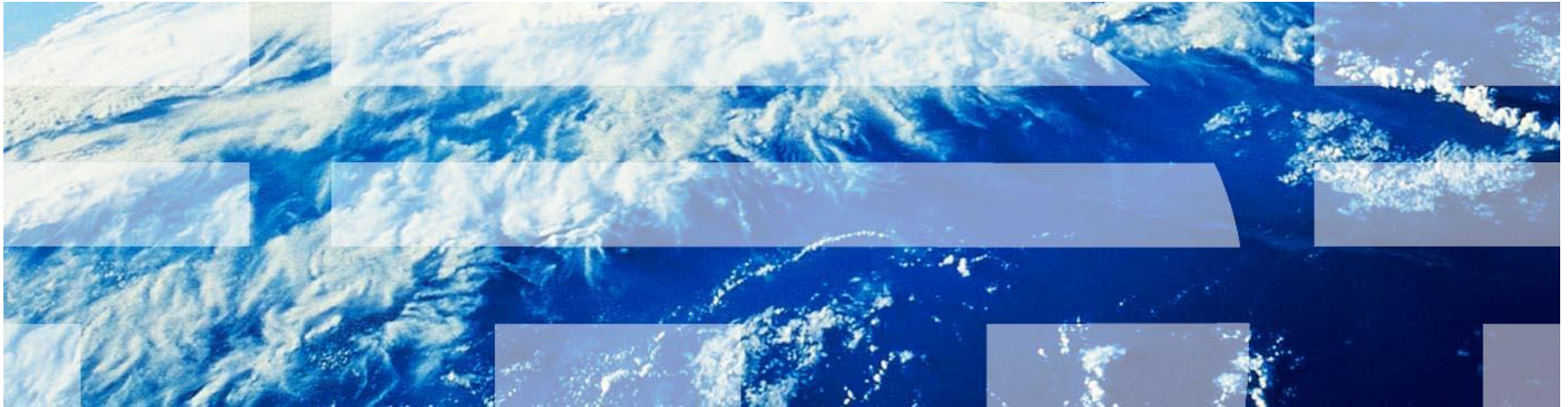

Introduction to Worklight Integration



Agenda

- **IBM Mobile Foundation**
- **Introduction to Worklight**
- **How to Integrate – Worklight Adapters**
- **WebAPI**
- **HTTP & SOAP**
- **Database (SQL)**
- **WebSphere Message Broker**
- **Cast Iron**

IBM Mobile Foundation

A graphic for IBM Mobile Foundation V5.0. It features a white cloud with the text 'IBM Mobile Foundation V5.0' inside. Below the cloud is a row of server racks on the left and a collection of various mobile devices (smartphones and tablets) on the right. The background is a blue gradient with a faint world map and binary code.

IBM Mobile Foundation V5.0

Build, connect, manage and secure
your mobile enterprise

Includes:

- IBM Worklight V5.0
- IBM WebSphere Cast Iron
- IBM Endpoint Manager for Mobile Devices

Plus New Services Offering:

- IBM Software Services for Mobile Foundation

Agenda

- **IBM Mobile Foundation**
- **Introduction to Worklight**
- How to Integrate – Worklight Adapters
- WebAPI
- HTTP & SOAP
- Database (SQL)
- WebSphere Message Broker
- Cast Iron

The Worklight Mobile Platform

The Worklight Mobile Platform is an open, complete and advanced mobile application platform for HTML5, hybrid and native apps.



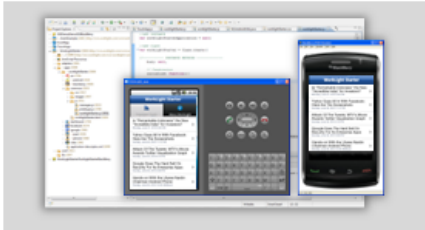
Evolving Mobile Landscape

Mobile Development is more expensive than traditional Web App Development:

- Which smartphone? Which tablet? Which form factor?
 - iOS, Android, Blackberry, Windows Phone
 - All of the above..
- Skills?
 - Web or native apps? Java or Objective C? Or other?
- Maintenance?
 - Separate software stacks for each major OS
 - Separate applications for each major OS
 - How do I keep software current?
- Security?
 - Encryption? Authentication?
 - Response to stolen/lost devices?
- Management?
 - Can I see my apps? Can I disable them remotely?
- Enterprise Integration?
 - How do I build cross-channel app?

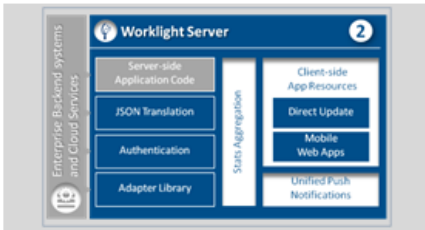
Worklight

Advanced mobile application platform for smartphones and tablets



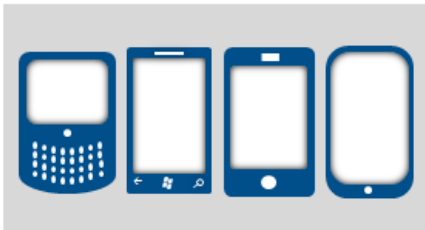
Worklight Studio

The most complete, extensible environment with maximum code reuse and per-device optimization



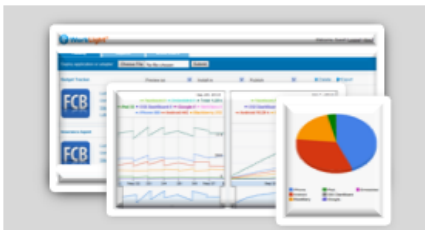
Worklight Server

Unified notifications, runtime skinning, version management, security, integration and delivery



Worklight Runtime Components

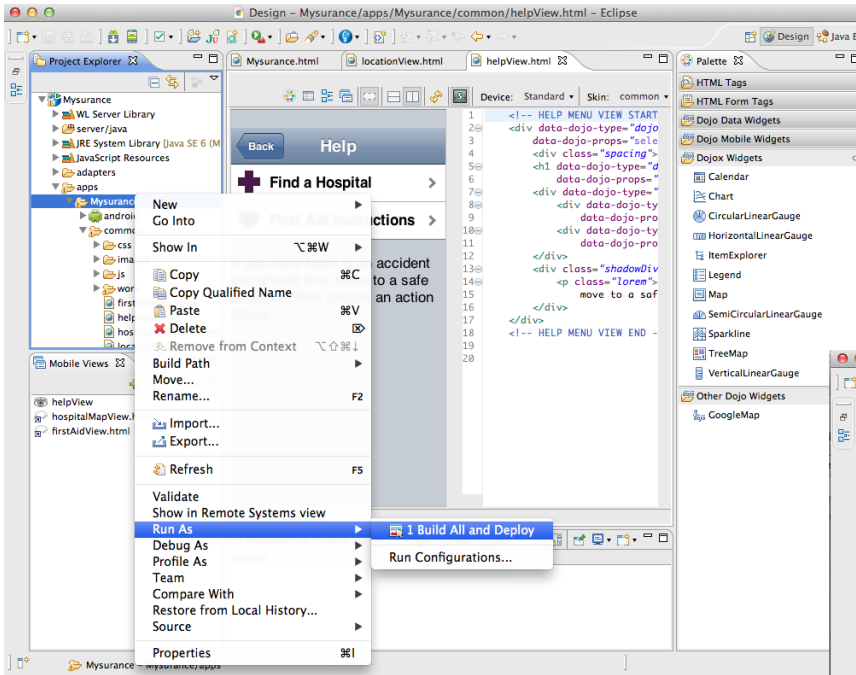
Extensive libraries and client APIs that expose and interface with native device functionality



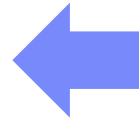
Worklight Console

A web-based console for real-time analytics and control of your mobile apps and infrastructure

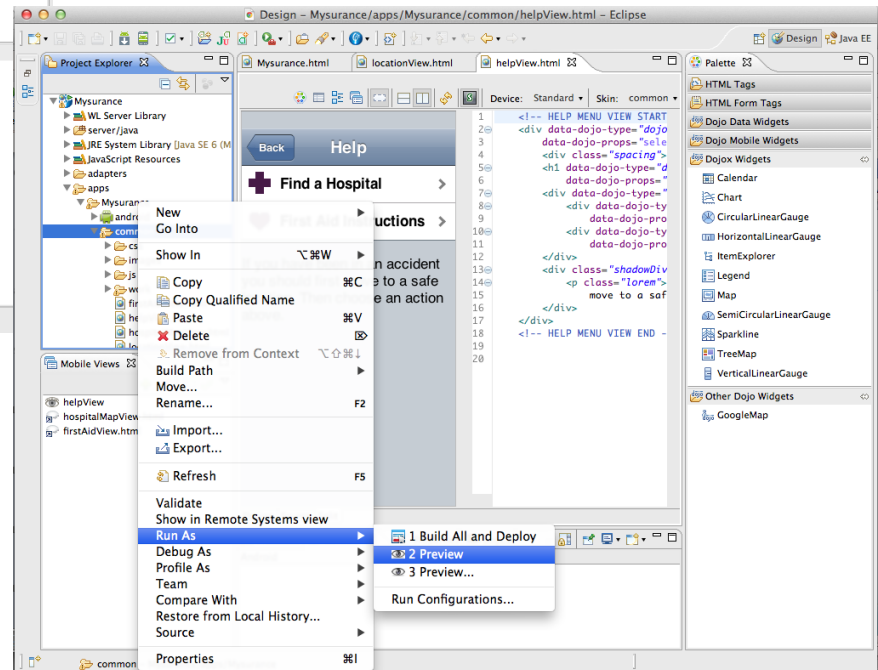
Worklight Studio – build, package and test applications



Build and Deploy
 Package the Worklight Application
 and deploy to Studio's lightweight
 web container



