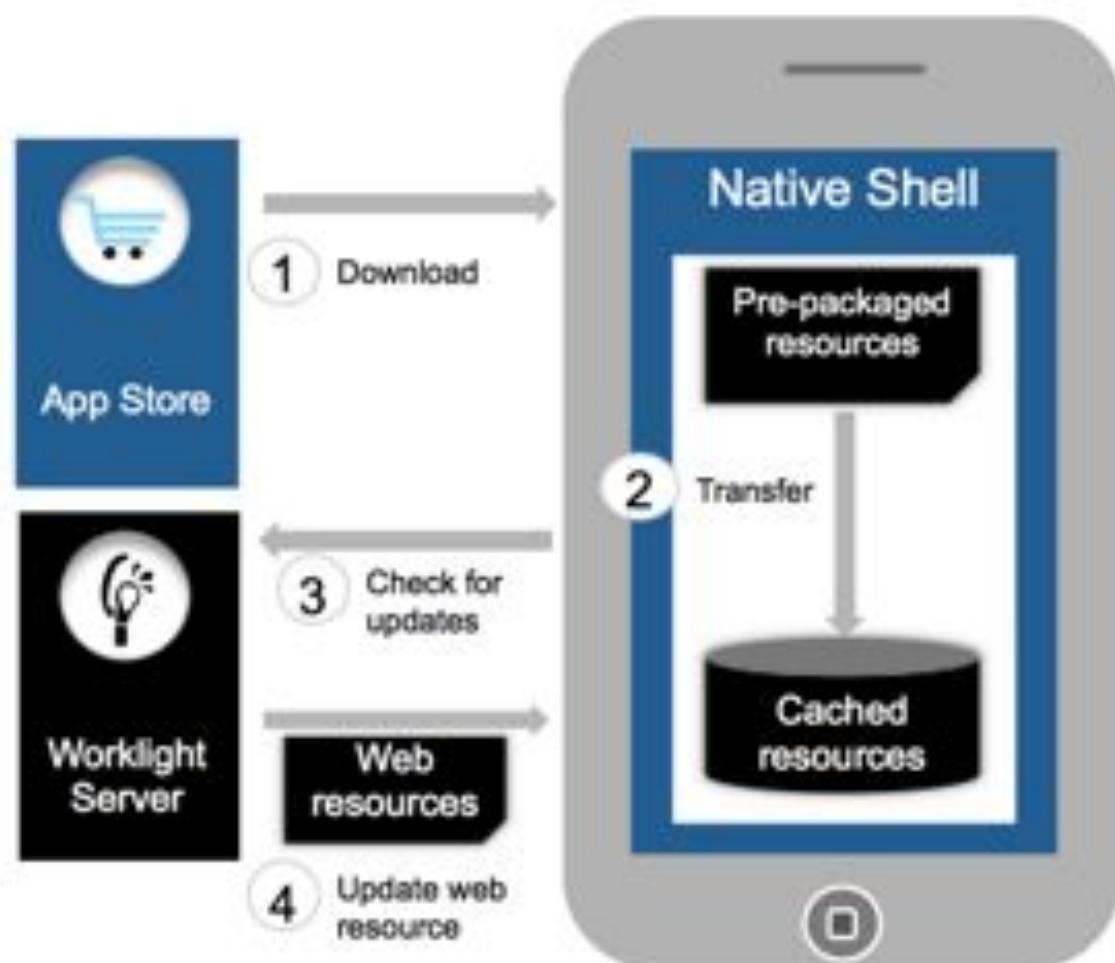
Two ways of updating:

1. Update web code only - redeploy .wlapp only

Implicitly encourages "Direct Update"

Suitable primarily for B2E scenarios





Understand Updates

2.
 - Update web code and (custom) native code
 - Re-release via consumer App Store



- More 'heavyweight', suitable for all scenarios

Preparing for Production

(or, a few gotchas...)

