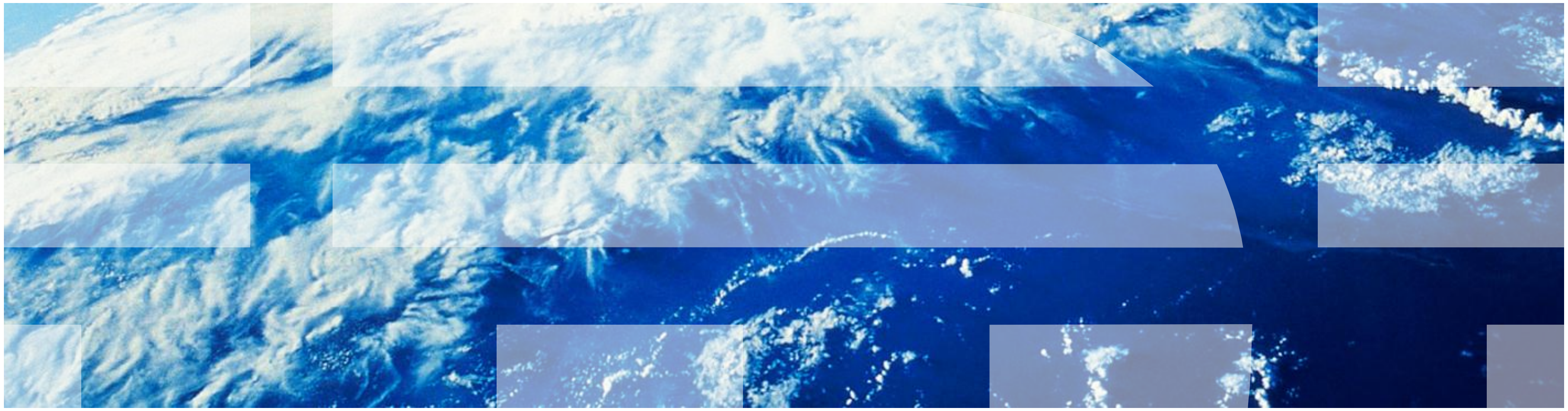
IBM Worklight Foundation V6.2.0
Getting Started

HTTP adapter – Communicating with HTTP back-end systems



20 June 2014

Trademarks

About IBM®

- See http://www.ibm.com/ibm/us/en/

Agenda

- What is it?
- How does it work?
- Creating the adapter
- Using SOAP
  - Creating SOAP-based service request
  - Service request invocation
  - Service discovery
- Back-end service discovery
- Exercise

What is it?

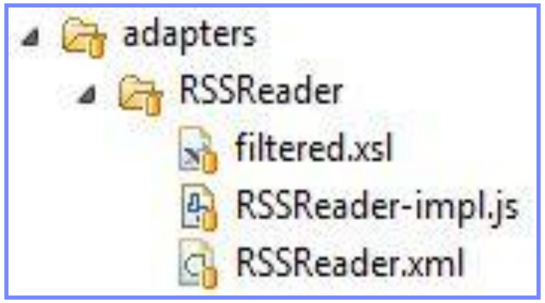
- A Worklight® HTTP adapter:
  - Works with RESTful and SOAP-based services.
  - Can read structured HTTP sources. For example, RSS feeds.
  - Allows sending a GET or POST HTTP request and retrieves data from the response headers and body.
  - Is easily customizable with simple server-side JavaScript™.
  - Enables optional server-side filtering.
  - Retrieved data can be in XML, HTML, JSON, or plain text formats.

Agenda

- What is it?
- How does it work?
- Creating the adapter
- Using SOAP
  - Creating SOAP-based service request
  - Service request invocation
- Back-end service discovery
- Exercise

How does it work?

- The adapter is configured with XML.
- It uses XML to define the adapter properties and procedures.
- It uses JavaScript to create procedures.
- *Optional*: It uses XSL to filter received records and fields.