Preview
 Launch the Mobile Browser
 Simulator for device-specific
 rendering and testing



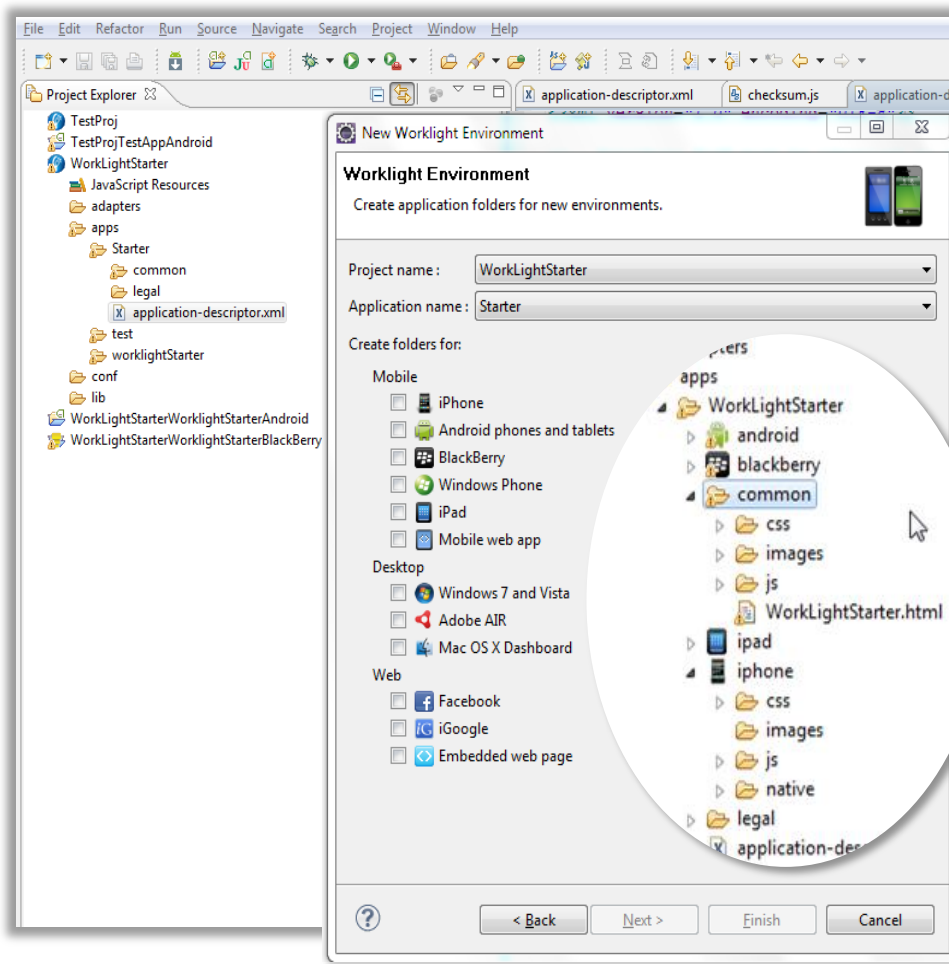
Worklight Studio – cross-platform code base

Common elements stored in the common folder

- HTML, JavaScript, CSS
- Worklight API usage
- 3rd party tools usage

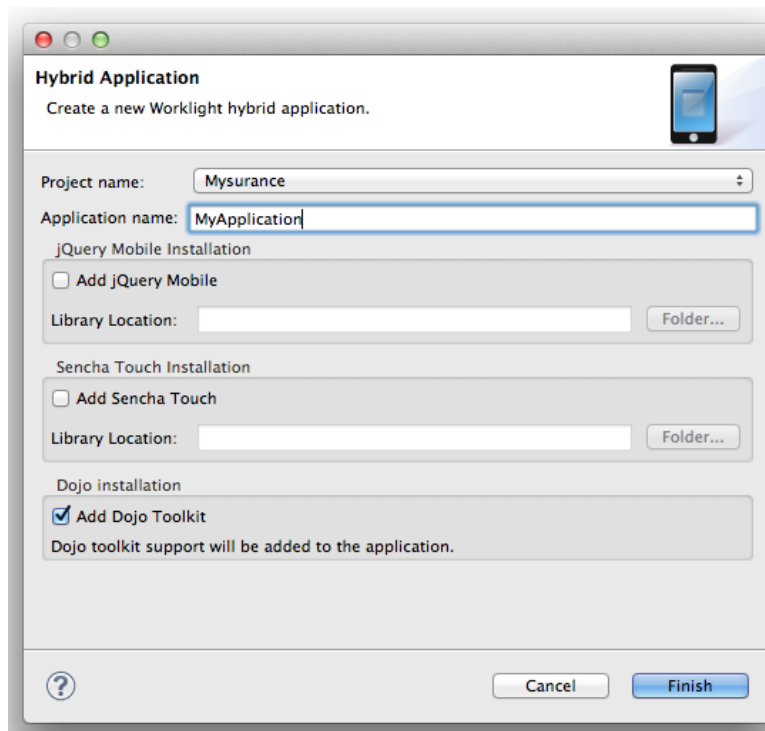
Named Environment directories (android, blackberry, iphone, etc...)

- Generated as Environments are added to the project
- Platform-specific code
- Environment-specific user optimizations
 - Over-ride default behaviors & styles
 - Behavior and branding changes



Worklight Studio – seamless integration with 3rd-party mobile tools

- Embedded Apache Cordova
 - Provides access to device features from a JavaScript API (formerly known as Phonegap)
- Optional Third Party tools
 - Simplified mobile programming models
 - Optimized mobile experience
 - Device-specific styling options
- Freedom of choice
 - Consistency with existing tool choices
 - Maximize existing skills
 - Options: choose none, one, multiple or bring-your-own



About pages, views and fragments

- Views are the mobile equivalent of html pages

```

45 <body onload="WL.Client.init({ timeout: 2000, onConnectionFailure: doConnect
46
47 <!-- MAIN VIEW START -->
48 <div data-dojo-type="dojox.mobile.ScrollableView" id="mainView">
81 <!-- MAIN VIEW END -->
82
83 <!-- PROFILE VIEW START -->
84 <div data-dojo-type="dojox.mobile.ScrollableView" id="profileView" data-dojo
128 <!-- PROFILE VIEW END -->
129
130 <!-- ACCIDENT TOOLBOX MENU VIEW START -->
131 <div data-dojo-type="dojox.mobile.ScrollableView" id="toolboxView">
146 <!-- ACCIDENT TOOLBOX MENU VIEW END -->
147
148 <!-- ACCIDENT LOCATION VIEW START -->
149 <div id="locationView" data-dojo-type="dojox.mobile.ScrollableView" data-do
241 <!-- ACCIDENT LOCATION VIEW END -->
242
243 <!-- EXCHANGE DRIVER INFO VIEW START -->
244 <div id="otherDriverView" data-dojo-type="dojox.mobile.ScrollableView" data
268 <!-- EXCHANGE DRIVER INFO VIEW END -->
269
270 <!-- ACCIDENT PHOTOS VIEW START -->
271 <div id="photoView" data-dojo-type="dojox.mobile.ScrollableView" data-dojo
313 <!-- ACCIDENT PHOTOS VIEW END -->
314
315 <!-- HELP MENU VIEW START -->
316 <div data-dojo-type="dojox.mobile.ScrollableView" id="helpView">
332 <!-- HELP MENU VIEW END -->
333
334 <!-- HOSPITAL MAP VIEW START -->
335 <div id="hospitalMapView" data-dojo-type="dojox.mobile.View"
336 style="width: 100%; height: 100%;" data-dojo-props="selected:false">
337 <div class="spacing"></div>
  
```

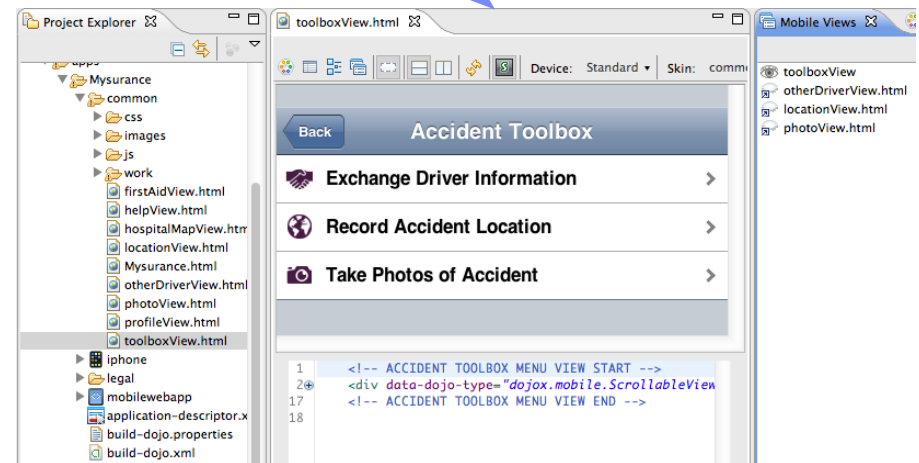
All views in single html file

- One html file
- Multiple views
- Mobile Views shows all views in the current html file