- Worklight Applications have good management for versioning
- However, currently Worklight Adapters **do not**
- Therefore, you should do it **in code** before you reach v1, to prepare yourself for v2:

```
WL.Client.invokeProcedure(  
    adapter: "Adapter1",  
    procedure: "getStarters",  
    parameters: [1, "someOtherData"],  
    onFailure: function(response) {  
        console.log("An error occurred: " + response.error.message);  
    }  
);
```



```
WL.Client.invokeProcedure({  
    adapter: "Adapter1",  
    procedure: "getStories",  
    parameters: [1, "someOtherData"],  
});|
```

```
function getStories(param) {
    if (version === 1) {
        path = getPath(param);

        var input = {
            method : 'get',
            returnedContentType : 'xml',
            path : path,
            transformation : {
                type : 'xslFile',
                xslFile : 'filtered.xsl'
            }
        };

        return WL.Server.invokeHttp(input);
    } else if (version === 2) {
        // ... this code is yet to be written

        // incidentally, it could be factored out into another adapter:

        // var invocationData = {
        //     "adapter" : "StoryV2Adapter",
        //     "procedure" : "getStories",
        //     "parameters" : []
        // };

        // WL.Server.invokeProcedure(invocationData);
    }
}
```

Managing Authentication

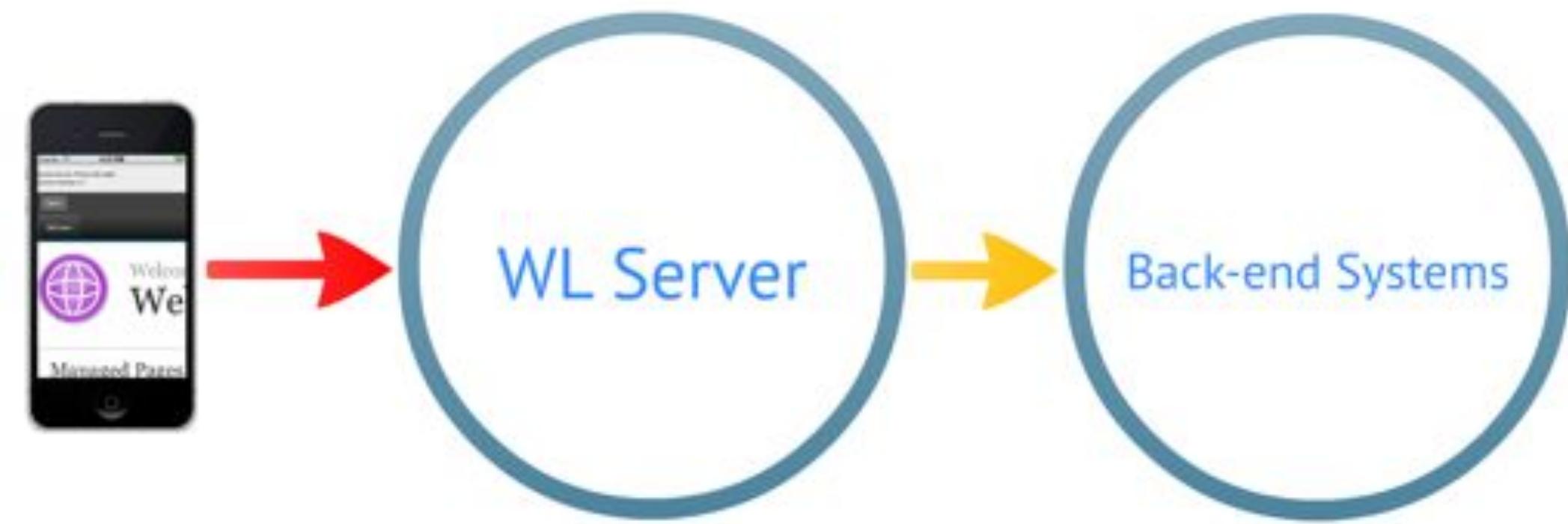
- Exploiting standard WL adapter framework outsources authentication to the WL Server 
- App uses WL Challenge Handler to manage client-side login lifecycle; independent of the type of authentication performed.

```
var challengeHandler = WL.Client.createChallengeHandler("WCSAuthenticationRealm");

challengeHandler.isCustomResponse = function(response) {
    // ...
};

challengeHandler.handleChallenge = function(response) {
    var authRequired = response.responseJSON.authStatus;

    if(authRequired === "required") {
        // ... Show our Login page
    } else if(authRequired === "complete") {
        // ... Hide our Login page; user has successfully
        // Logged in.
    }
};
```



