I got my REST API for z/OS Connect

Now what?

Haley Fung
IMS Mobile and APIM Development Lead
hfung@us.ibm.com

Evgeni Liakhovich
IMS Software Developer
evgueni@us.ibm.com

IBM

Trademarks, copyrights, disclaimers

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at http://www.ibm.com/legal/copytrade.shtml

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Cloud / Mobile topics at the Symposium

IMS Connect: Much more than a TCP/IP Gateway!

- IMS Connect as foundational to cloud and mobile alignment

1 **A02**

Evolving mobile systems of engagement in your enterprise

- Discussion of strategic mobile enablement options in IMS

2 **B02**

Mix it up: How your enterprise assets fit perfectly with Bluemix

- Covers publishing z assets as REST services in z/OS Connect all the way to Bluemix and using the secure gateway and API Management services to access the z services

3 **B14**

I got my REST API for z/OS Connect – Now What?

- Using the IBM MobileFirst platform to build a hybrid mobile application consuming REST/JSON services hosted on the z platform

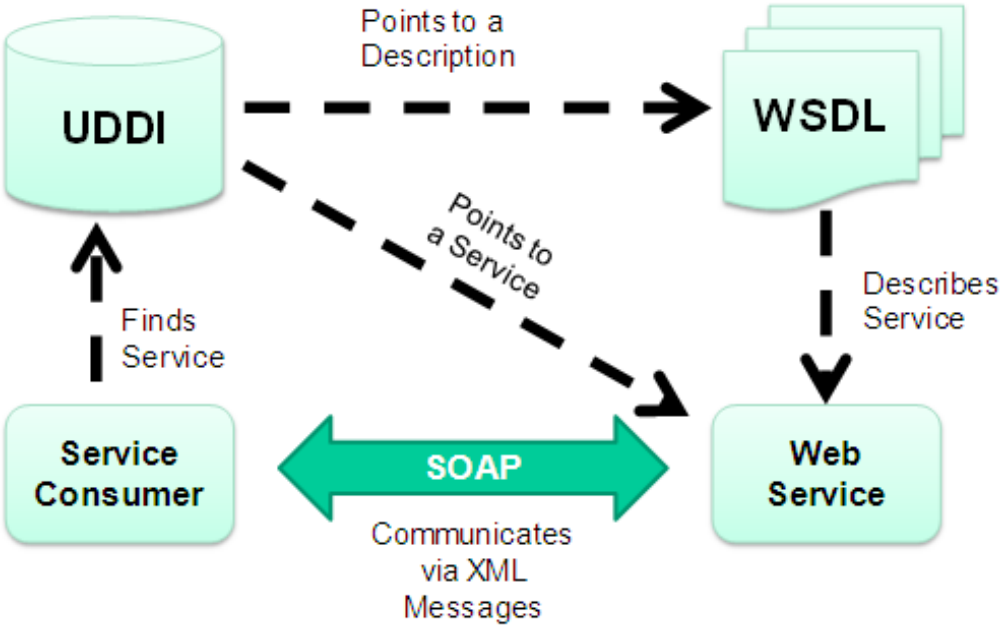
4 **B16**

Evolution of Web Services

From SOAP/WSDL to REST/JSON

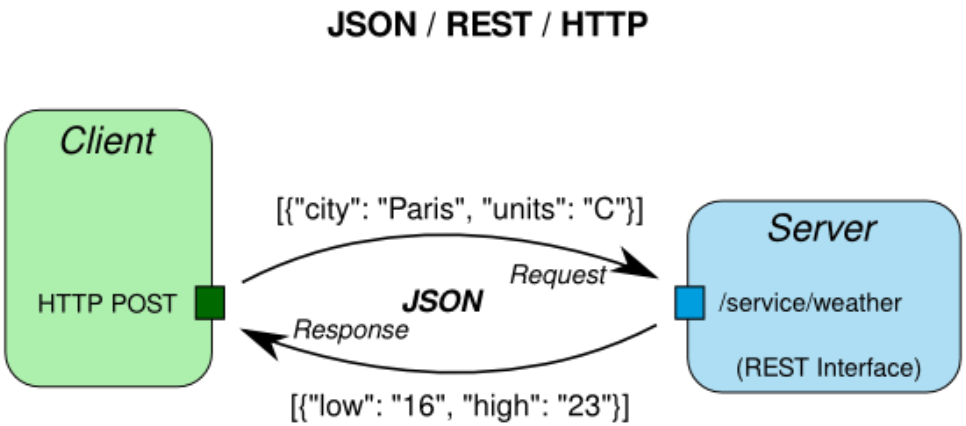
Evolution of Web Services – SOAP/WSDL

- Over the last decade, enterprises adopted traditional Web Services (e.g., SOAP over HTTP)
  - Business-to-business (B2B) and business-to-consumer (B2C) interaction
- SOAP WS are described using Web Services Definition Language (WSDL)
- Input and output messages are described using XML Schema (XSD)
- Provide very rich description of services, demanding "***high ceremony***" approach
- **Problem**: We tried to use them for everything



Evolution of Web Services – REST/JSON

- For mobile and cloud applications, SOAP eventually lost favor to "***low-ceremony***" services that use the REST architectural style
  - RESTful services rely more on the basic convention of the web
  - For example, instead of using specialized registry, they rely on existing common means to locate resources (URL)
- **RESTful** services prefer to use JSON over XML as the wire format
- **JSON** has been referred to as "The fat-free alternative XML"
  - JSON documents are typically more compact than XML that describe the same data, and JSON closely resembles the data structures that it will be marshaled into



IMS Mobile Strategy

Mobile Feature Pack

IMS Mobile Business

- The IMS mobile strategy gives our customers a mobile foundation they can depend on
- Expand IMS ecosystem via delivery of mobile infrastructure
- **New**: *IMS Mobile Feature Pack (Enterprise Suite 3.1 GA June 2014)*
  - Offer an integrated platform for full discovery, modeling, deployment and execution of both transaction and data assets for mobile consumption
  - REST interface with JSON wire protocol
  - A singular approach for System z clients using WAS, CICS, IMS and DB2

74% of CIOs say mobile solutions are part of their vision for increasing competitiveness

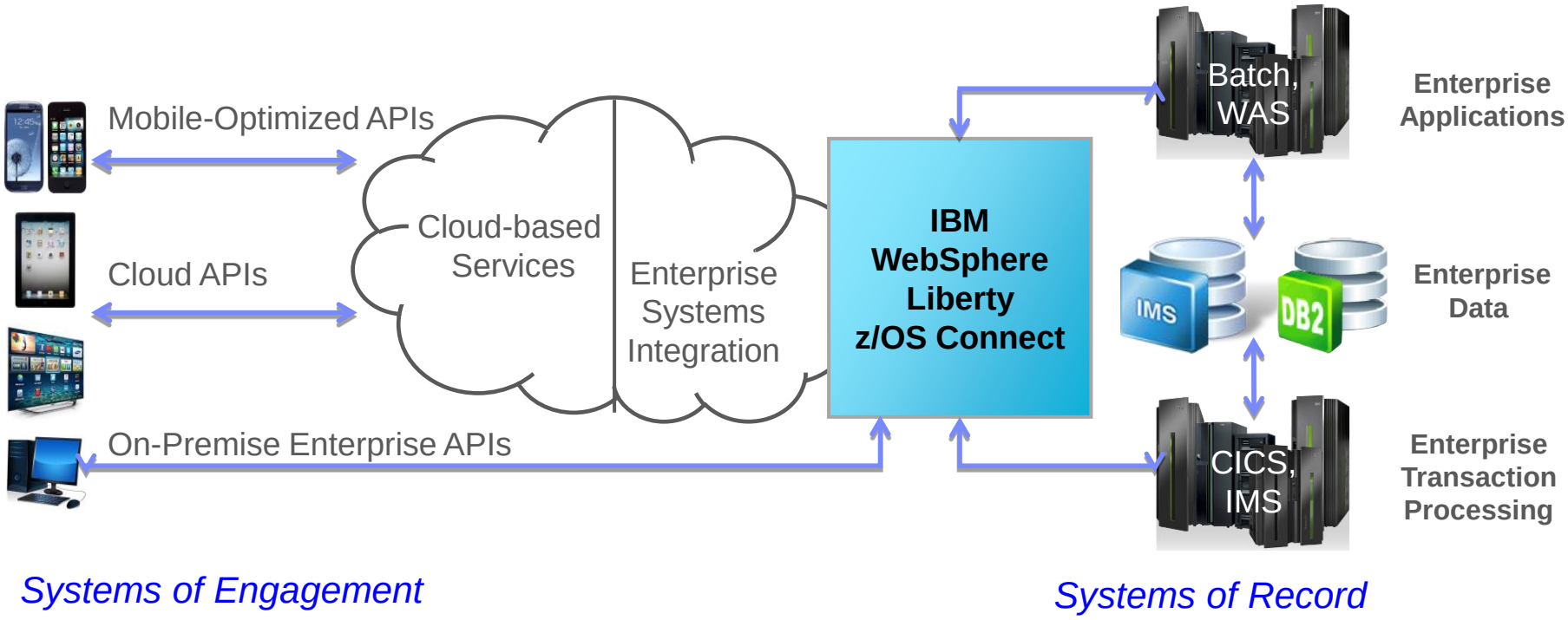
IMS Mobile Feature Pack …

- Supports
  - Discovery of IMS assets
  - Modeling of asset metadata
  - Ability to publish those assets as RESTful services
- Once published, those services are hosted by IMS Mobile for discovery by mobile and cloud application developers
- Associated tooling is delivered via IMS Explorer for Development