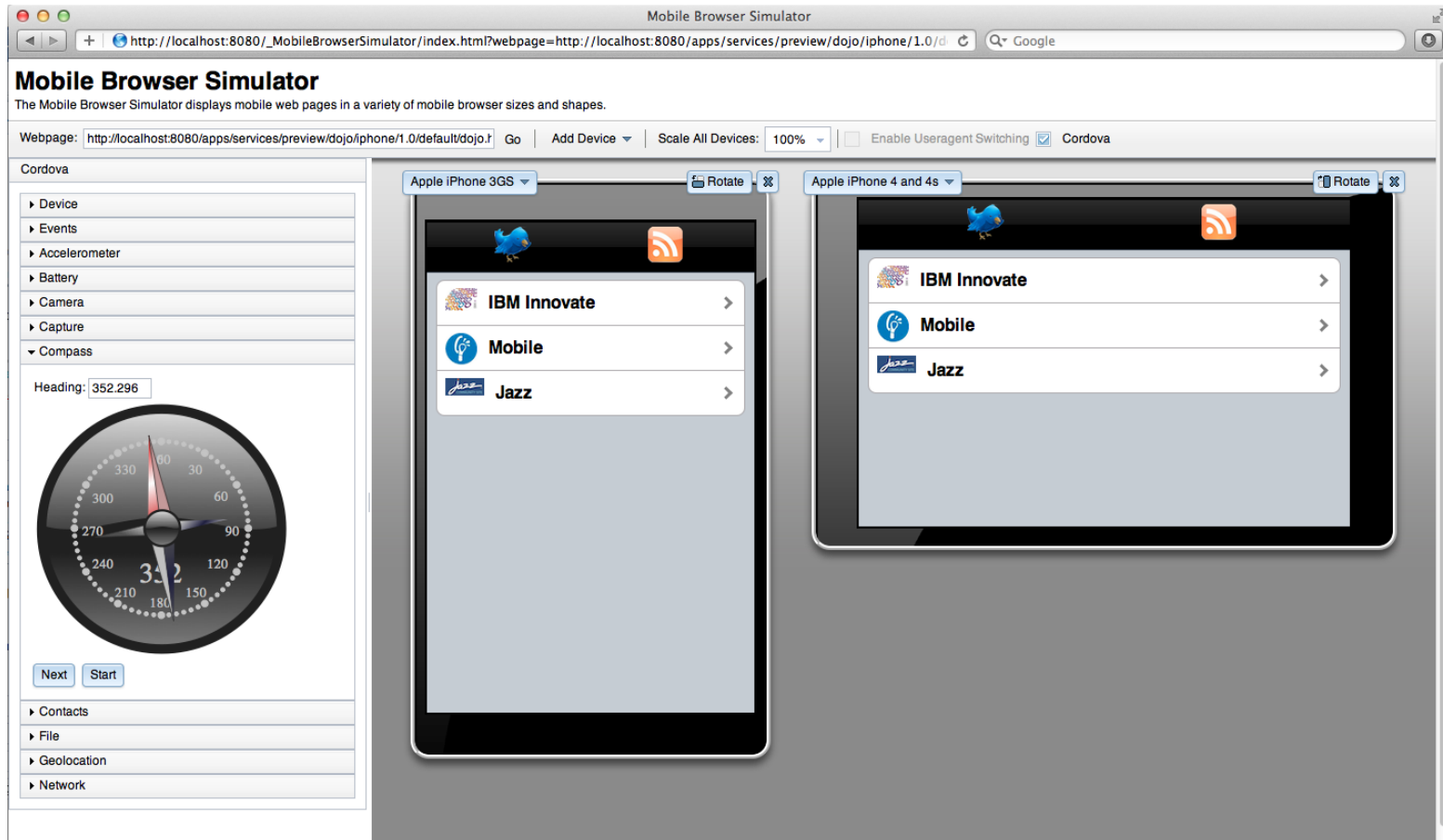
Views in individual html fragments

- Many html files (fragments)
- One view per file
- Mobile Views tab shows all views linked from current html fragment

Best Practice – modularity, MVC-friendly organization, enabling multiple authors, sharing of common views between applications, smaller files for parsing & rendering.



Worklight Studio – preview in Mobile Browser Simulator



- Perform device specific tests in the Mobile Browser Simulator, during development.
- Change device types, orientation, device settings (GPS loc, compass, etc...)
- Supports Cordova and Worklight client API

Agenda

- **IBM Mobile Foundation**
- **Introduction to Worklight**
- **How to Integrate – Worklight Adapters**
- WebAPI
- HTTP & SOAP
- Database (SQL)
- WebSphere Message Broker
- Cast Iron

Adapters

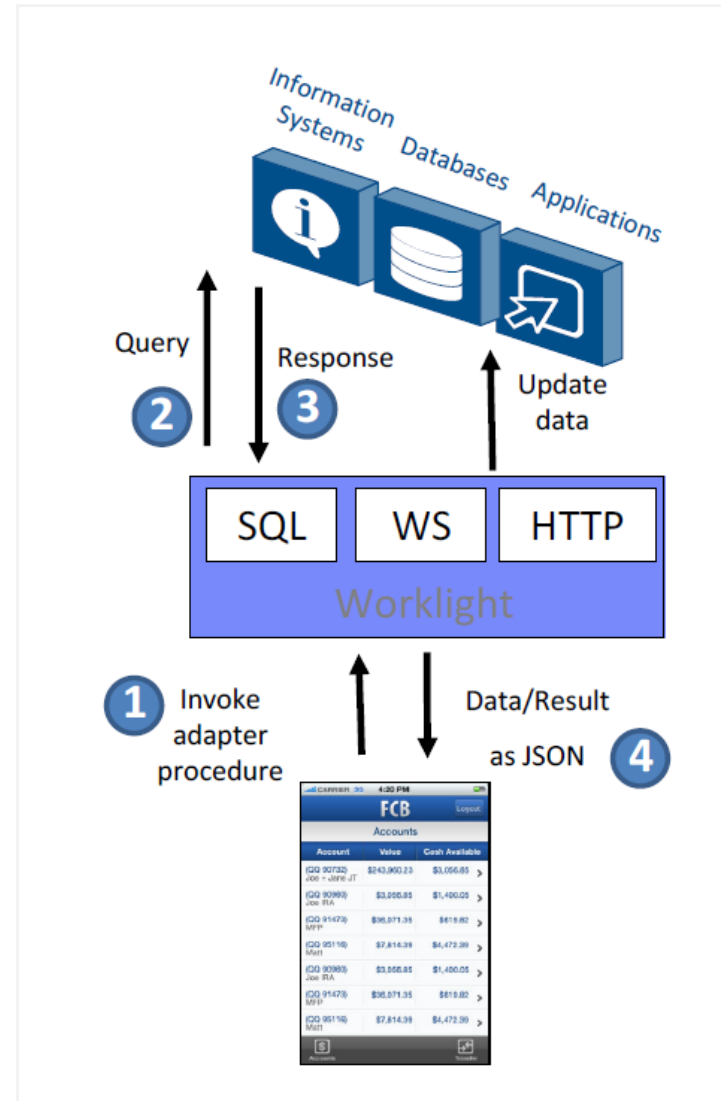
An adapter is a transport layer used by the Worklight® Platform to connect to various back-end systems.

Adapters are used to:

- Retrieve information
- Perform actions

Adapters provided with the product:

- SQL adapter
- HTTP adapter (supports both REST and SOAP)
- Cast Iron® adapter



Adapter Benefits

- **Universality**
Supports multiple integration technologies and back-end information systems
- **Read-only as well and Transactional Capabilities**
Adapters support read-only and transactional access modes to back-end systems
- **Fast Development**
Use simple XML syntax and easily configured with JavaScript API
- **Security**
Use of flexible authentication facilities to create connections with back-end systems
Adapters offer control over the identity of the connected user
- **Scalability**
Adapters can reduce the number of transactions on back-end systems by using cache to store retrieved back-end data
- **Transparency**
Data retrieved from back-end applications is exposed in a uniform manner regardless of the adapter type

What's in an Adapter...

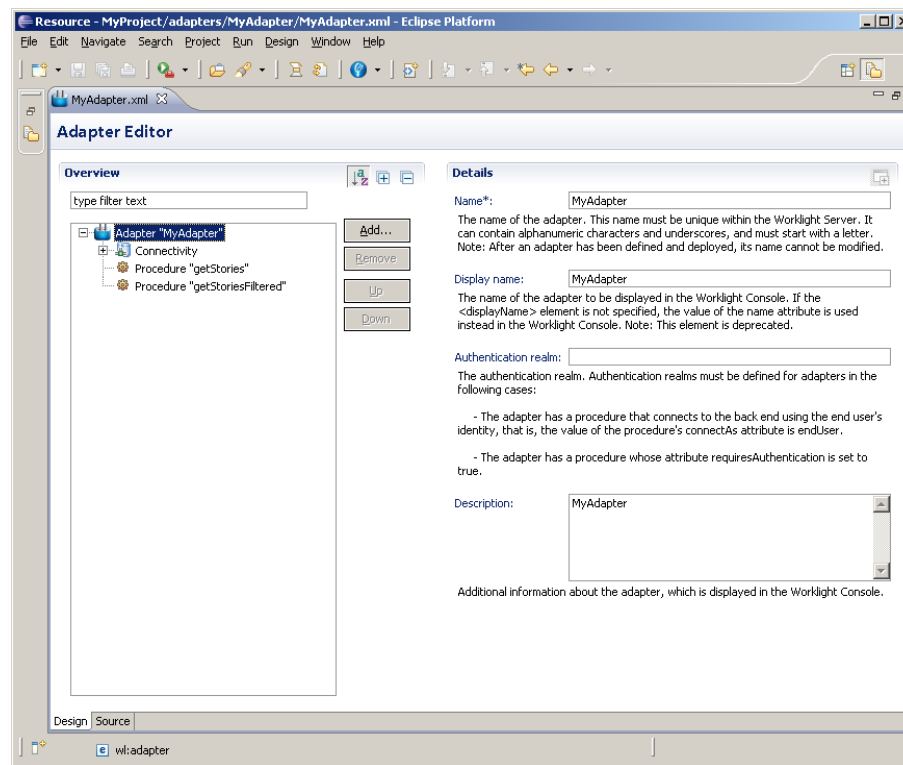
Each Worklight adapter consists of:

- An XML file, describing the connectivity options and listing the procedures exposed to the application or other adapters.
- A JavaScript file, containing the implementation of procedures declared in the XML file.
- Zero or more XSL files, containing a transformation scheme for retrieved raw XML data.

Data retrieved by an adapter can be returned

- raw
- or preprocessed by the adapter itself.