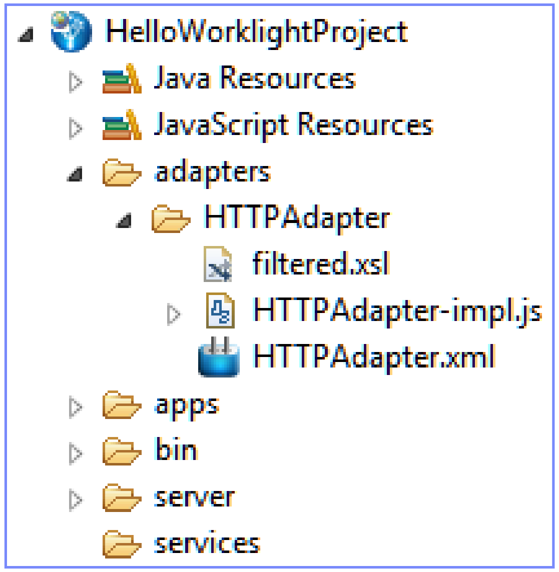


Agenda

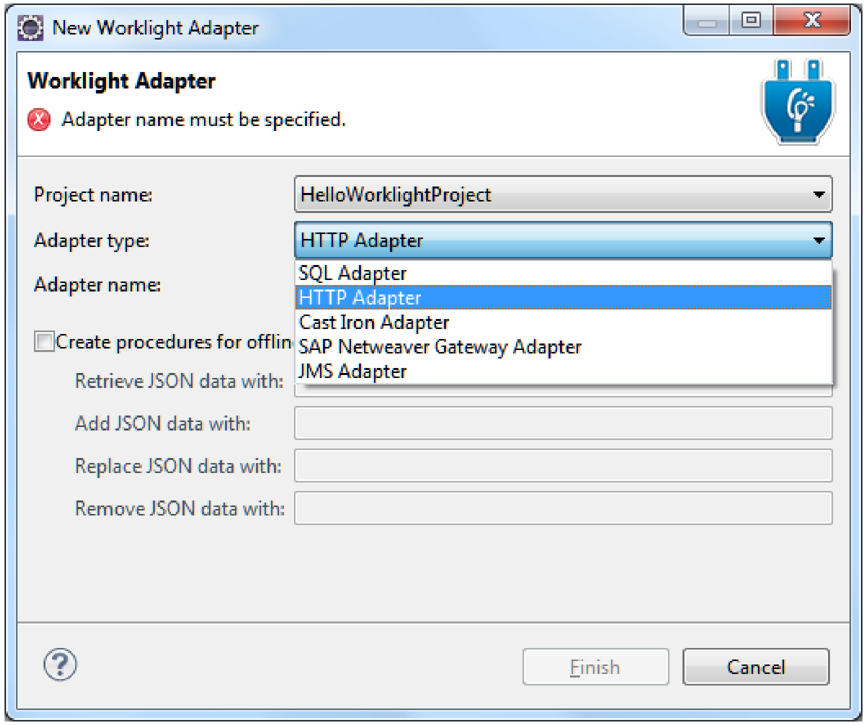
- What is it?
- How does it work?
- Creating the adapter
- Using SOAP
  - Creating SOAP-based service request
  - Service request invocation
- Back-end service discovery
- Exercise

Creating the adapter

- In Worklight Studio, create a Worklight Adapter.
  - Choose the HTTP Adapter type.
  - A standard HTTP adapter structure is created:



```
◢ HelloWorklightProject
    ▷ Java Resources
    ▷ JavaScript Resources
    ◢ adapters
        ◢ HTTPAdapter
              filtered.xsl
            ▷ HTTPAdapter-impl.js
              HTTPAdapter.xml
    ▷ apps
    ▷ bin
    ▷ server
      services
```



New Worklight Adapter

Worklight Adapter

Adapter name must be specified.

Project name:	HelloWorklightProject
Adapter type:	HTTP Adapter
Adapter name:	

Create procedures for offlin

Retrieve JSON data with:

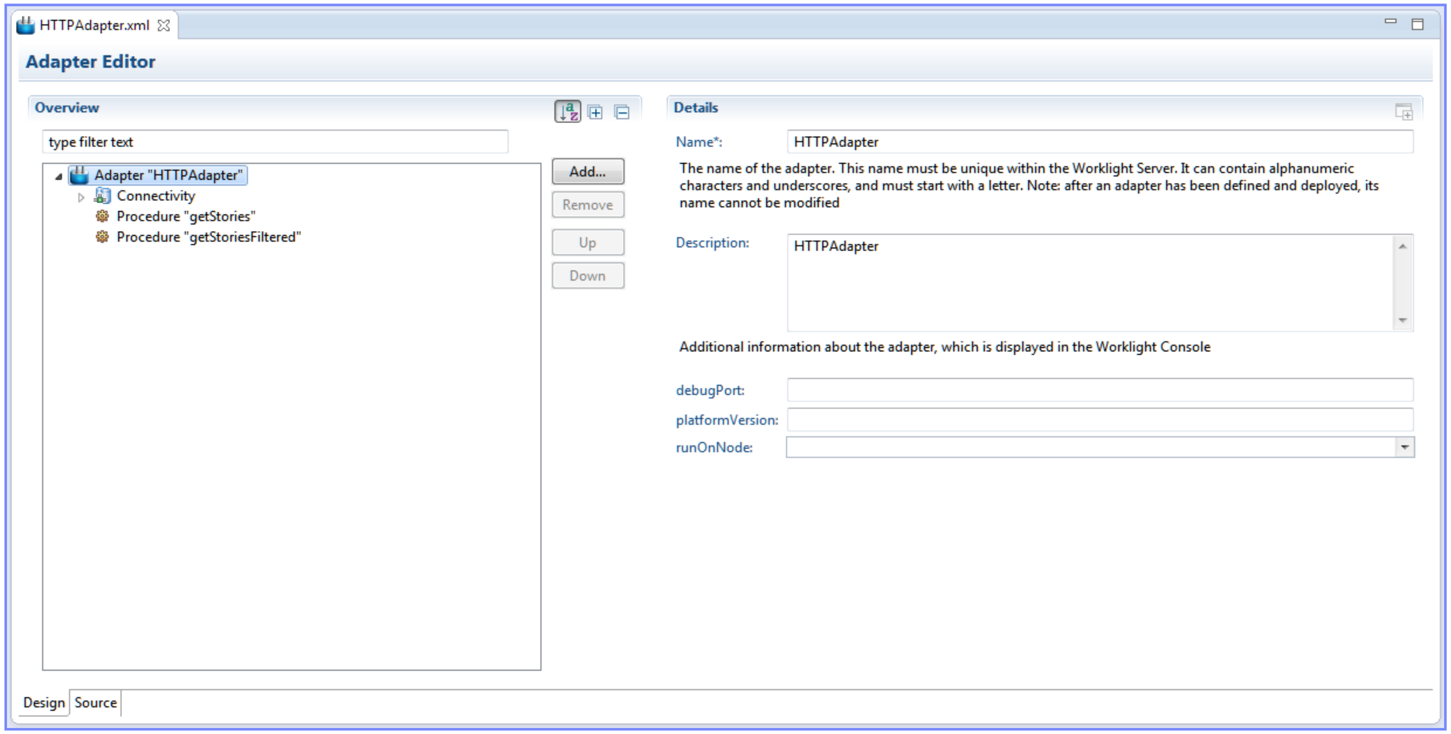
Add JSON data with:

Replace JSON data with:

Remove JSON data with:

Creating the adapter – continued
Adapter XML editor

- Settings and metadata are stored in the adapter XML file.
- You can use either the Design or Source editor to modify the adapter XML file.



Name*: HTTPAdapter

The name of the adapter. This name must be unique within the Worklight Server. It can contain alphanumeric characters and underscores, and must start with a letter. Note: after an adapter has been defined and deployed, its name cannot be modified

Description: HTTPAdapter

Additional information about the adapter, which is displayed in the Worklight Console

debugPort:

platformVersion:

runOnNode:

Creating the adapter – continued
XML file: connectivity settings

- To edit the adapter XML file, you must:
  - Set the protocol to HTTP or HTTPS.
  - Set the HTTP domain to domain part of HTTP URL.
  - Set the TCP Port.

```
<connectivity>
    <connectionPolicy xsi:type="http:HTTPConnectionPolicyType">
        <protocol>http</protocol>
        <domain>rss.cnn.com</domain>
        <port>80</port>
        <!-- Following properties used by adapter's key manager for
        choosing specific certificate from key store
        <sslCertificateAlias></sslCertificateAlias>
        <sslCertificatePassword></sslCertificatePassword>
        -->
    </connectionPolicy>
    <loadConstraints maxConcurrentConnectionsPerNode="2" />
</connectivity>
```

Creating the adapter – continued
XML file: procedures declaration

- Edit the adapter XML file.
- Declare the required procedures below the connectivity element.

```
<connectivity>
  …
  …
  …
</connectivity>

<procedure name="getStories"/>
<procedure name="getStoriesFiltered"/>
```

Creating the adapter – continued
JS file: procedures implementation

- Procedures are implemented in the adapter JavaScript file.
- The service URL is used for procedure invocation.
- Some parts of the URL are constant; for example, http://example.com/. They are declared in the XML file.
- Other parts of the URL can be parameterized; that is, substituted at run time by parameter values that are provided to the Worklight procedure.
- URL parts that can be parameterized are:
  - Path elements.
  - Query string parameters.
  - Fragments.

$$\text{http://example.com}\underbrace{\text{/rest/customers}}_{\text{Path elements}}?\overbrace{\text{custid=12}}^{\text{Query string}}$$

- See the Worklight user documentation for advanced options for adapters, such as cookies, headers, and encoding.