Secure and Consistent Enterprise Connectivity for Mobile and Cloud

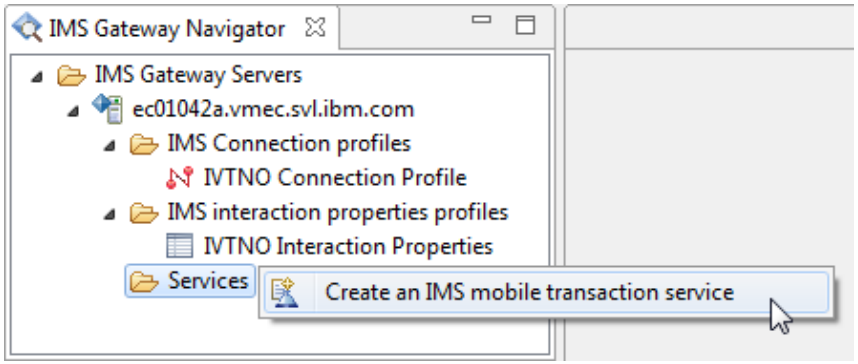
New

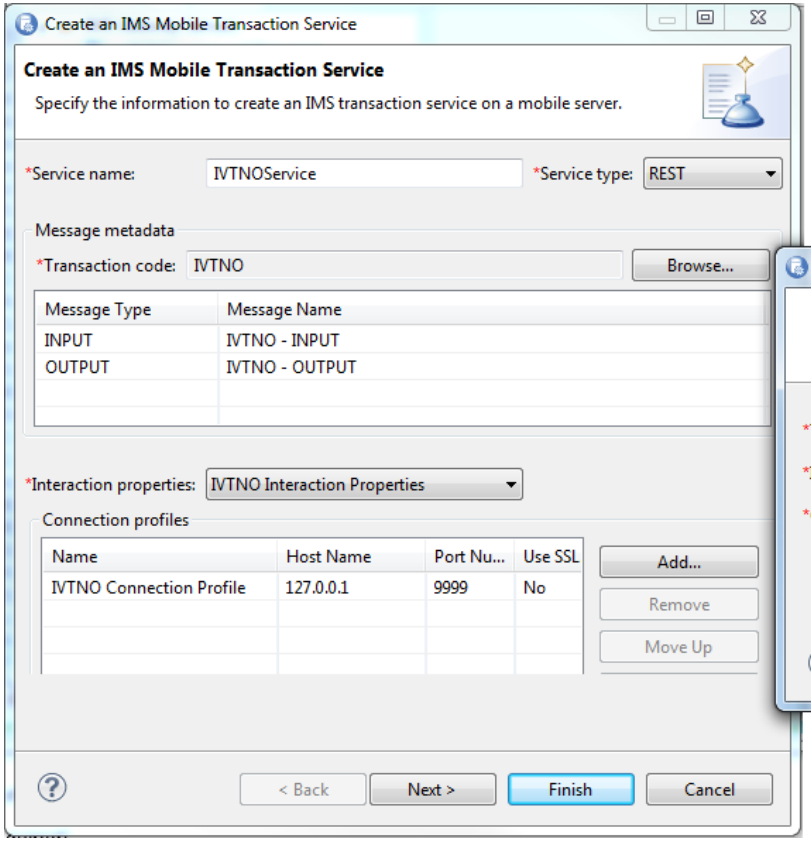
- *IBM WebSphere Liberty z/OS Connect* – Shipped with WAS, CICS, and IMS
- *Unifies z/OS connectors* – a common solutions for mobile, cloud, and web
- *Simplified integration* – Hide complexity of connecting to z/OS using REST

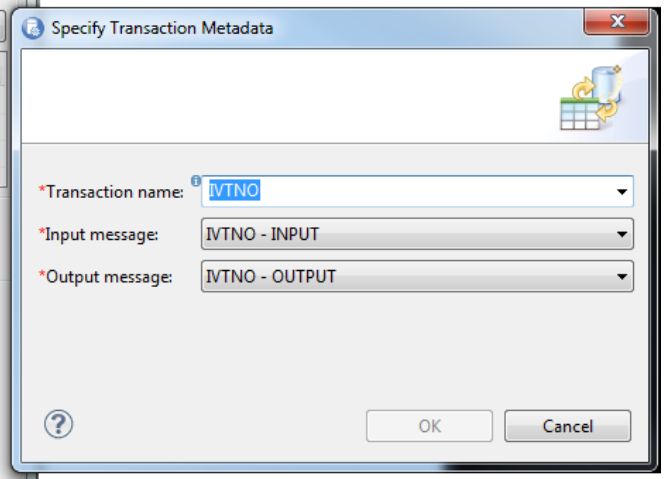


Enable IMS Transactions as REST Services

- Using IMS Explorer (E4D), resources that comprise a service are defined and stored seamlessly on the IMS Mobile Server

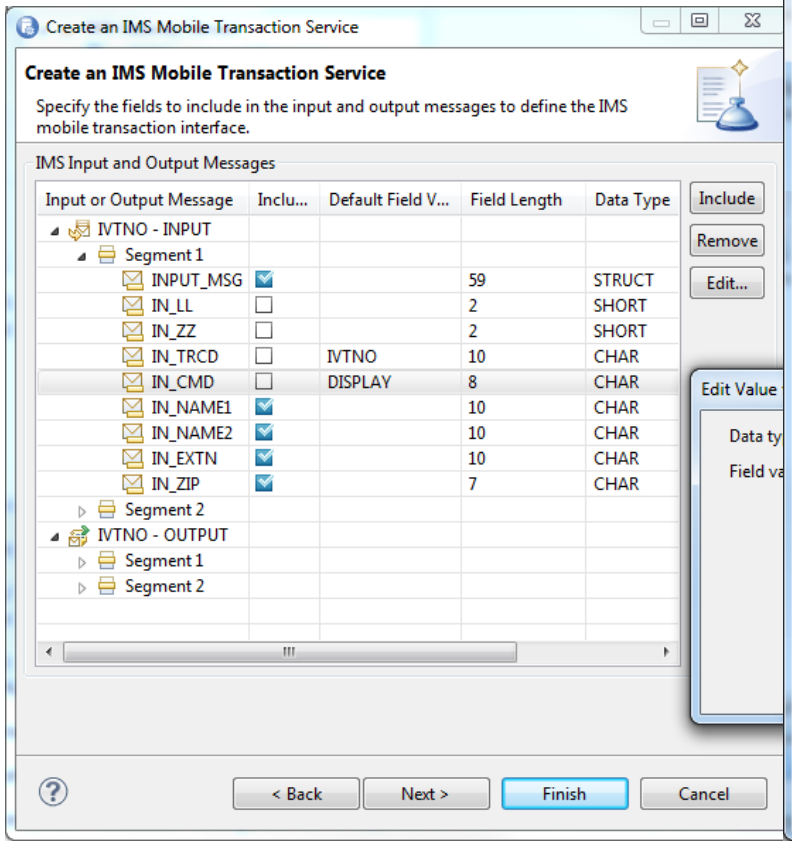


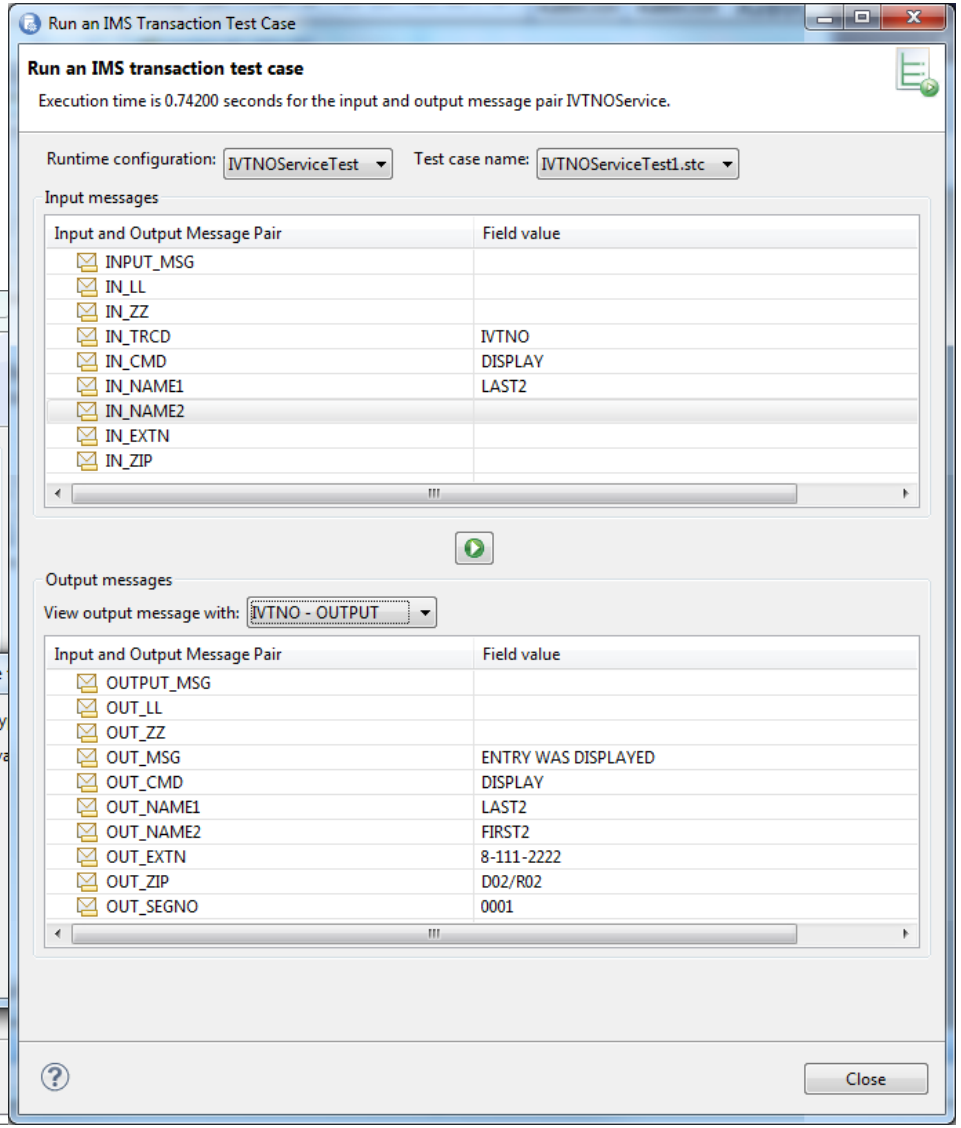




Unit test REST services

- Unit testing an IMS Mobile service using E4D is easy and does not require knowledge of JSON