Managing Authentication

- Exploiting standard WL adapter framework outsources authentication to the WL Server 
- App uses WL Challenge Handler to manage client-side login lifecycle; independent of the type of authentication performed.

```
var challengeHandler = WL.Client.createChallengeHandler("WCSAuthenticationRealm");

challengeHandler.isCustomResponse = function(response) {
    // ...
};

challengeHandler.handleChallenge = function(response) {
    var authRequired = response.responseJSON.authStatus;

    if(authRequired === "required") {
        // .. Show our login page
    } else if(authRequired === "complete") {
        // .. Hide our login page; user has successfully
        // logged in.
    }
};
```

in lifecycle; independent of the type of authentication performed.

```
var challengeHandler = WL.Client.createChallengeHandler("WCSAuthenticationRealm");

challengeHandler.isCustomResponse = function(response) {
    // ...
};

challengeHandler.handleChallenge = function(response) {
    var authRequired = response.responseJSON.authStatus;

    if(authRequired === "required") {
        // .. Show our login page
    } else if(authRequired === "complete") {
        // .. Hide our login page; user has successfully
        // Logged in.
    }
};
```

How is Server-Side Authentication Managed?

- WL ships with some standard server-side authentication mechanisms (LDAP, LTPA, etc..)
- Many real-world authentication scenarios require more sophisticated authentication
- Worklight provides **Adapter Authentication** for this purpose.

"Custom" Adapter Authentication

- An Authentication Adapter is a specialised HTTP adapter which integrates with the "back-end" login services, etc.

```
function login(username, password) {  
    var input = {  
        method : 'get',  
        returnedContentType : 'json',  
        path : '/customer/login?login=' + username + '&password=' + password + '&key=' + apitkey,  
    };  
  
    WL.Server.setActiveUser("worklightrealm", { user: username });  
  
    return result;  
}
```

- This is often the most complex part of a Worklight implementation, and should be planned and PoCed early.

HTTP adapter which integrates "back-end" login services, etc.

```
function login(username, password) {
    var input = {
        method : 'get',
        returnedContentType : 'json',
        path : "/customer/login?loginId=" + username + "&password=" + password + "&" + apiKey,
    };
    WL.Server.setActiveUser("WorklightRealm", { user: username });
    return result;
}
```

- This is often the most complex

Worklight implementation.

"Custom" Adapter Authentication

- An Authentication Adapter is a specialised HTTP adapter which integrates with the "back-end" login services, etc.

```
function login(username, password) {
    var input = {
        method : 'get',
        returnedContentType : 'json',
        path : "/customer/login?loginId=" + username + "&password=" + password + "&t=" + spkKey,
    };
    WL.Server.setActiveSession("worklightrealm", { user: username });
    return results;
}
```

- This is often the most complex part of a Worklight implementation, and should be planned and PoCed early.

Conclusions

- Worklight is ready for production application construction now - we are doing it with real customers.
- Continuous Integration and Continuous Testing are very powerful for driving up quality and driving down time-to-market even with enterprise complexities.

Agile Mobile Apps for the Enterprise with IBM Worklight

See how would one "normally" develop a Worklight app?



conclusions

- We can take advantage of the existing infrastructure
- We can reuse existing code
- We can reuse existing components
- We can reuse existing databases
- We can reuse existing security
- We can reuse existing business logic



Automated Adapter Testing

- Create test cases for the adapter
- Run the test cases against the adapter
- Identify errors in the adapter
- Fix the errors in the adapter