Creating the adapter – continued
JS file: procedures implementation

- The procedure name in the JavaScript file must be the same as in the XML file.
- The required invocation parameters are `method`, `path`, and `returnedContentType`.
- The procedure can be parameterized at run time.

```
<connectivity>
   …
   …
   …
</connectivity>
<procedure name="getStories"/>
<procedure name="getStoriesFiltered"/>
```

The procedure XML file

```
function getStories(interest) {
    path = getPath(interest);
    var input = {
        method : 'get',
        returnedContentType : 'xml',
        path : path
    };
    return WL.Server.invokeHttp(input);
}
```

The procedure JavaScript file

Creating the adapter – continued
JS file: procedures implementation

- To invoke an HTTP request, use the `WL.Server.invokeHttp` method.
- Provide an input parameters object.
- You must specify:
  - The HTTP method: `GET` or `POST`.
  - The returned content type: `XML`, `JSON`, `HTML`, or `plain`.
  - The service path.
  - The query parameters (optional).
  - The request body (optional).
  - The transformation type (optional).
- For a complete list of invocation options, see the Worklight user documentation.

```
function getStories(interest) {
    path = getPath(interest);
    var input = {
        method : 'get',
        returnedContentType : 'xml',
        path : path
    };
    return WL.Server.invokeHttp(input);
}
```

Creating the adapter – continued
XSL transformation filtering

- XSL transformation can be applied to the received data.
- It can be used to filter received data.
- To apply, specify the transformation options in the procedure invocation input parameters.

```
function getStoriesFiltered(interest) {
    path = getPath(interest);

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

Agenda

- What is it?
- How does it work?
- Creating the adapter
- Using SOAP
  - Creating SOAP-based service request
  - Service request invocation
- Back-end service discovery
- Exercise

Creating a SOAP-based service request

- 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 must 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 only SOAP envelopes.

- If you receive error messages for SOAP envelopes, disable the JavaScript validator.
  - Click **Project > Properties > Builders** and clear **JavaScript Validator**.

Creating a SOAP-based service request – continued

- Use JavaScript to create a SOAP Envelope.

```
var request =
    <soap:Envelope
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Body>
            <CelsiusToFahrenheit
            xmlns="http://www.w3schools.com/webservices/">
            <Celsius>{celsiusTemp}</Celsius>
            </CelsiusToFahrenheit>
        </soap:Body>
    </soap:Envelope>;
```

Creating a SOAP-based service request – continued

- It is possible to insert JavaScript code and variables into SOAP XML. It is evaluated at run time.