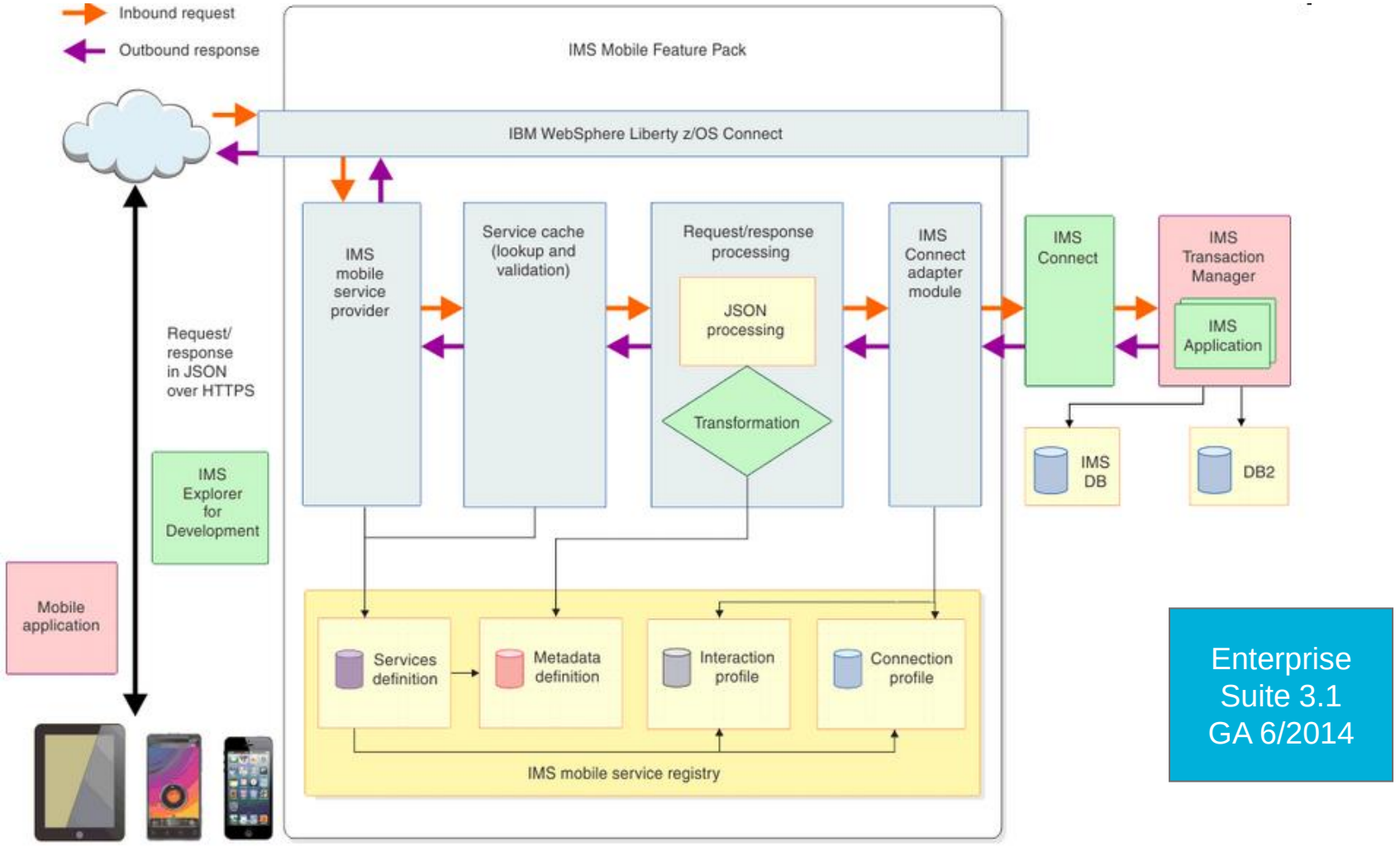
Considerations – Enabling IMS transactions as REST

- REST is stateless
- Best for simple request response type of interaction
- Additional preparation work needed for
  - MFS, Conversational, or generated-COBOL code
  - You may need to hand-code the COBOL copybook for your transaction for E4D tooling
  - You may need to create a simple COBOL code wrapper or convert your conversational transaction to be non-transactional
  - You may to remove the MFS dependency in your application

So, what now?

You published and tested web service – what's next?

IMS Mobile Feature Pack

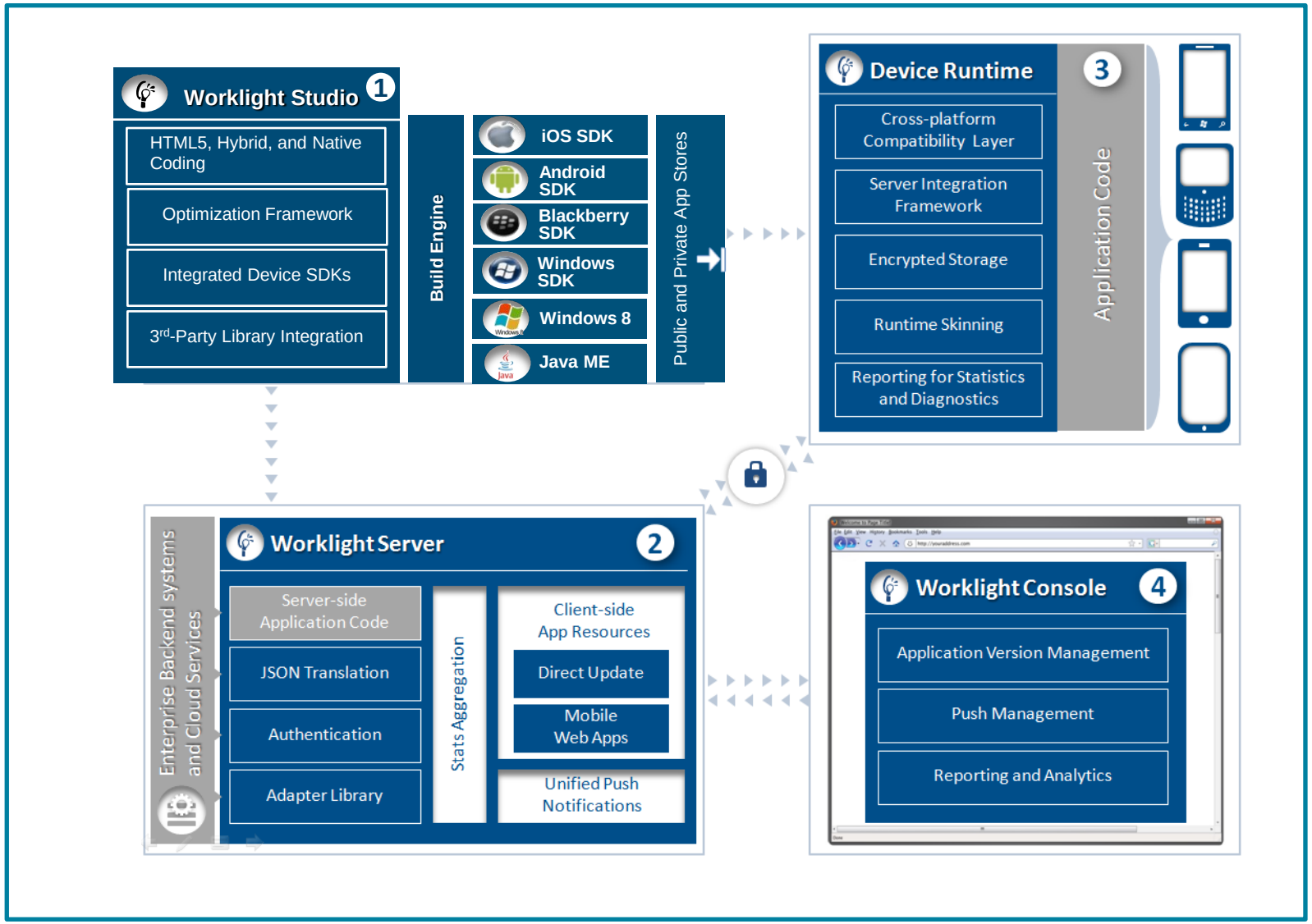


Enable mobile and cloud clients to access IMS transactions as REST/JSON services

IBM Mobile First Platform architecture

formerly known as Worklight

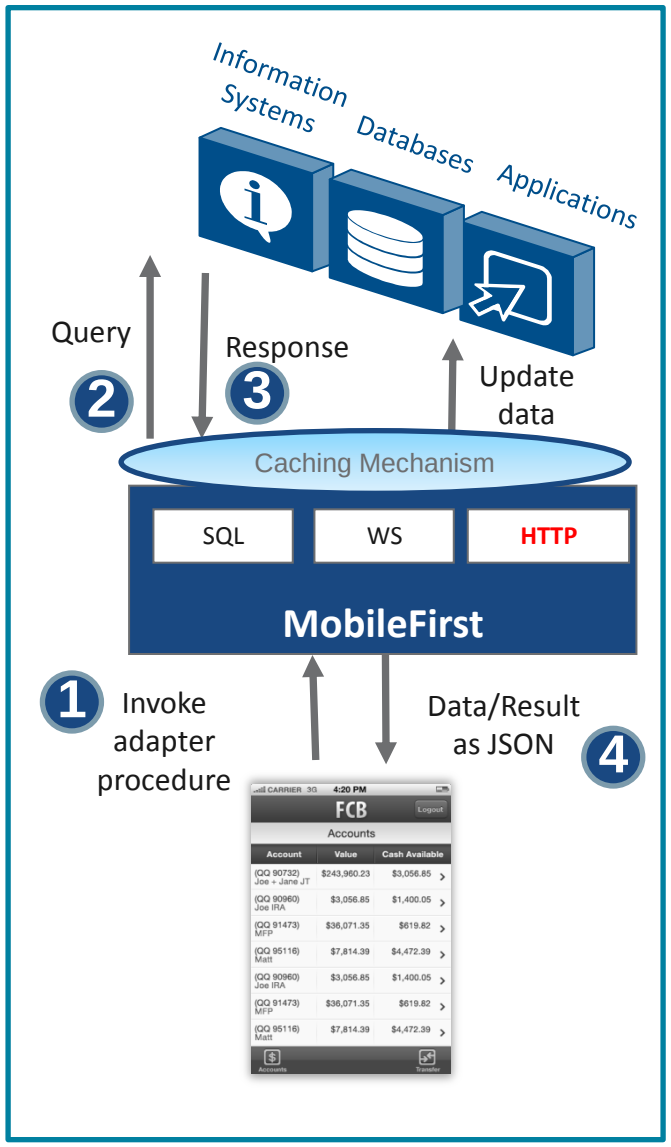
IBM MobileFirst



IBM MobileFirst…

- Supports the development of mobile apps in four ways
  - Web Apps - Quick and low-cost development effort
    - Written entirely in HTML5, CSS and JavaScript code
    - Executed by the mobile browser and therefore cross-platform by default, but less powerful than native apps
  - Hybrid Apps (Web) - The app's source code consists of web code executed within a native container that is provided by Worklight and consists of native libraries
  - Hybrid Apps (Mix) - The web code is augmented with native language to create unique features and access native APIs that are not yet available via JavaScript, such as AR, NFC and others
  - Native Apps - Platform-specific requiring unique expertise and knowledge
    - Pricey and time consuming to develop but delivers the highest user experience