Always presented to the application as a JSON object.



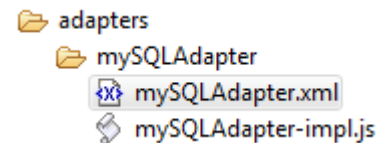
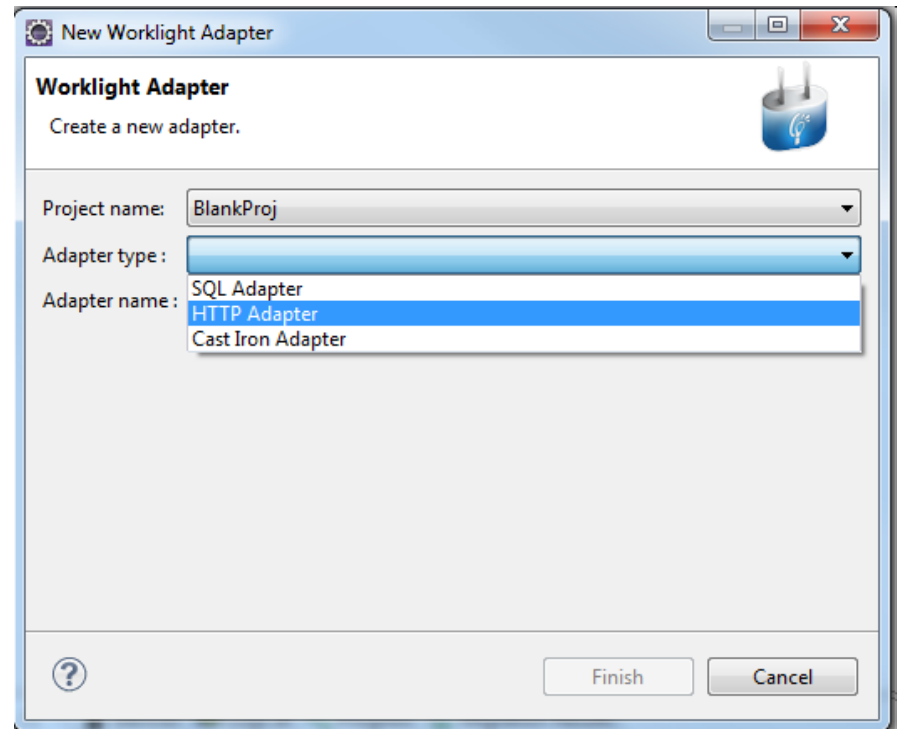
How to Create an Adapter

Creating an adapter

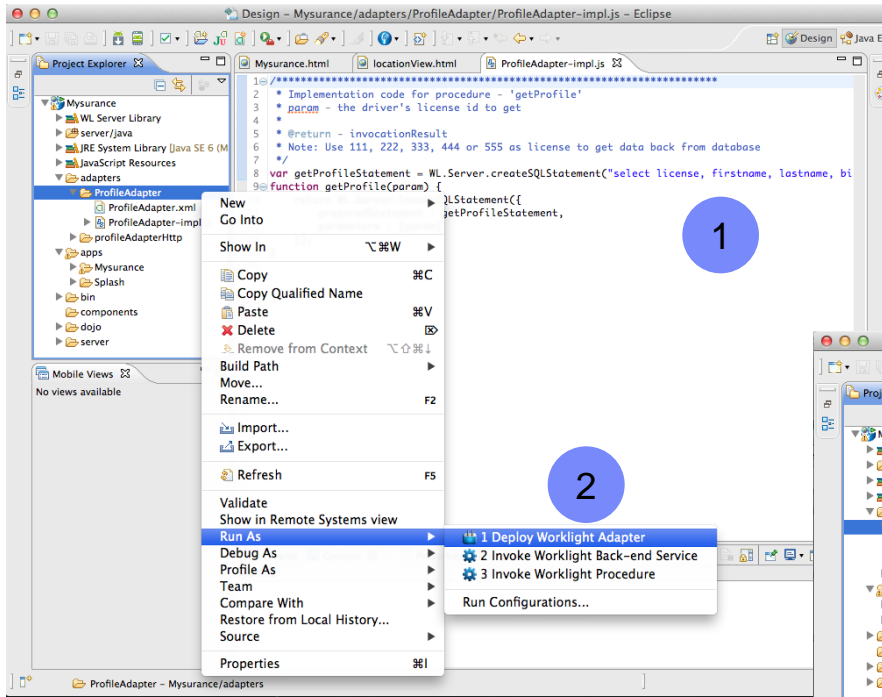
- Select Worklight project
- Select Adapter type
- Applications access the adapter with this name
- Click **Finish**

The result

- XML files which declare procedures and connection properties
- JavaScript file which defines procedures and adapter logic

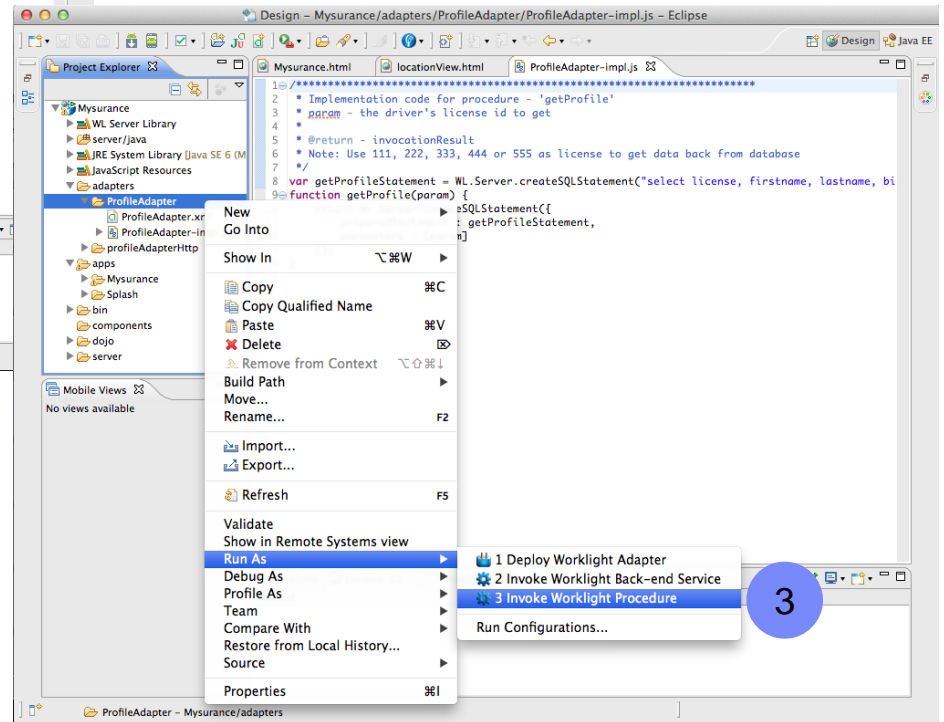


Developing and Testing Adapters



(1) Implement Adapter procedures

(2) Deploy Adapter to embedded web container



(3) Invoke Adapter Procedure

Agenda




- **IBM Mobile Foundation**
- **Introduction to Worklight**
- **How to Integrate – Worklight Adapters**
- **WebAPI**
- HTTP & SOAP
- Database (SQL)
- WebSphere Message Broker
- Cast Iron

WebAPI

IBM Cast Iron Web API is a 'software as a service' offering that enables companies to rapidly create new web APIs, socialize the web APIs in various communities, and manage web APIs.

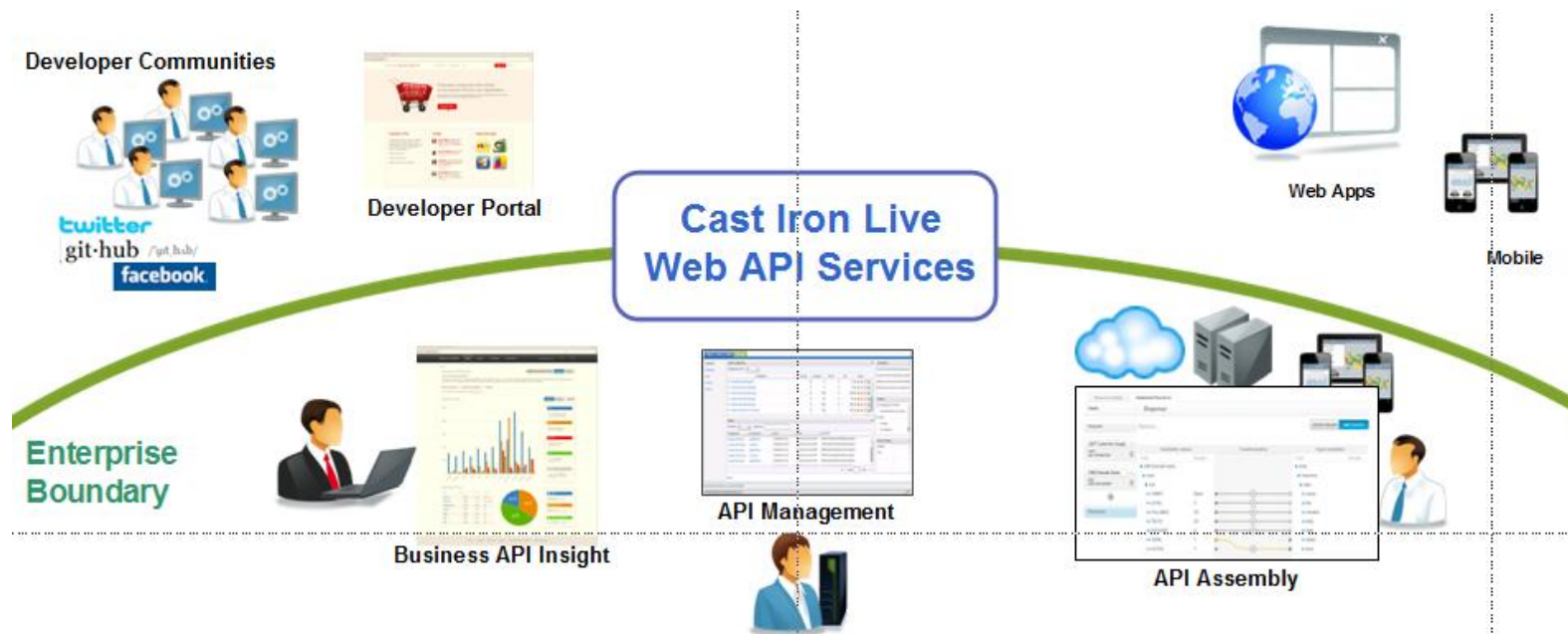
Getting Started with Web API Don't Show Again

Welcome to Web API. The following outlines all of the tasks that Web API can help you with. How do you want to get started?
To access this Getting Started page at any time, access the user menu top-right and click Show Getting Started.