```
<soap:Envelope
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        </soap:Body
        <messageHeader>
            <originatingIP>
                { WL.Server.configuration["local.IPAddress"]}
            </originatingIP>
            <requestTimestamp>
                { new Date().toLocalString() }
            </requestTimestamp>
         </messageHeader>
     </soap:Body>
</soap:Envelope>;
```

Service request invocation

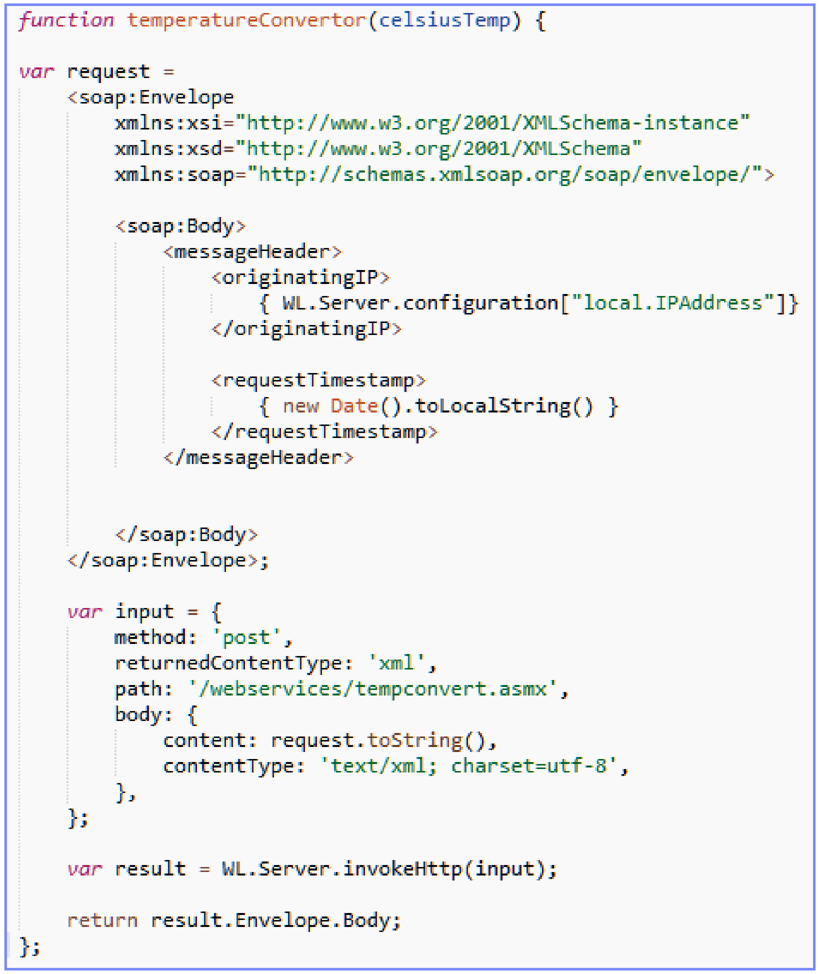
- The `WL.Server.invokeHttp(options)` method is used to invoke a request for a SOAP service.

- The Options object must include the following properties:
  - A method – usually `POST`
  - A returnedContentType – usually `XML`
  - A path – a service path
  - A body:
    - content – SOAP XML as a string
    - contentType

```
var input = {
    method: 'post',
    returnedContentType: 'xml',
    path: '/webservices/tempconvert.asmx',
    body: {
        content: request.toString(),
        contentType: 'text/xml; charset=utf-8',
    },
};
```

Service request invocation – continued

- Full SOAP-based service invocation procedure, as provided in the sample project:



```
function temperatureConvertor(celsiusTemp) {

var request =
    <soap:Envelope
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

        <soap:Body>
            <messageHeader>
                <originatingIP>
                    { WL.Server.configuration["local.IPAddress"]}
                </originatingIP>

                <requestTimestamp>
                    { new Date().toLocalString() }
                </requestTimestamp>
            </messageHeader>


        </soap:Body>
    </soap:Envelope>;

    var input = {
        method: 'post',
        returnedContentType: 'xml',
        path: '/webservices/tempconvert.asmx',
        body: {
            content: request.toString(),
            contentType: 'text/xml; charset=utf-8',
        },
    };

    var result = WL.Server.invokeHttp(input);