MobileFirst Server- Adapters

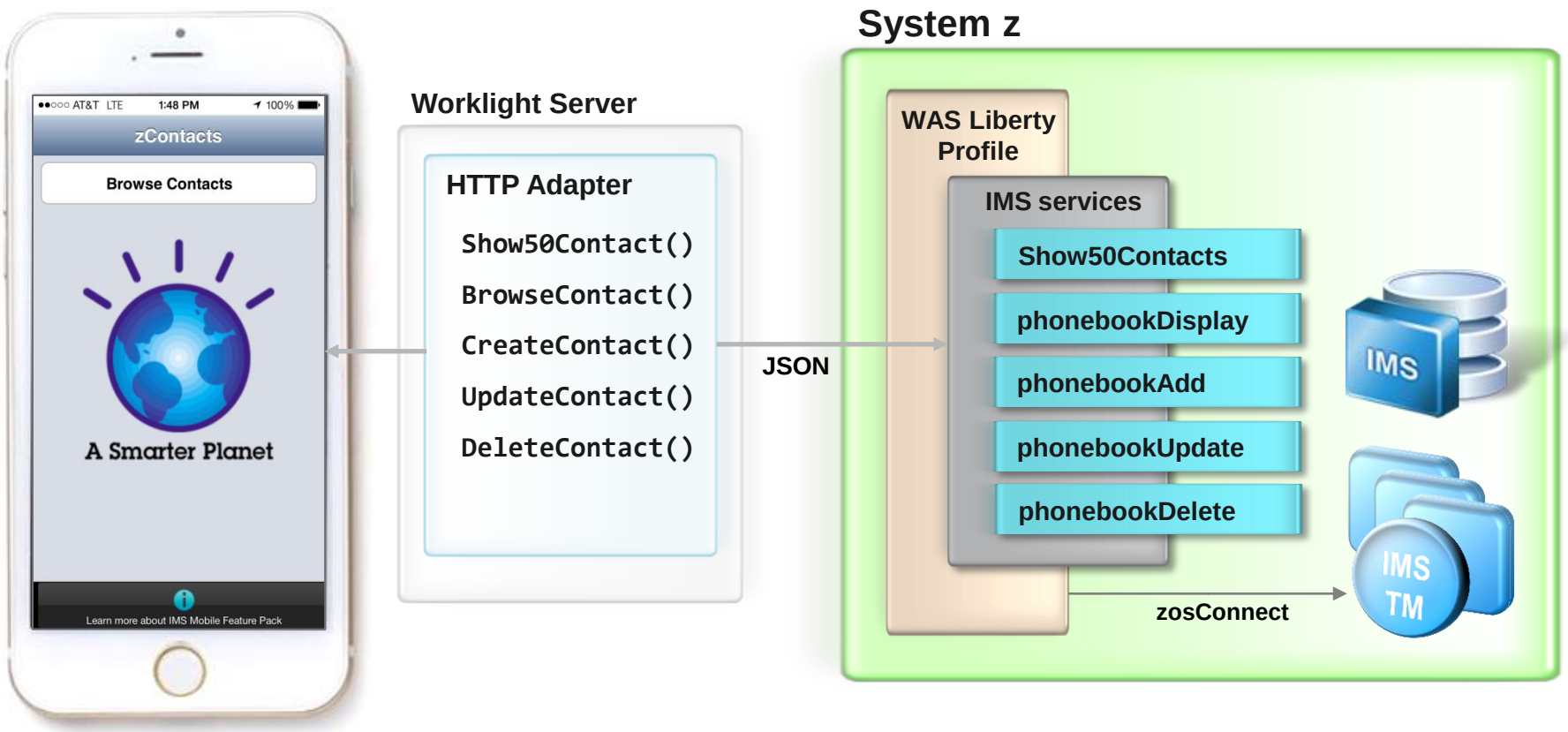


Worklight includes **Adapters** with support for SAP, SOAP, **REST**, SQL, JMS, CASTIRON, LDAP & more

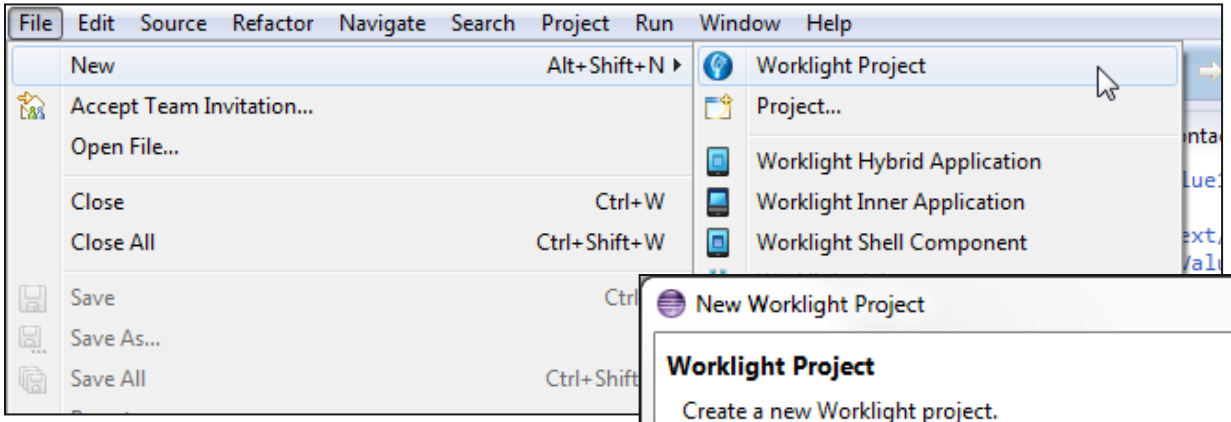
Worklight HTTP Adapter

- 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

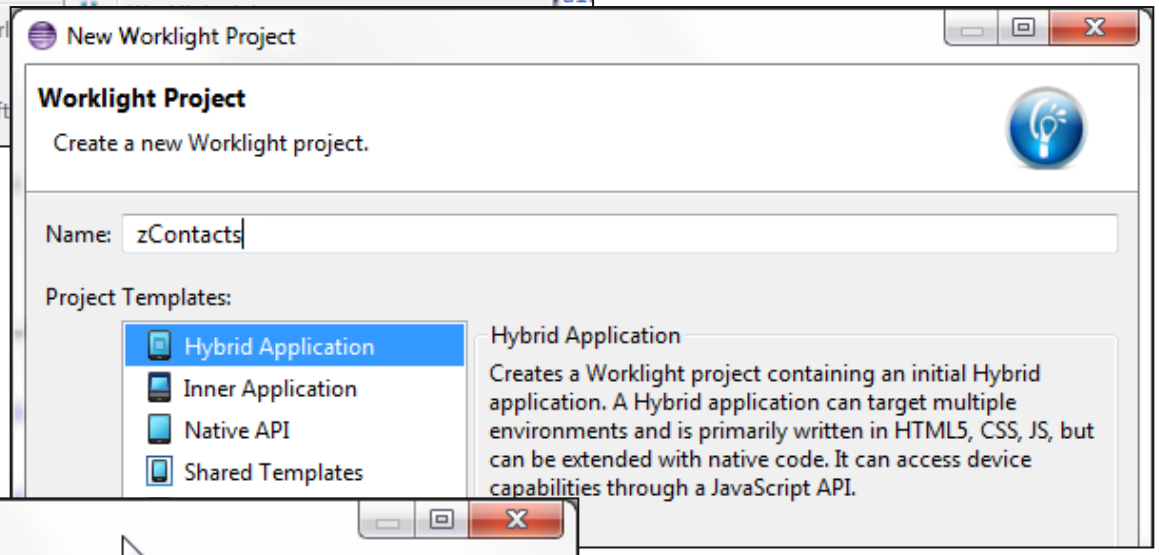
Project: Create mobile app based on IMS service