 <i>Create</i>	 <i>Socialize</i>	 <i>Manage</i>
Create new APIs in hours by assembling data from multiple backend systems and Cloud applications using a configuration, not coding approach. Define and Implement APIs	Socialize these APIs to customers, partners and communities of developers and gain instant adoption. Customize & Publish Developer Portal	With built-in dashboards, you can manage service levels, set quotas, establish controls, setup security policies, manage communities and analyze trends. Analyze API Traffic

<https://www.webapi.castiron.com>

WebAPI



- Up and running in minutes
- Proxy to existing services
- Rapidly assemble new APIs
- Documentation of APIs
- Full analytics
- Rate limiting
- Developer portal
- Caching and flood control

- **IBM Mobile Foundation**
- **Introduction to Worklight**
- **How to Integrate – Worklight Adapters**
- **WebAPI**
- **HTTP & SOAP**
- Database (SQL)
- WebSphere Message Broker
- Cast Iron

HTTP Adapter

- Works with RESTful and SOAP-based services
- Can read structured HTTP sources, for example RSS feeds
- Allows sending a GET or POST HTTP request and retrieves data from the response headers and body
- Easily customizable with simple server-side JavaScript
- Optional server-side filtering
- Retrieved data can be in XML, HTML, JSON, or plain text formats

HTTP Adapter: Javascript file

To invoke an HTTP request, use the `WL.Server.invokeHttp` method.

Provide an input parameters object specifying:

- HTTP method: GET or POST
- Returned content type: XML, JSON, HTML, or plain
- Service path
- Query parameters
- Request body
- Transformation type

```
function getStoriesFiltered(interest) {  
    var path = 'rss/' + interest + '.rss';  
    var input = {  
        method : 'get',  
        returnedContentType : 'xml',  
        path : path,  
        transformation : {  
            type : 'xslFile',  
            xslFile : 'filtered.xsl'  
        }  
    };  
  
    return WL.Server.invokeHttp(input);  
}
```

XSL Transformation Filtering

XSL transformation can be applied to the received data

SOAP Requests

The `WL.Server.invokeHttp` method can be used to create a SOAP envelope which can be sent directly

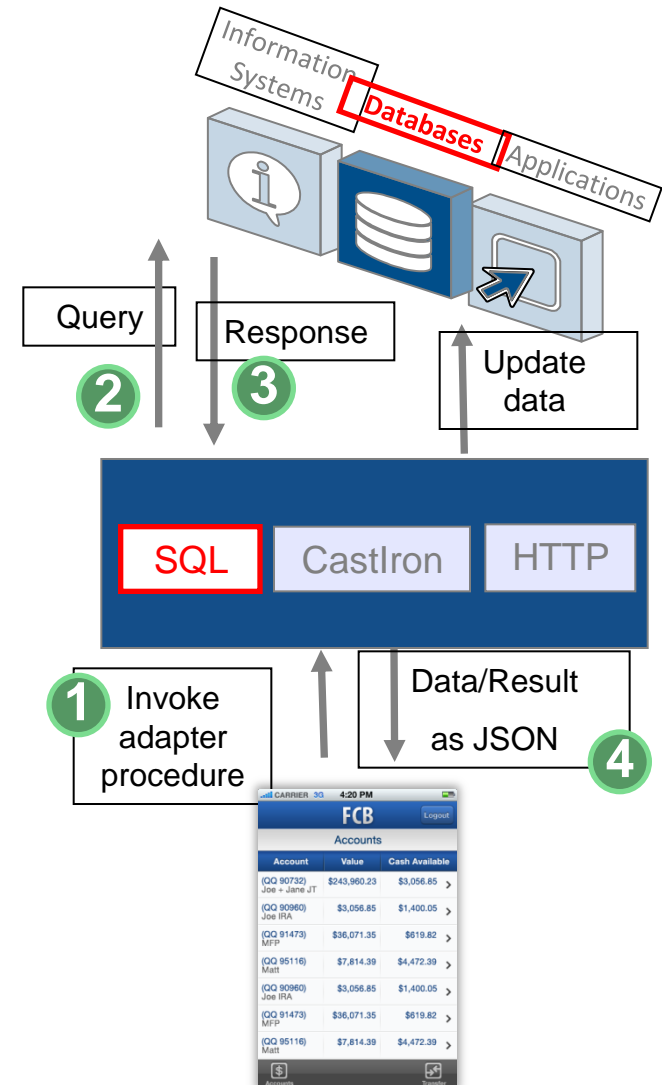
- To invoke a SOAP-based service in an HTTP adapter you need to encode the SOAP XML envelope within the request body
- Encoding XML within JavaScript is simple by using E4X, which is officially part of JavaScript 1.6
- This technology can be used to encode any type of XML document, not necessarily SOAP envelopes

```
var request =
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
"http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <CelsiusToFahrenheit xmlns="http://tempuri.org/">
      <Celsius>{celsiusTemp}</Celsius>
    </CelsiusToFahrenheit>
  </soap:Body>
</soap:Envelope>;
```

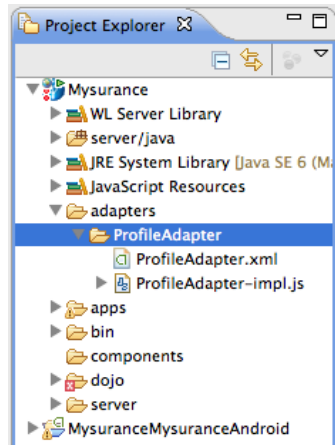
- **IBM Mobile Foundation**
- **Introduction to Worklight**
- **How to Integrate – Worklight Adapters**
- **WebAPI**
- **HTTP & SOAP**
- **Database (SQL)**
- **WebSphere Message Broker**
- **Cast Iron**

Worklight SQL Adapters

- An Adapter is a transport layer used by the Worklight Platform to connect to various back-end systems.
 - Executed on Worklight server
 - Implemented in Javascript by default
 - Can be custom-coded in Java
 - Simple client-side Javascript invocation model through Worklight client API
- A Worklight® SQL adapter is designed to communicate with any SQL data source
 - Both plain SQL queries or stored procedures can be used
 - Worklight supports MySQL, Oracle 11g, DB2® and Derby databases
 - JDBC connector driver must be downloaded separately by the developer and added to the build path of the Worklight project



Implementing a Worklight SQL Adapter



Adapter XML file

- dataSourceDefinition
 - JDBC driver class
 - JDBC database URL
 - userid
 - password
- procedure definition(s)



```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <wl:adapter name="ProfileAdapter"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xmlns:wl="http://www.worklight.com/integration"
6   xmlns:sql="http://www.worklight.com/integration/sql">
7
8   <displayName>ProfileAdapter</displayName>
9   <description>ProfileAdapter</description>
10  <connectivity>
11    <connectionPolicy xsi:type="sql:SQLConnectionPolicy">
12      <!-- Example for using a JNDI data source, replace with actual data source name -->
13      <!-- <dataSourceJNDIName>java:/data-source-jndi-name</dataSourceJNDIName> -->
14
15      <!-- Example for using MySQL connector, do not forget to put the MySQL connector
16      library in the project's lib folder -->
17      <dataSourceDefinition>
18        <driverClass>com.mysql.jdbc.Driver</driverClass>
19        <url>jdbc:mysql://localhost:3306/potdb</url>
20        <user>potuser</user>
21        <password>password</password>
22      </dataSourceDefinition>
23    </connectionPolicy>
24
25    <!-- For Tomcat use following -->
26    <!-- <dataSourceJNDIName>${potdb-jndi-name}</dataSourceJNDIName> -->
27    <!-- For WAS use following -->
28    <!-- <dataSourceJNDIName>jdbc/potdb</dataSourceJNDIName> -->
29
30    <loadConstraints maxConcurrentConnectionsPerNode="5" />
31  </connectivity>
32
33  <!-- Replace this with appropriate procedures -->
34  <procedure name="getProfile"/>
35
36 </wl:adapter>
  
```

Adapter Implementation (Javascript)