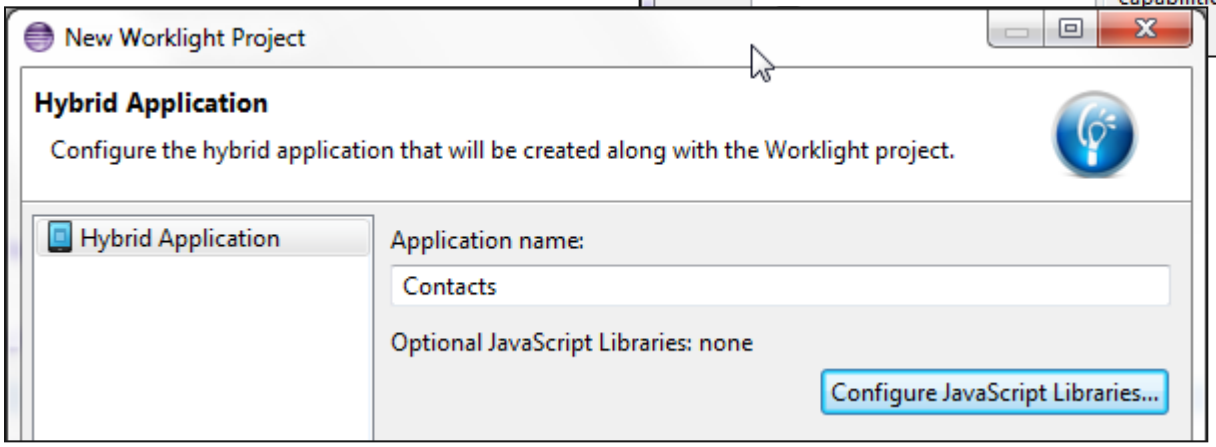
Create Worklight Project



- In the Eclipse menu, select **File > New > Worklight Project**



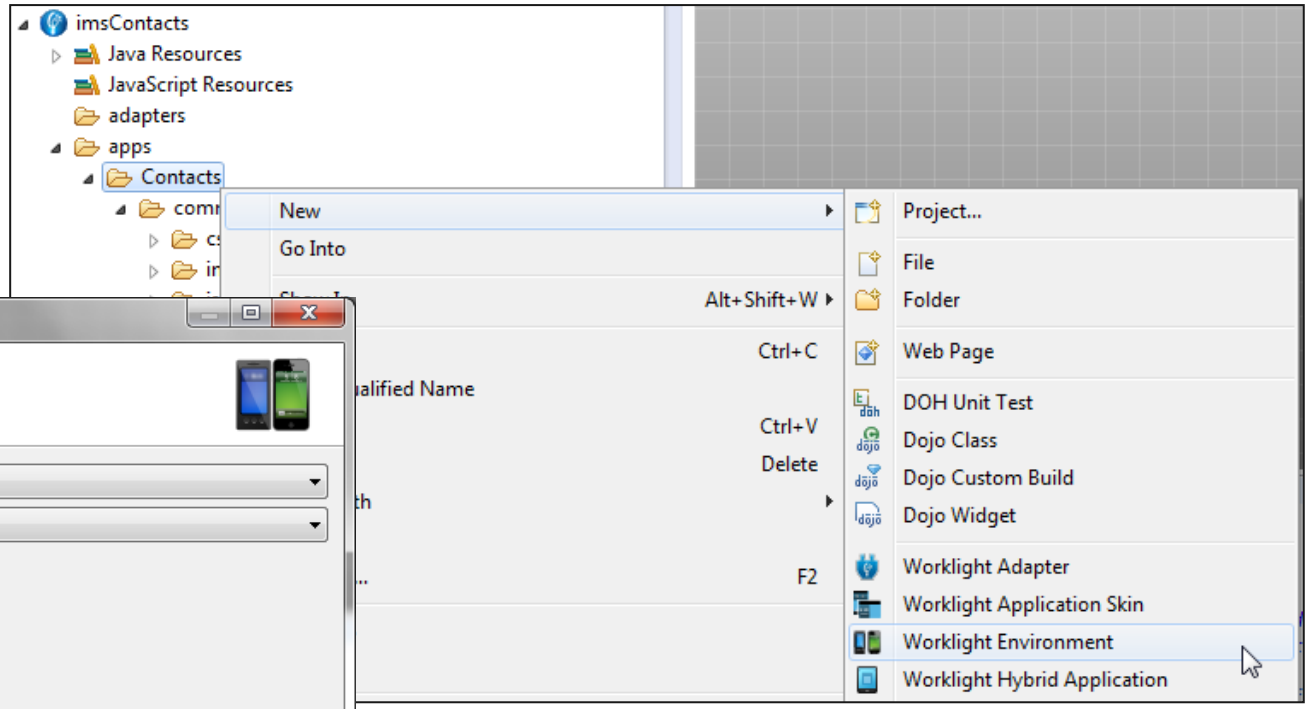
- Enter project name, keep the default Project Templates selection for **Hybrid Application** then click **Next**

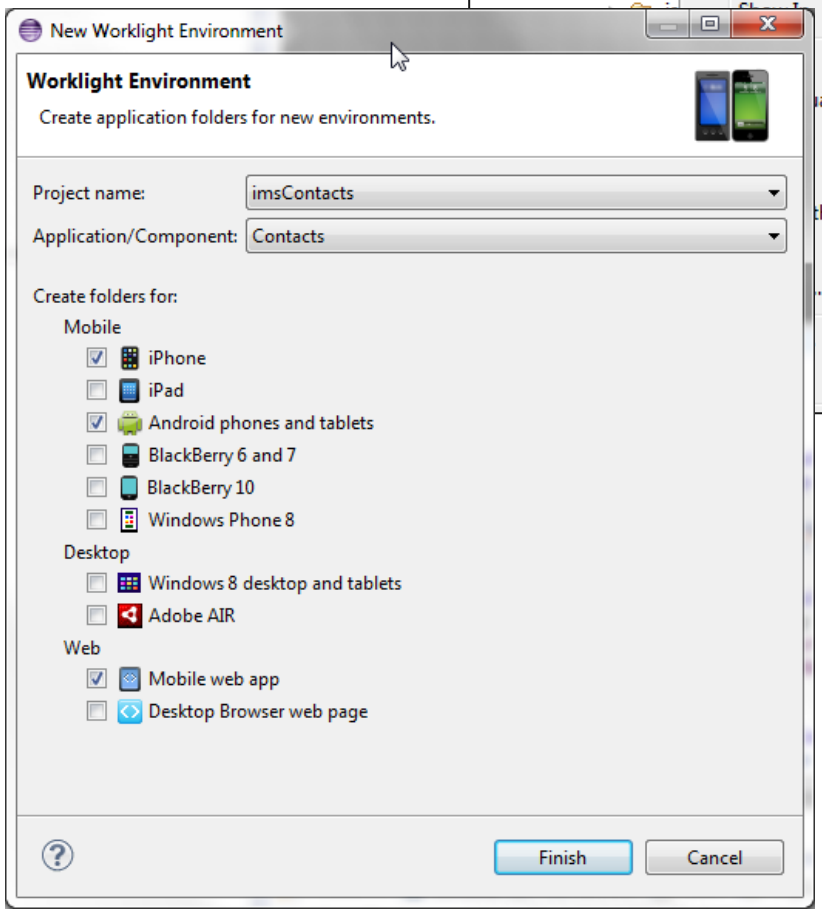


- Enter application name
- Click **Configure JavaScript Libraries** and select the **Add Dojo toolkit** check box then click **Finish**

Add Environments

- In the *Project Explorer* right click the **Contacts** application and select **New > Worklight Environment**



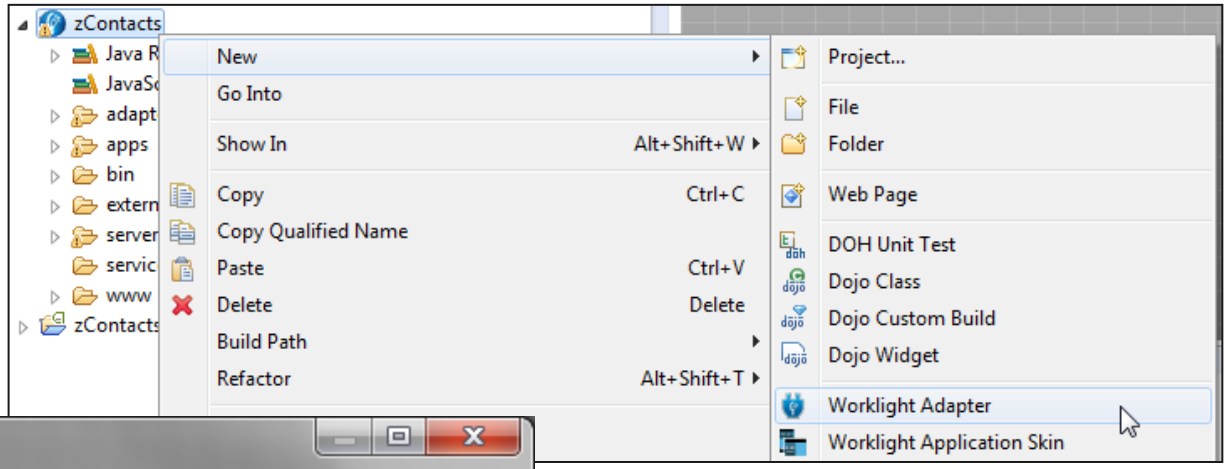


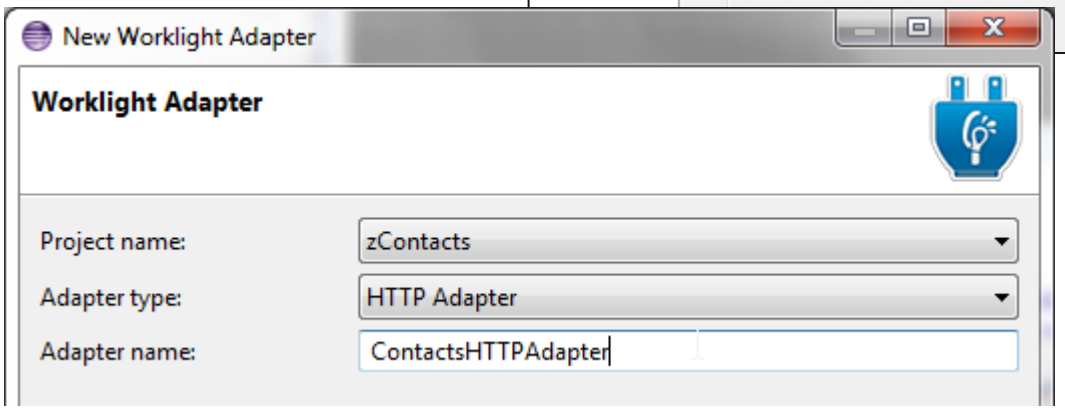
- In the *Worklight Environment* dialog, select **iPhone, Android phones and tablets,** and **Mobile web app** then click **Finish**

Retrieving Data with a Worklight Adapter

- A **Worklight adapter** is hosted on the Worklight server, and interacts with remote data sources, retrieving data or performing actions
- The **Worklight client** runtime provides a simple, common JavaScript interface to invoke the adapter and exchange data from a mobile application

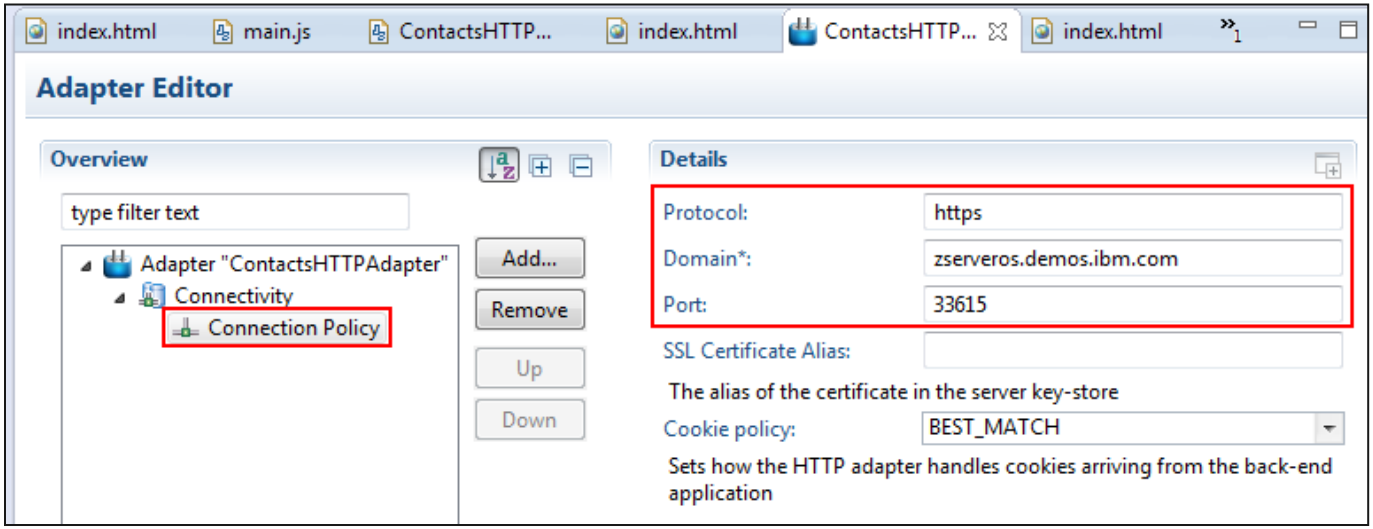
- In the Project Explorer view, right-click on **zContacts** project > **New** > **Worklight Adapter**

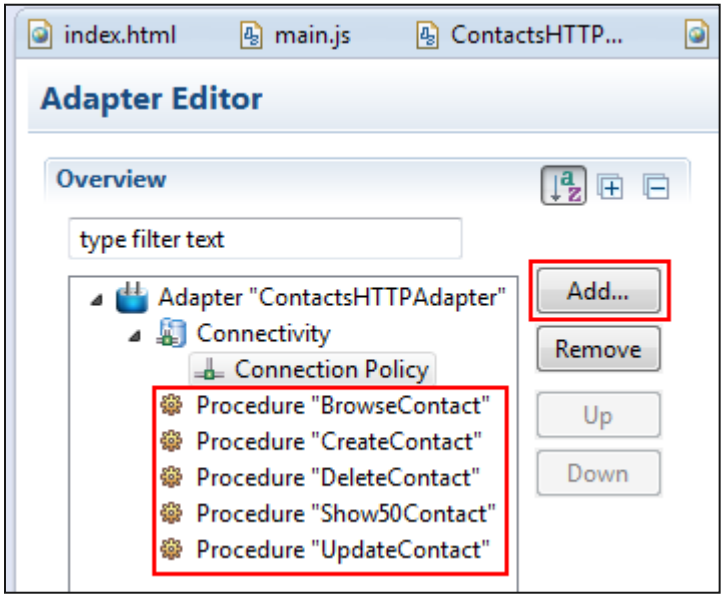




- Select the Adapter type **HTTP Adapter** and enter **ContactsHTTPAdapter** as the name then click the **Finish**

Configure adapter and add procedures

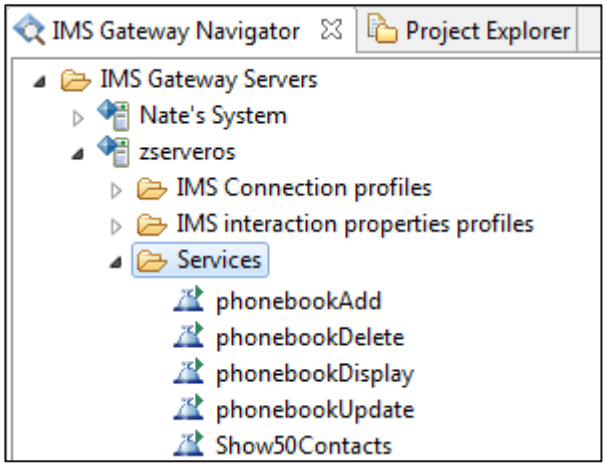




- Select the **Design** tab of the ContactsHTTPAdapter.xml.xml editor > **Connectivity** > **Connection Policy** to edit the HTTP connection details
- Click **Add…** button to add a **Procedure**, add one procedure for every IMS Transaction Service

How to invoke IMS service from adapter procedure??

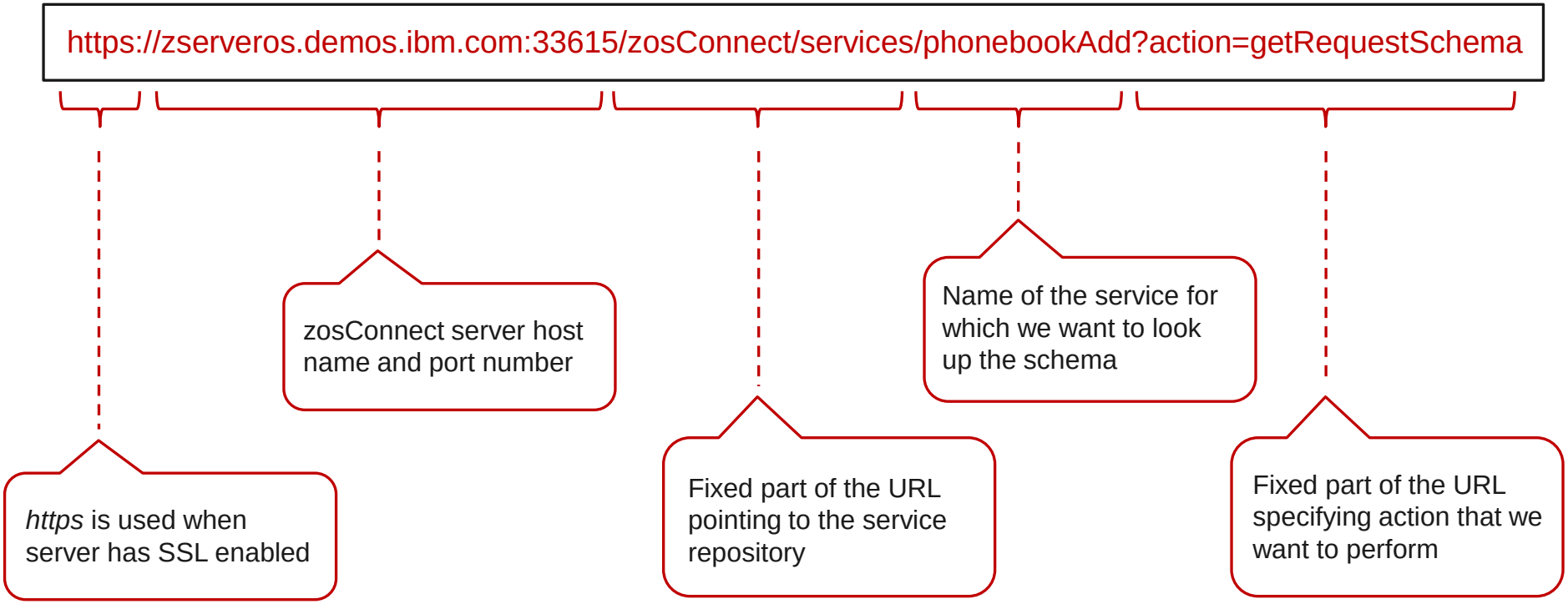
- We have some investigating to do…
- So, we have our services published on zosConnect server:



- We need to find out what JSON request schema looks like for each service…

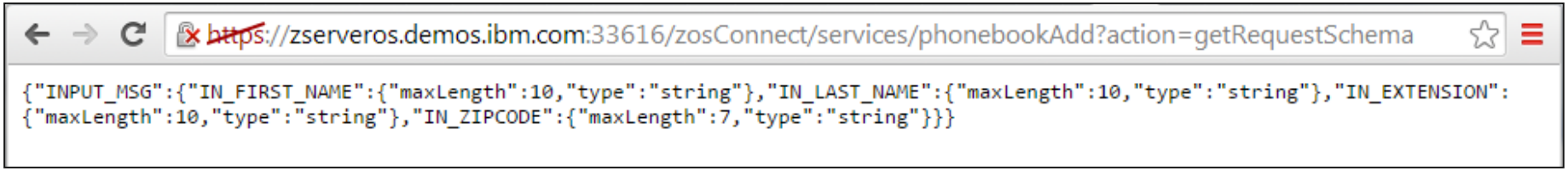
Obtain JSON Request Schema

- We will use good old Web Browser to fetch the schema
  - Open your favorite browser and type in URL that follows this format:



JSON Request Schema to Procedure Payload

- Result should look something like this:



- Formatted schema:

```
{"INPUT_MSG":
  {
    "IN_FIRST_NAME":{"maxLength":10,"type":"string"},
    "IN_LAST_NAME":{"maxLength":10,"type":"string"},
    "IN_EXTENSION":{"maxLength":10,"type":"string"},
    "IN_ZIPCODE":{"maxLength":7,"type":"string"}
  }
}
```

- Let's map this to a JavaScript variable:

```
var payload = {
  "SERVICE_INPUT" : {
    "INPUT_MSG" : {
      "IN_FIRST_NAME" : first,
      "IN_LAST_NAME" : last,
      "IN_EXTENSION" : ext,
      "IN_ZIP_CODE" : zip
    }
  }
};
```