- Procedure source
 - Prepare SQL statement
 - Return results from invoking SQL



```

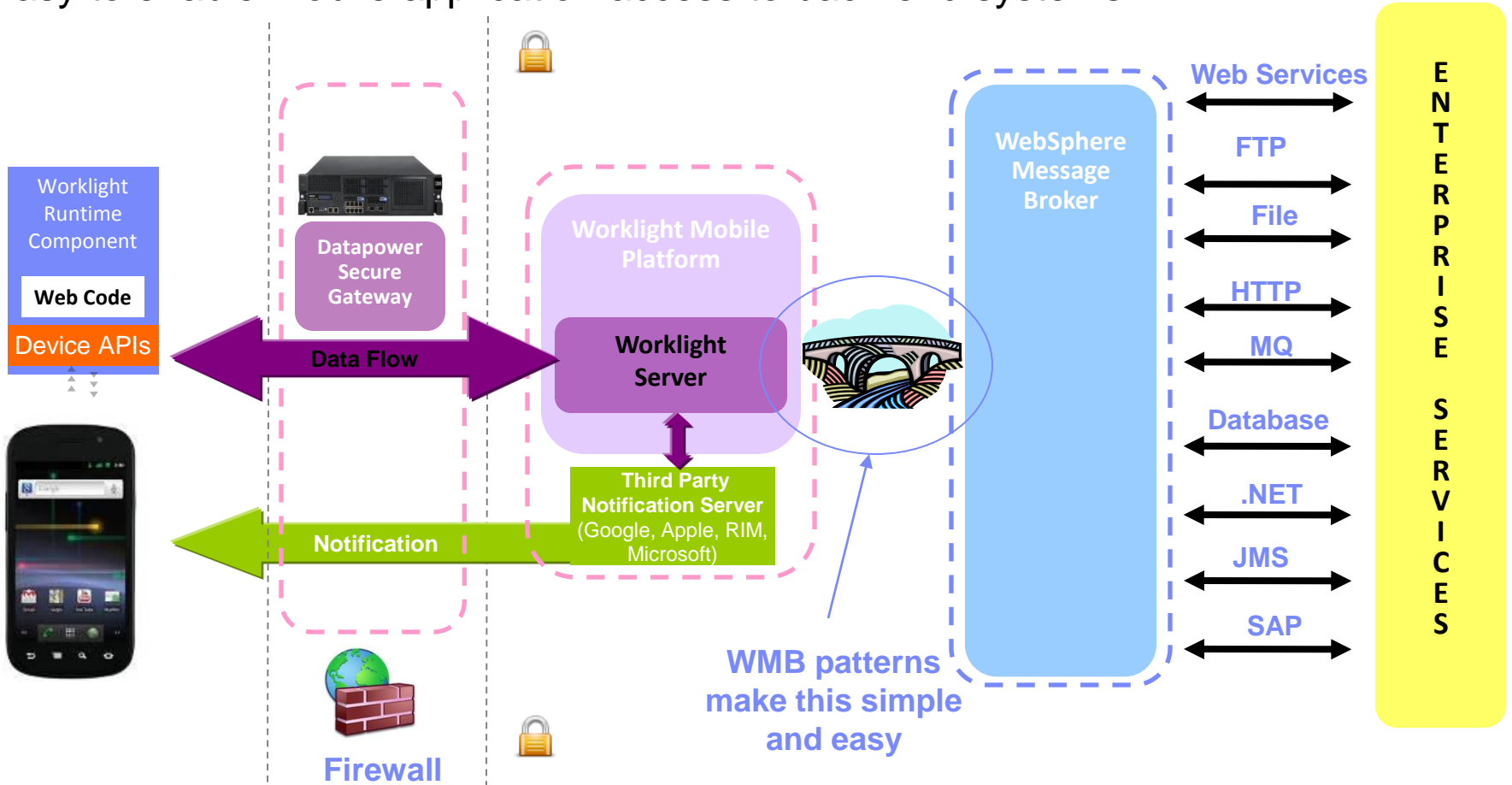
1 *****
2 * Implementation code for procedure - 'getProfile'
3 * param - the driver's license id to get
4 *
5 * @return - invocationResult
6 * Note: Use 111, 222, 333, 444 or 555 as license to get data back from database
7 */
8 var getProfileStatement = WL.Server.createStatement("select license, firstname, lastname, birthdate from profiles where license = ?");
9 function getProfile(param) {
10   return WL.Server.invokeSQLStatement({
11     preparedStatement : getProfileStatement,
12     parameters : [param]
13   });
14 }
15
  
```

Agenda

- **IBM Mobile Foundation**
- **Introduction to Worklight**
- **How to Integrate – Worklight Adapters**
- **WebAPI**
- **HTTP & SOAP**
- **Database (SQL)**
- **WebSphere Message Broker**
- **Cast Iron**

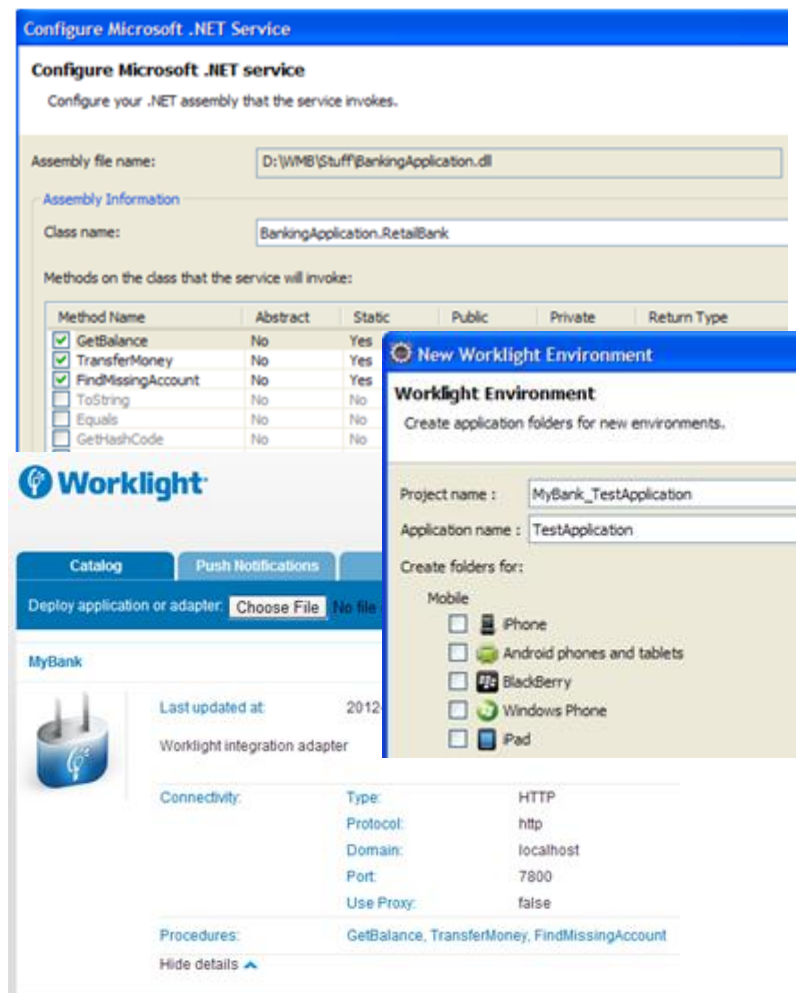
WebSphere Message Broker and Worklight

- Simple to “mobile enable” enterprise services
- Easy to enable mobile application access to back-end systems



WMB Scenario 1 - Expose a Microsoft .NET application as a mobile service

- **Simple to configure** – Drag and drop .NET assembly and enter Worklight adapter details
- **Super quick** – Pattern does all the hard work in less than a minute, generating...
 - A Web service implementation exposing desired operations
 - An adapter **ready for deployment to Worklight Server**
 - A sample mobile application for **easy testing**
 - Optimised for small screen mobile devices; easy to add extra environments for iOS, Android and many more!



The screenshot displays the Worklight console interface. The top section, titled "Configure Microsoft .NET Service", allows for configuring a .NET assembly. The "Assembly file name" is set to "D:\WMB\Stuff\BankingApplication.dll" and the "Class name" is "BankingApplication.RetailBank". A table lists methods on the class that the service will invoke:

Method Name	Abstract	Static	Public	Private	Return Type
<input checked="" type="checkbox"/> GetBalance	No	Yes			
<input checked="" type="checkbox"/> TransferMoney	No	Yes			
<input checked="" type="checkbox"/> FindMissingAccount	No	Yes			
<input type="checkbox"/> ToString	No	No			
<input type="checkbox"/> Equals	No	No			
<input type="checkbox"/> GetHashCode	No	No			

Below this, the "New Worklight Environment" dialog is open, showing the configuration for a new environment. The "Project name" is "MyBank_TestApplication" and the "Application name" is "TestApplication". Under "Create folders for:", the "Mobile" section is expanded, showing checkboxes for "iPhone", "Android phones and tablets", "BlackBerry", "Windows Phone", and "iPad".

The bottom section of the console shows the "MyBank" adapter configuration. It includes a "Deploy application or adapter" button, a "Last updated at" timestamp of "2012", and a "Worklight integration adapter" icon. The "Connectivity" section is expanded, showing the following details:

Type:	HTTP
Protocol:	http
Domain:	localhost
Port:	7800
Use Proxy:	false

The "Procedures" section is also expanded, showing the list: "GetBalance, TransferMoney, FindMissingAccount". A "Hide details" link is visible at the bottom.

WMB Scenario 2 - Mobile enable ANY enterprise service

■ As few as 2 Clicks!

- Right-click on any enterprise service (MQ, Database, Web service, CICS, IMS, etc...)
- Left-click to create mobile service with default options

■ Pattern supports **extra options**...

- Choose operations to be available to mobile applications
- Enable auditing of service requests

The screenshot shows the 'Application Development' interface. A right-click context menu is open over the 'Service Description' of 'AddressBookService'. The 'Patterns' option is selected, which has opened a sub-menu with 'Create Mobile Service' and 'Worklight' options.

Below the context menu, the 'Configure Worklight service' dialog is shown. It displays the service name 'AddressBookService' and a table of selected operations:

Operation Name	Input Type Name	Output type Name	One-way?
<input checked="" type="checkbox"/> SaveAddress	Person	SaveAddressResponse	No
<input checked="" type="checkbox"/> FindAddress	Name	Address	No

At the bottom of the dialog are 'Select All', 'Clear All', 'OK', and 'Cancel' buttons.