HTTP Adapter Procedure

- Finally, we implement the procedure in the **ContactsHTTPAdapter-impl.js:**

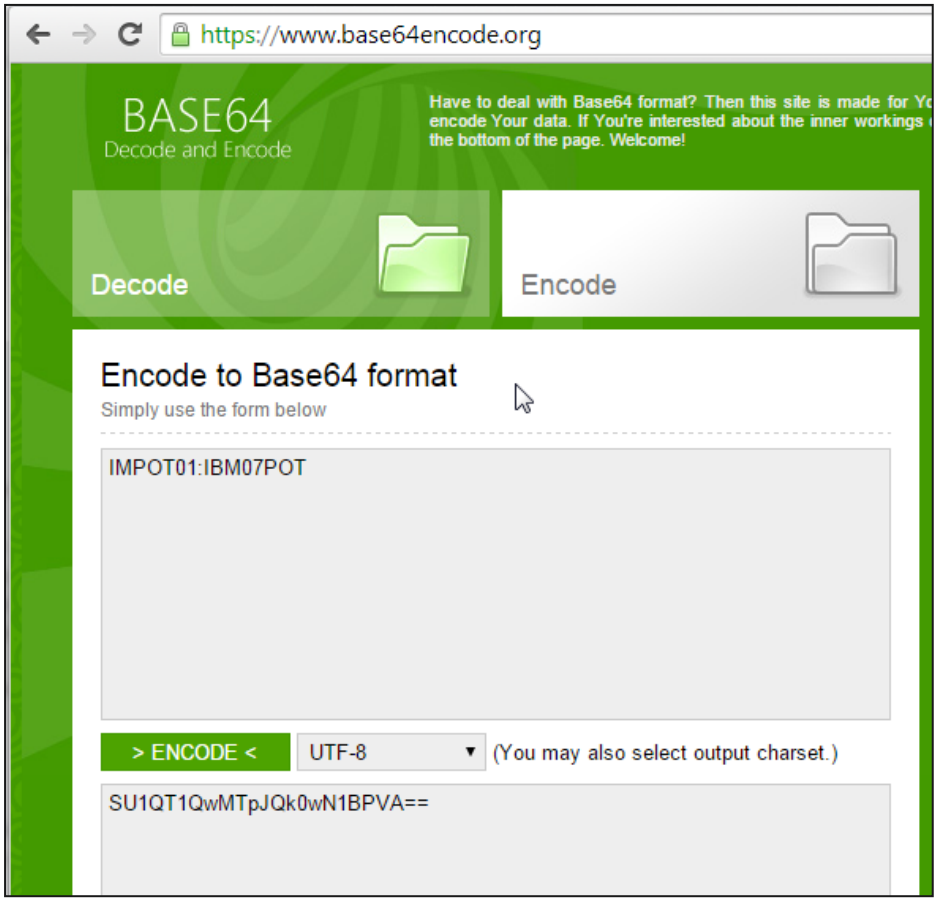
```
function CreateContact(first, last, ext, zip) {

    var payload = {
      "SERVICE_INPUT" : {
        "MOBILE_DEMO_INPUT_MSG" : {
          "IN_FIRST_NAME" : first,
          "IN_LAST_NAME" : last,
          "IN_EXTENSION" : ext,
          "IN_ZIP_CODE" : zip
        }
      }
    };

    var data = JSON.stringify(payload);

    var input = {
          path: '/zosConnect/services/phonebookAdd?action=invoke',
          method: 'PUT',
          headers: {
            'Authorization': 'Basic SU1QT1QwMTpJQk0wN1BPVA==',
            'Content-Type': 'application/json'
          },
          body: {
            'contentType' : 'application/json',
            'content' : data
          }
    };

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

Service invocation syntax for zosConnect

What's this?

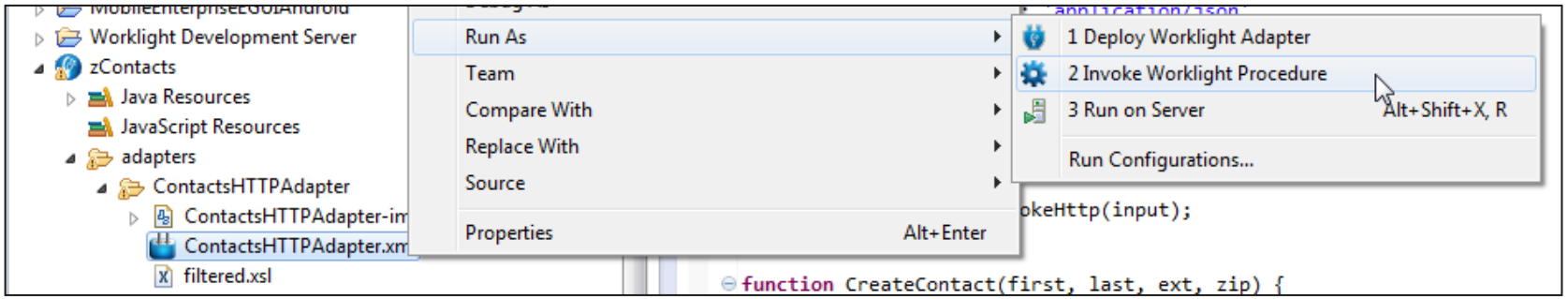
Credentials are encoded in Base64

- Authorization header contains the string Basic , followed by a Base64 encoded string of the username, a colon character, and the password
  - In our example, the credentials IMPOT01:IMS07POT are base64 encoded into the string SU1QT1QwMTpJQk0wN1BPVA==
- Web sites like www.base64encode.org/ can do encoding for you:

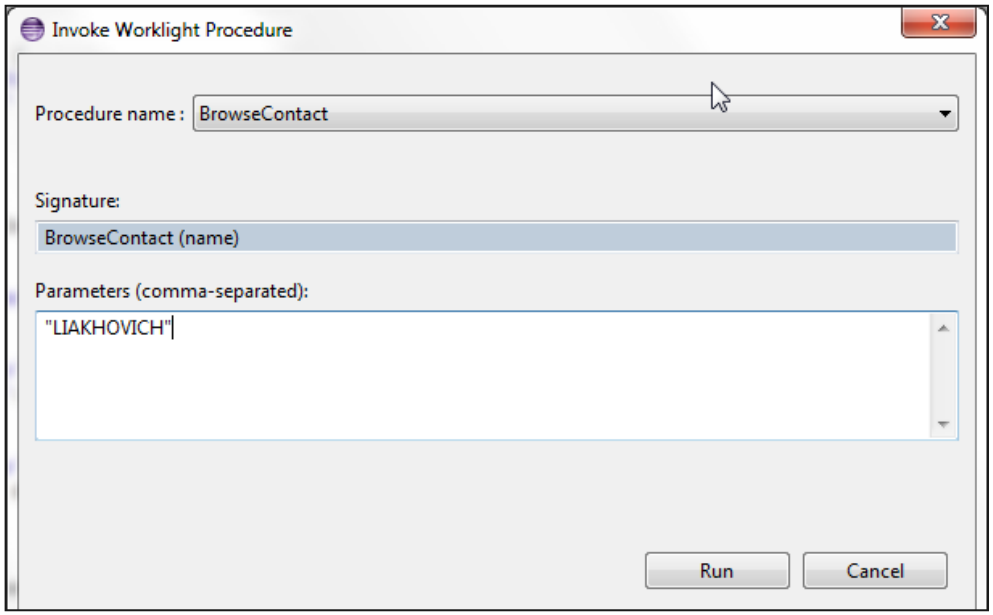


Test Adapter Procedure

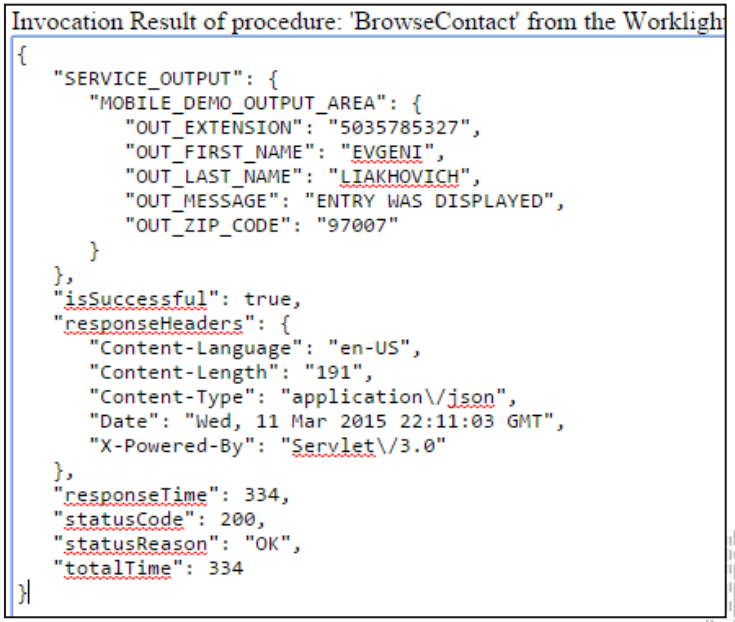
- Right click **ContactsHTTPAdapter.xml**, **Run As > Invoke Worklight Procedure**



- Enter parameter value(s) and click **Run**:

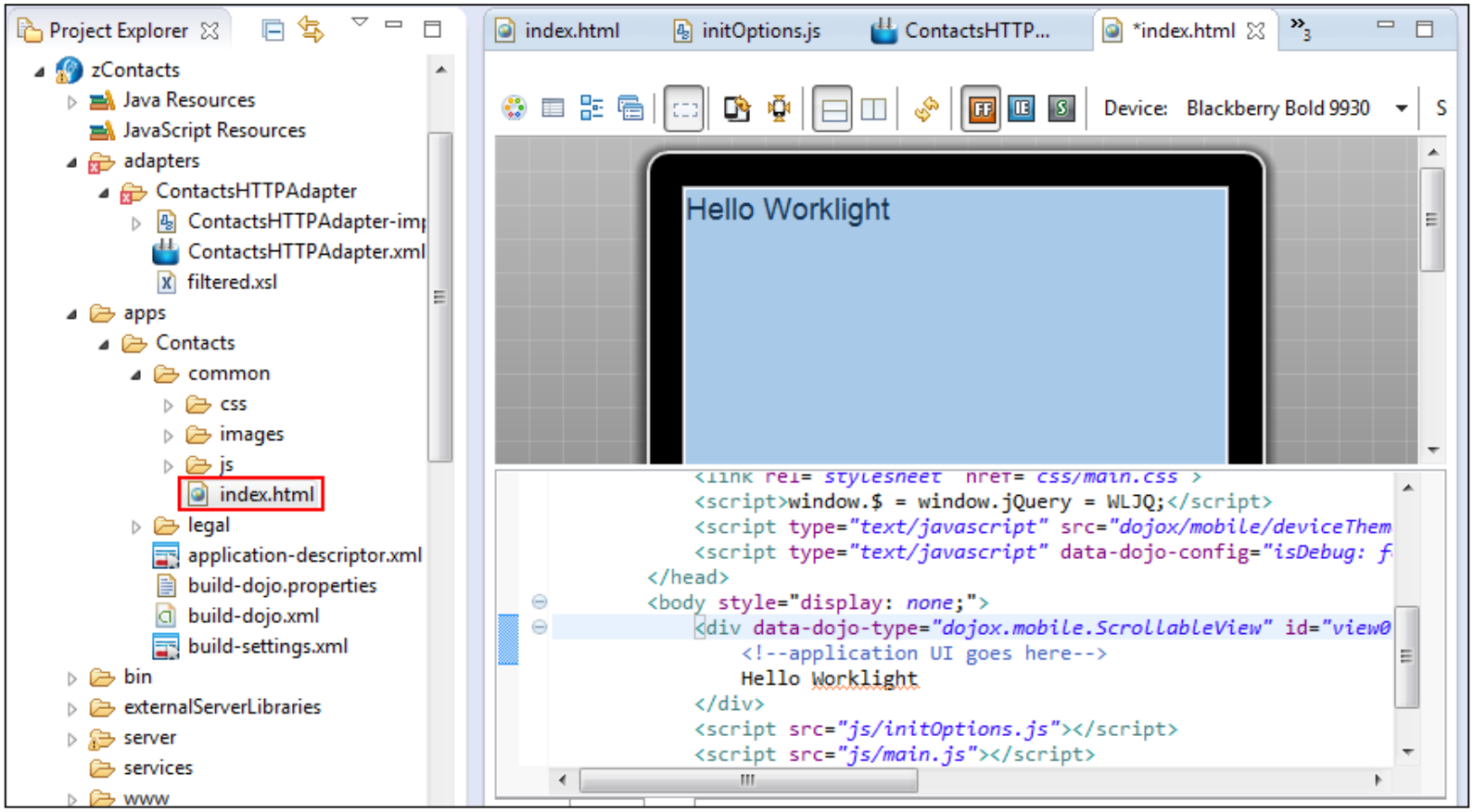


- Your browser should display result:



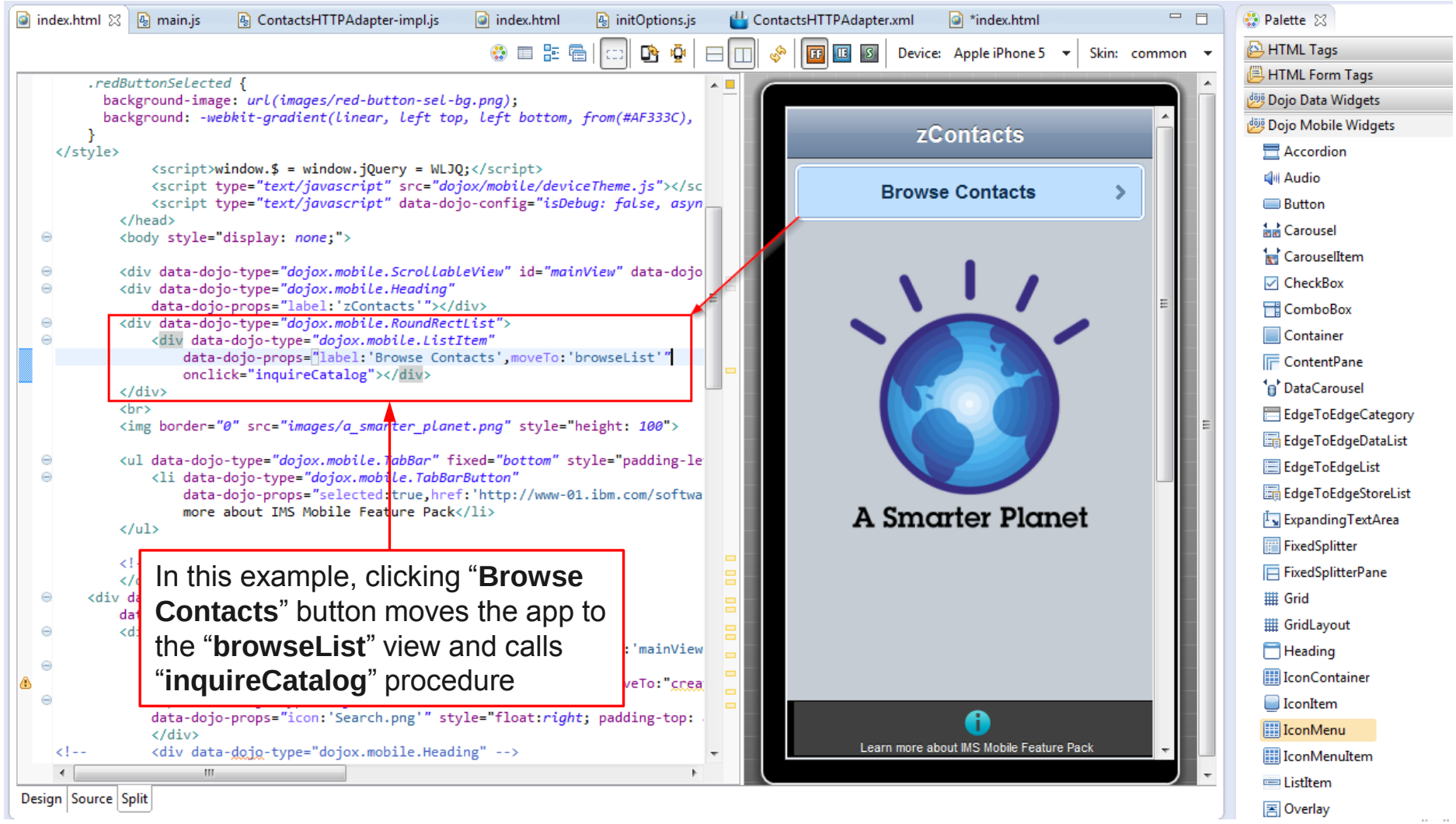
Time to make this look pretty

- Now we have a set of working adapter procedures in the Worklight server. They fetch data from IMS services. Let's create a simple mobile user interface.
- Open **index.html** (follow the path displayed on the screen below):



Design user interface

- Using **Dojo Mobile, jQuery** or similar HTML5 framework, we can visually design user interface for the mobile application



Add logic to the client

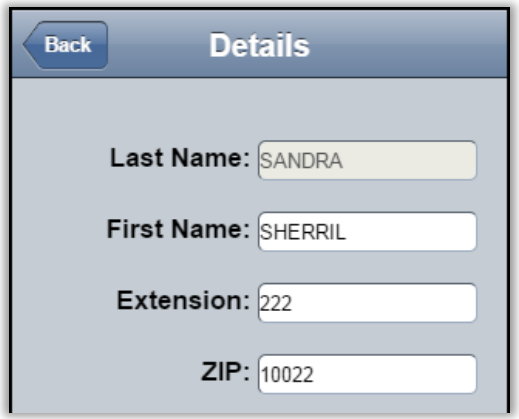
- **main.js** contains client side logic written in JavaScript:

Getting results in JavaScript

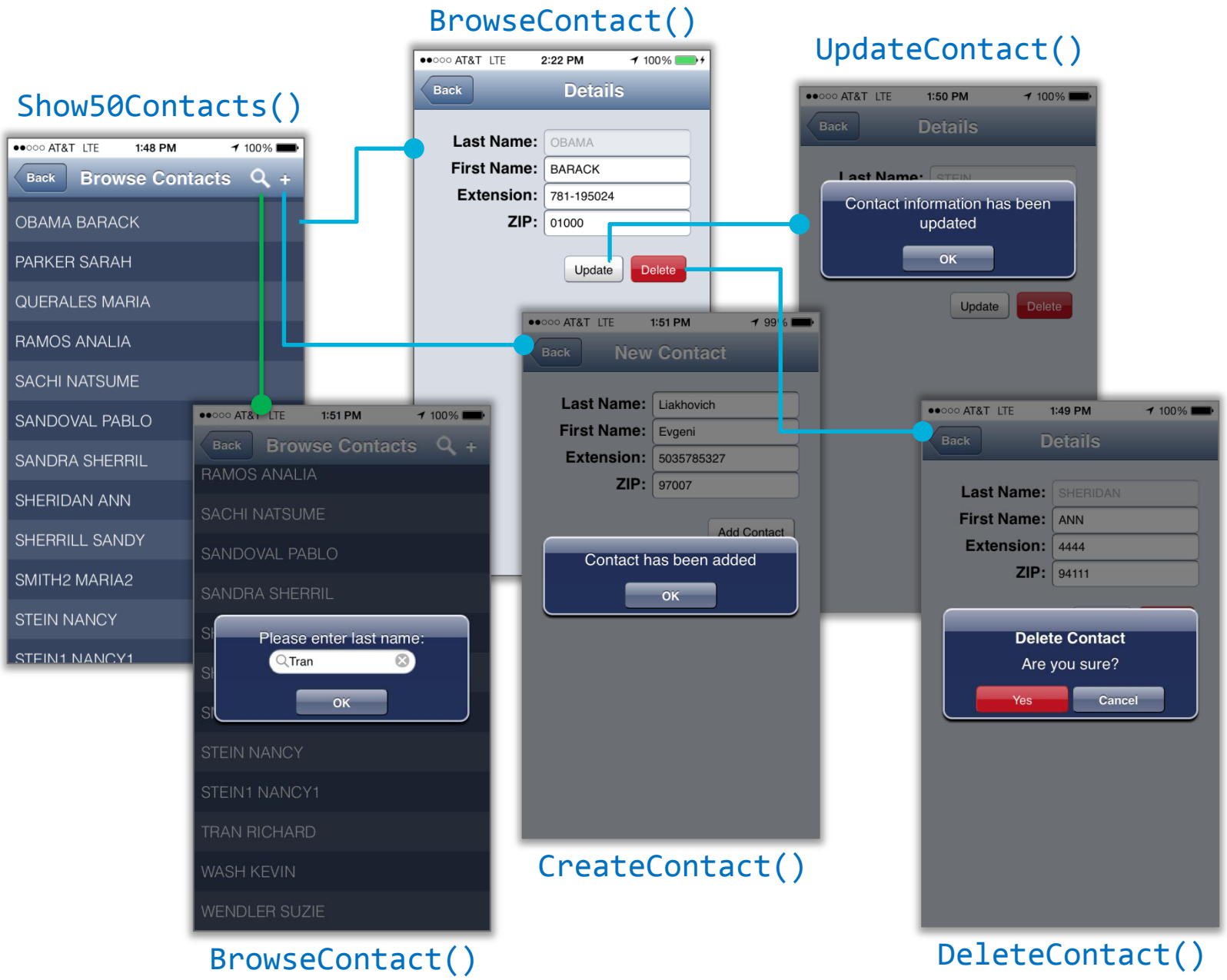
showCatalog():


```
showDetails = function(itemIndex) {

    currentItemIndex = itemIndex;  // Store the current itemIndex

    // Set all of the textBox fields to the selected item's data
    registry.byId("last").set("value",catalogResults.invocationRes
    registry.byId("first").set("value",catalogResults.invocationRe
    registry.byId("ext").set("value",catalogResults.invocationResu
    registry.byId("zip").set("value",catalogResults.invocationResu

    if (catalogResults.invocationResult.SERVICE_OUTPUT.MOBILE_DEMO
        registry.byId("itemDetails").show();
    else
        registry.byId("error_message").show();
};
```

```
registry.byId("last").set("value",catalogResults.
invocationResult.SERVICE_OUTPUT.MOBILE_DEMO_OUTPU
T_AREA.OUT_LAST_NAME); // this is how we get
values from the output JSON
```



zContacts in screenshots



Additional Information

- Mobile Feature Pack documentation:
  http://www-01.ibm.com/support/knowledgecenter/SS9NWR_3.1.0/com.ibm.ims.mobile31.doc/mobile_intro.htm

- MobileFirst Platform – Getting Started:
  https://developer.ibm.com/mobilefirstplatform/documentation/getting-started/

- Technical article on IMS Mobile solution:
  https://www-304.ibm.com/connections/blogs/systemz/entry/ibm_announces_mobile_solution_for_accessing_ims_assets_using_rest_and_json?lang=en_us

Thank you!

Your feedback is important to us!