To the right, the 'Interface' configuration is visible, showing the 'Configuration' tab with 'Name' set to 'AddressBook' and 'Namespace' set to 'http://addressbook.com/'. Below that, the 'Operations' tab shows a table of operations and their parameters:

Message Type	Name	Type
SaveAddress		
Person	Person	PersonType
SaveAddressResponse	SaveAddressResponse	boolean
FindAddress		
Name	Name	string
Address	Address	AddressType
FindAddressFault	FindAddressFault	FindAddressFaultType

WMB Scenario 3 - Allow mobile apps controlled access to enterprise data

- Simple to enable mobile applications to **Create, Read, Update and Delete** enterprise data
 - Message Broker has excellent support for a wide range of enterprise applications (SAP, Siebel, JDEdwards, PeopleSoft etc...)
 - Pattern generates Worklight adapter and stubs for implementing CRUD operations
- **Quickly configure security policy** to authorize and authenticate access via external LDAP provider
- **One click to cache read resources** in WXS
 - High performing data access crucial for large volumes of mobile devices



Adapter Connection

Find and Discover Services

To discover objects on the enterprise system, create a query by pressing "Edit Query" and then press "Execute Query".

Objects discovered by query:

- ALE
 - Discover IDoc From System (filtered)
 - MATMAS01
 - MATMAS02
 - MATMAS03
 - MATMAS04
 - MATMAS05
 - MATMASDI
 - MATMAS_BAPI01
 - MATMAS_MASS_BAPI01

Objects to be imported:

- MATMAS05

Security

Configure options for security policy enforcement

Security Policy Enforcement

LDAP connection *

LDAP baseDN *

LDAP uid attr

LDAP search scope *

LDAP readers group base DN *

LDAP reader group member

LDAP writers group base DN *

LDAP writer group member

Agenda

- **IBM Mobile Foundation**
- **Introduction to Worklight**
- **How to Integrate – Worklight Adapters**
- **WebAPI**
- **HTTP & SOAP**
- **Database (SQL)**
- **WebSphere Message Broker**
- **Cast Iron**

Cast Iron helps to engage across traditional boundaries

Mobile Applications



Cloud Applications



Web APIs



IBM

WebSphere Cast Iron
Cloud Integration

Extending boundaries
via Cloud apps, mobile
apps and business APIs



Fit for Integration purpose

EAI & Existing Backbone (ESB)

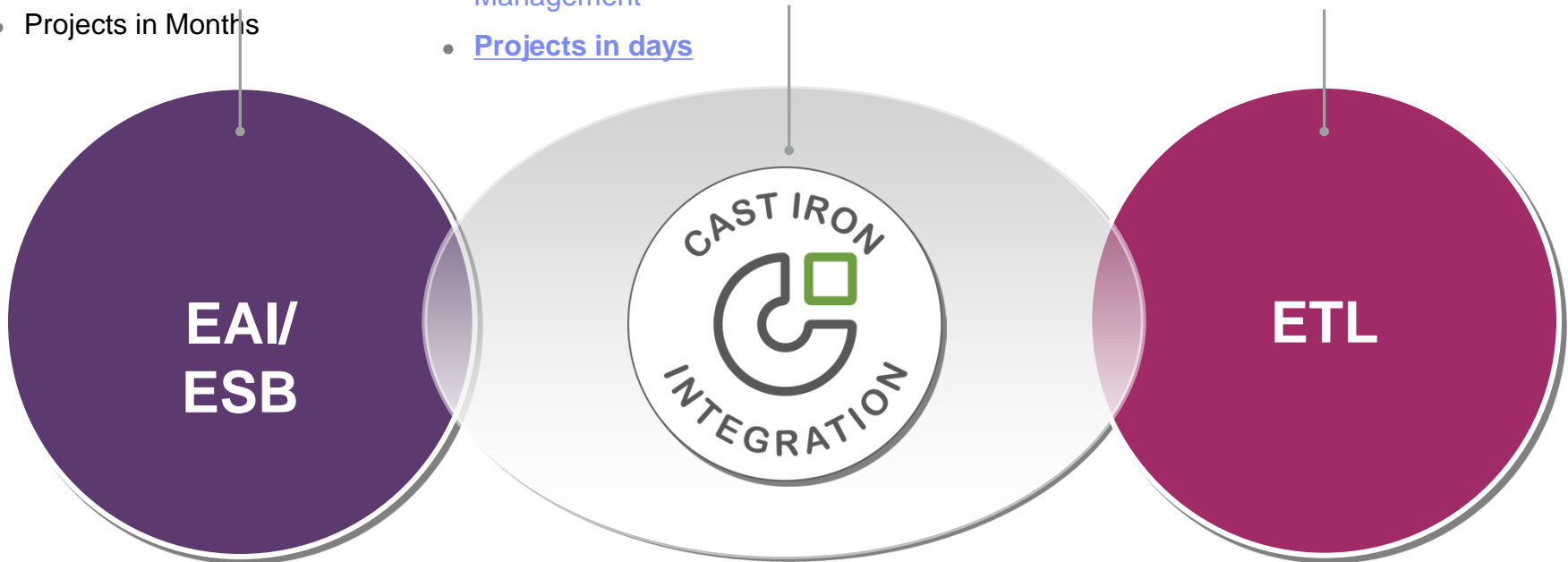
- Focus on On-premise integration
- BPM/BAM
- Human Workflow and very high volumes (> 1 mil/hr)
- Projects in Months

Rapid Application Integration & ESB Augmentation

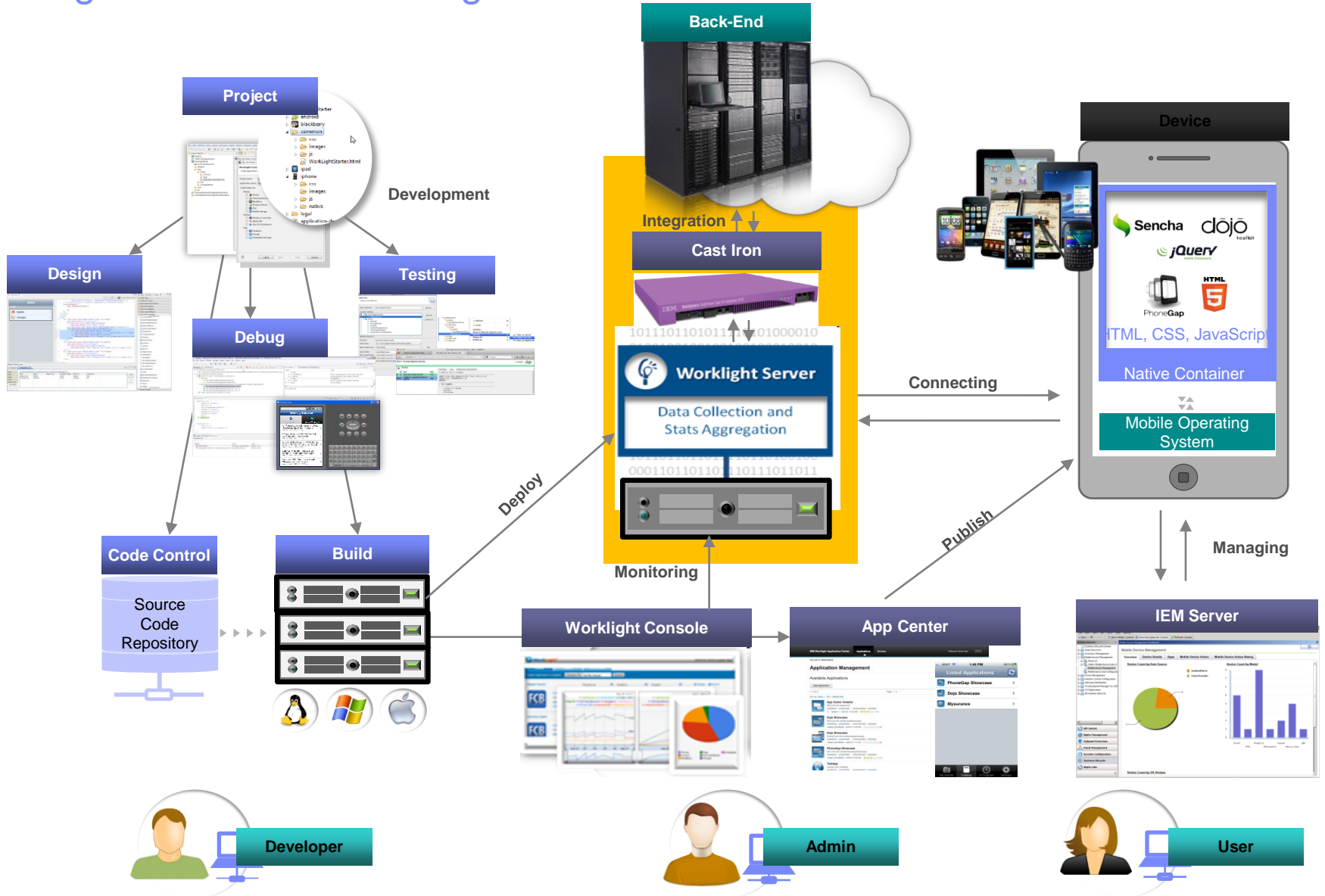
- Focus on Hybrid use cases - Cloud & On Premise, Cloud to Cloud
- Packaged and Custom Apps exposed via standards based connectivity
- Connectivity, Transformation, Logic & Management
- Projects in days

ETL - Cleansing/Loading Master Data

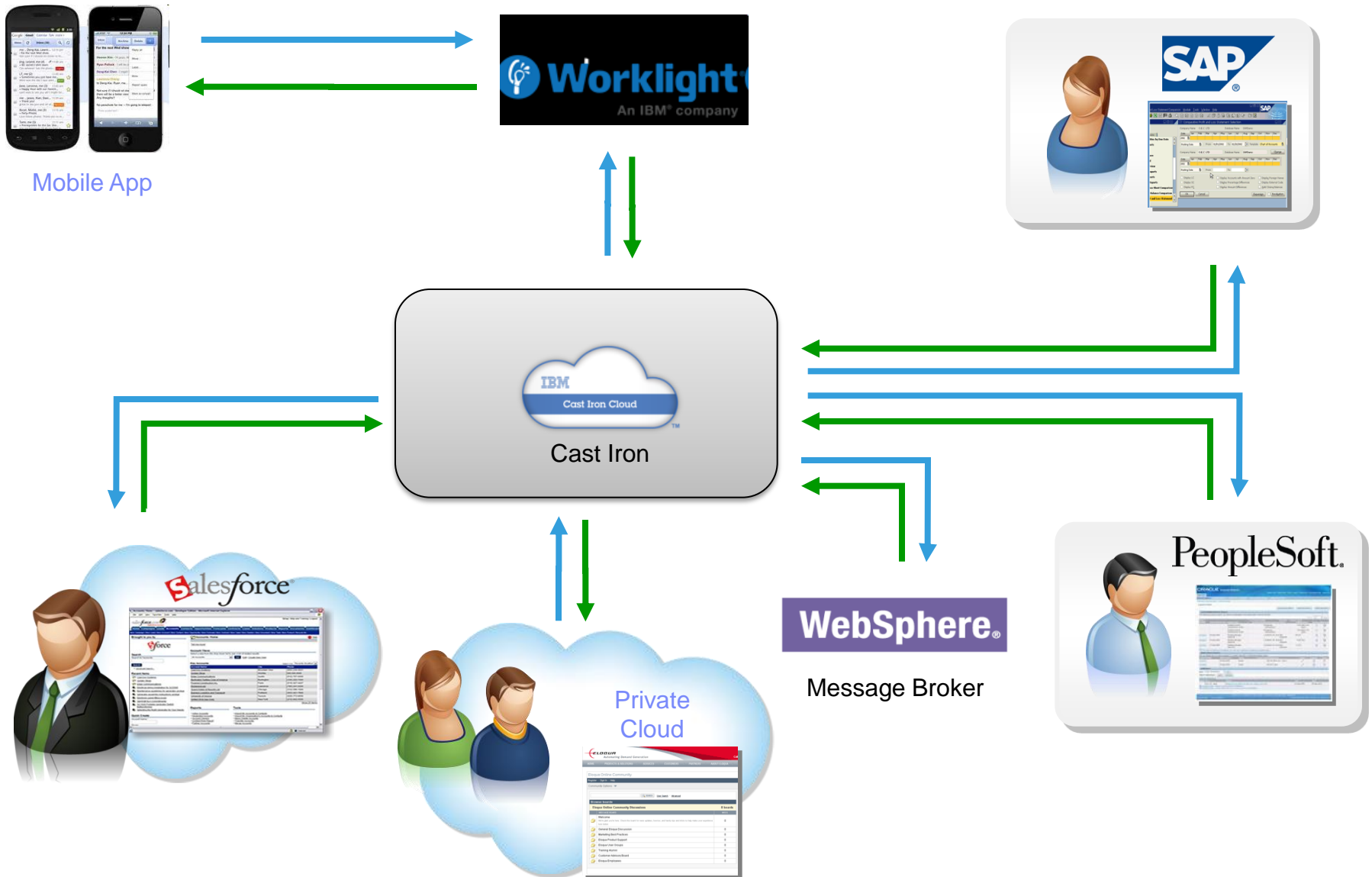
- Mass unload/load (> 1gb)
- Data Cleansing
- Bulk/batch migration
- Projects in weeks



Worklight and Cast Iron Integration Architecture

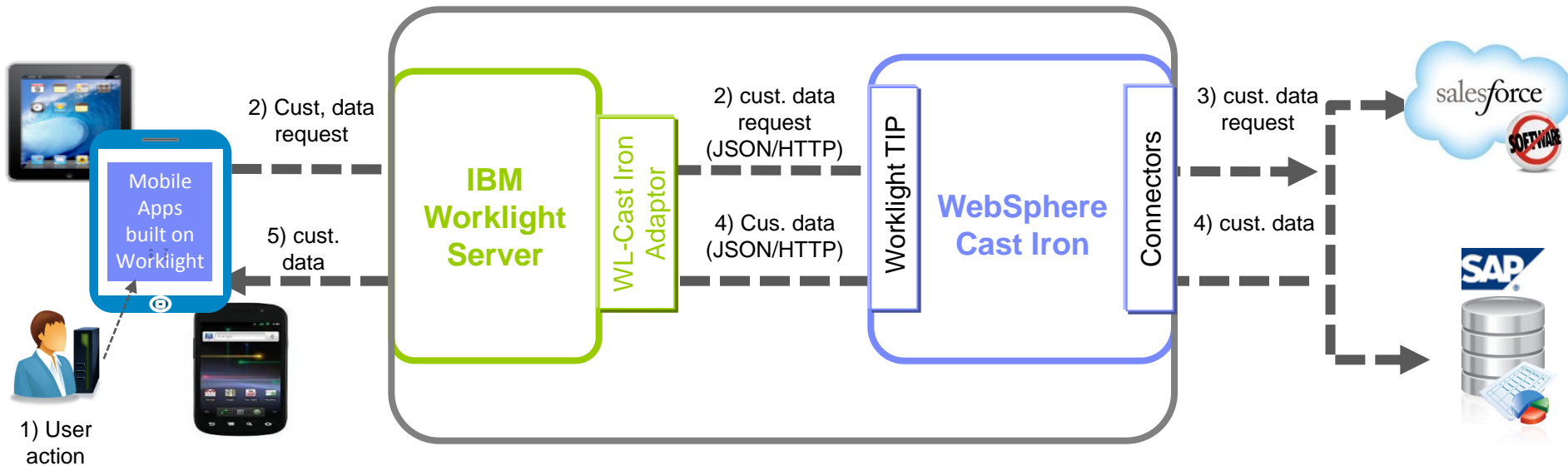


Cast Iron demo integration topology



Cast Iron - How it works

Scenario 1: Worklight Application makes request for enterprise data

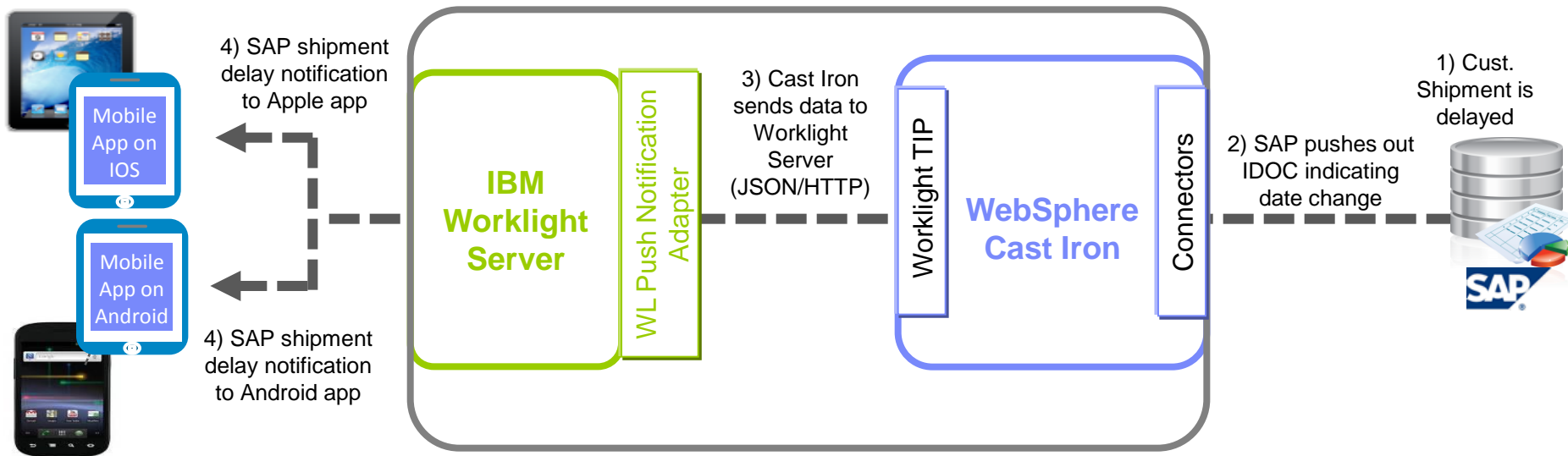


■ Request from Mobile app for customer data from SAP & salesforce

- 1) User invokes action on Mobile application to request data
- 2) Worklight Server invokes it's WL Cast Iron Adaptor to send customer data request to Cast Iron
- 3) Cast Iron receives request & invokes connectivity with salesforce.com and SAP to extract customer data
- 4) Cast Iron sends customer to Worklight Server
- 5) Worklight Server delivers data to Mobile application on the device

Cast Iron - How it works

Scenario 2: Cast Iron provides real-time notification to Mobile applications



▪ Instant notification to Mobile apps on enterprise data changes (e.g. customer shipment delay in SAP)

- 1) Manufacturing is delayed and this causes change of customer shipment date in SAP
- 2) SAP is configured to push out an IDOC indicating this date change. Cast Iron TIP picks up DOC & transforms the data.
- 3) Cast Iron TIP transforms data to Worklight format (JSON) and sends to Worklight Server via HTTP (REST)
- 4) IBM Worklight server receives message and sends notification to Mobile app using appropriate protocol for Android and iOS apps

Questions



Grazie धन्यवाद *Merci* ありがとうございます *Obrigado* 多谢
ITALIAN HINDI FRENCH JAPANESE BRAZILIAN PORTUGUESE SIMPLIFIED CHINESE

Thank You

多謝 Gracias Спасибо நன்றி ชอบรุณ *Danke* شكراً
TRADITIONAL CHINESE SPANISH RUSSIAN TAMIL THAI GERMAN ARABIC

Reference materials

For more information:

- IBM Worklight Training Modules
 - <http://www-01.ibm.com/software/mobile-solutions/worklight/library/>
- IBM Worklight User Documentation
 - <http://www-01.ibm.com/software/mobile-solutions/worklight/library/v50/documentation/>