    return result.Envelope.Body;
};
```

SOAP envelope

Options

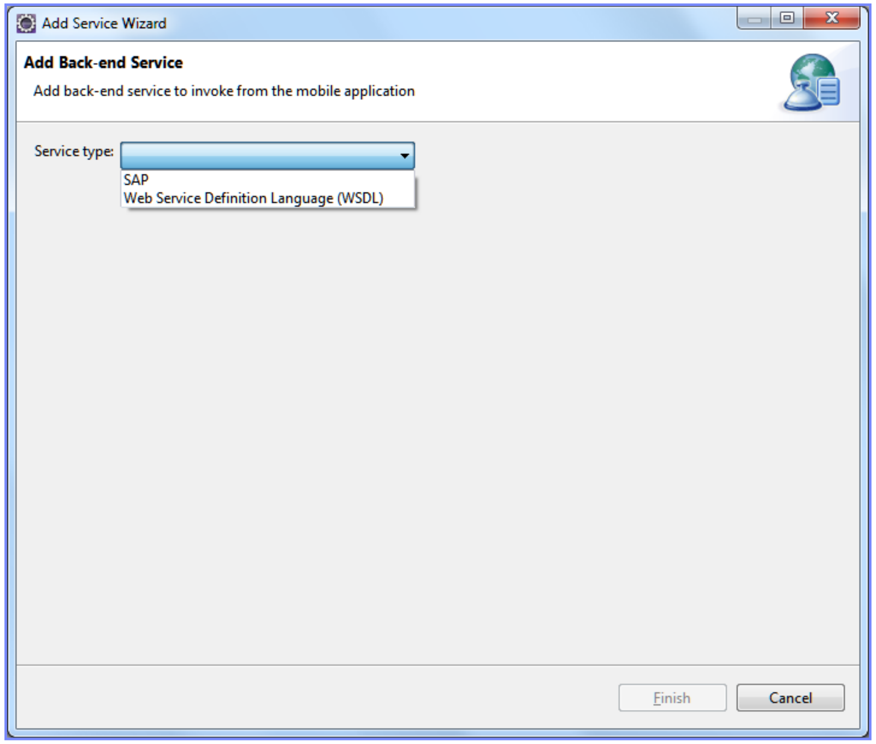
Request invocation

Agenda

- What is it?
- How does it work?
- Creating the adapter
- Using SOAP
  - Creating SOAP-based service request
  - Service request invocation
- Back-end service discovery
  - Exercise

Back-end service discovery

- If you are developing HTTP adapters for SOAP (or SAP) services, you can reduce development time by using the Discover Back-end Services tool to auto-generate the adapter with procedures based on the provided WSDL.
- Right-click on the **services** folder in a Worklight project and choose **Discover back-end services**.
- Select the type of service, **SAP** or **SOAP**:


Add Service Wizard

Add Back-end Service

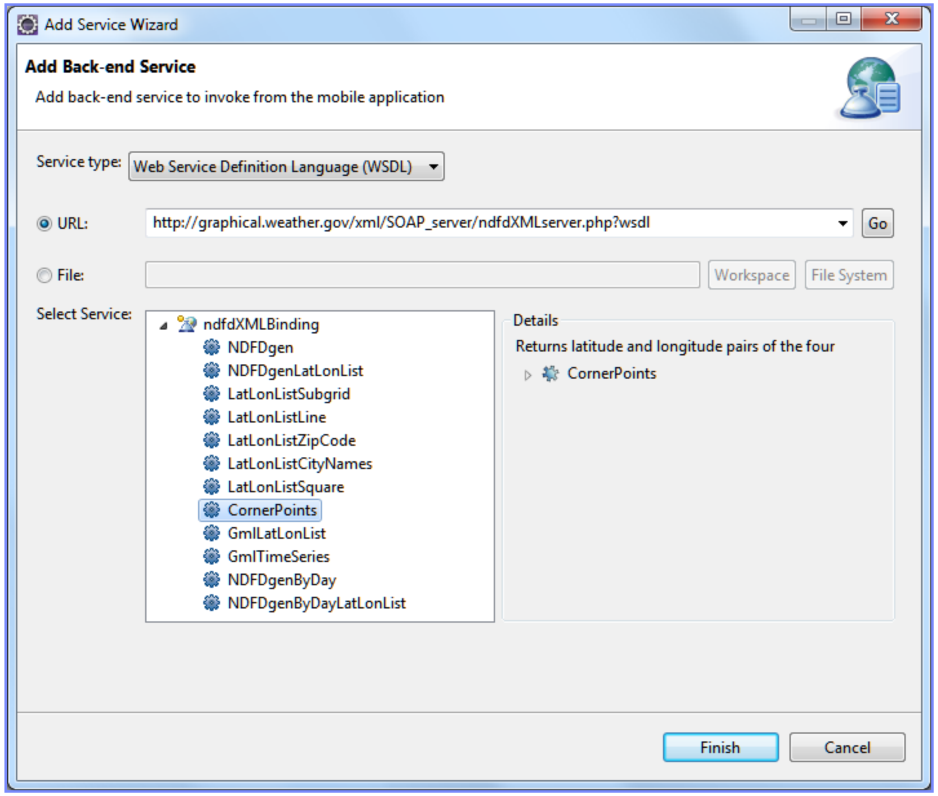
Add back-end service to invoke from the mobile application

Service type:

SAP
Web Service Definition Language (WSDL)

Back-end service discovery – continued

- Add the services location to use and select the service that you want to add to the adapter.
- Repeat for each service that you want to add; it will be added to the same adapter.


Add Service Wizard

Add Back-end Service

Add back-end service to invoke from the mobile application

Service type: Web Service Definition Language (WSDL)

URL: http://graphical.weather.gov/xml/SOAP_server/ndfdXMLserver.php?wsdl

File:

Select Service:
- ndfdXMLBinding
  - NDFDgen
  - NDFDgenLatLonList
  - LatLonListSubgrid
  - LatLonListLine
  - LatLonListZipCode
  - LatLonListCityNames
  - LatLonListSquare
  - CornerPoints
  - GmlLatLonList
  - GmlTimeSeries
  - NDFDgenByDay
  - NDFDgenByDayLatLonList

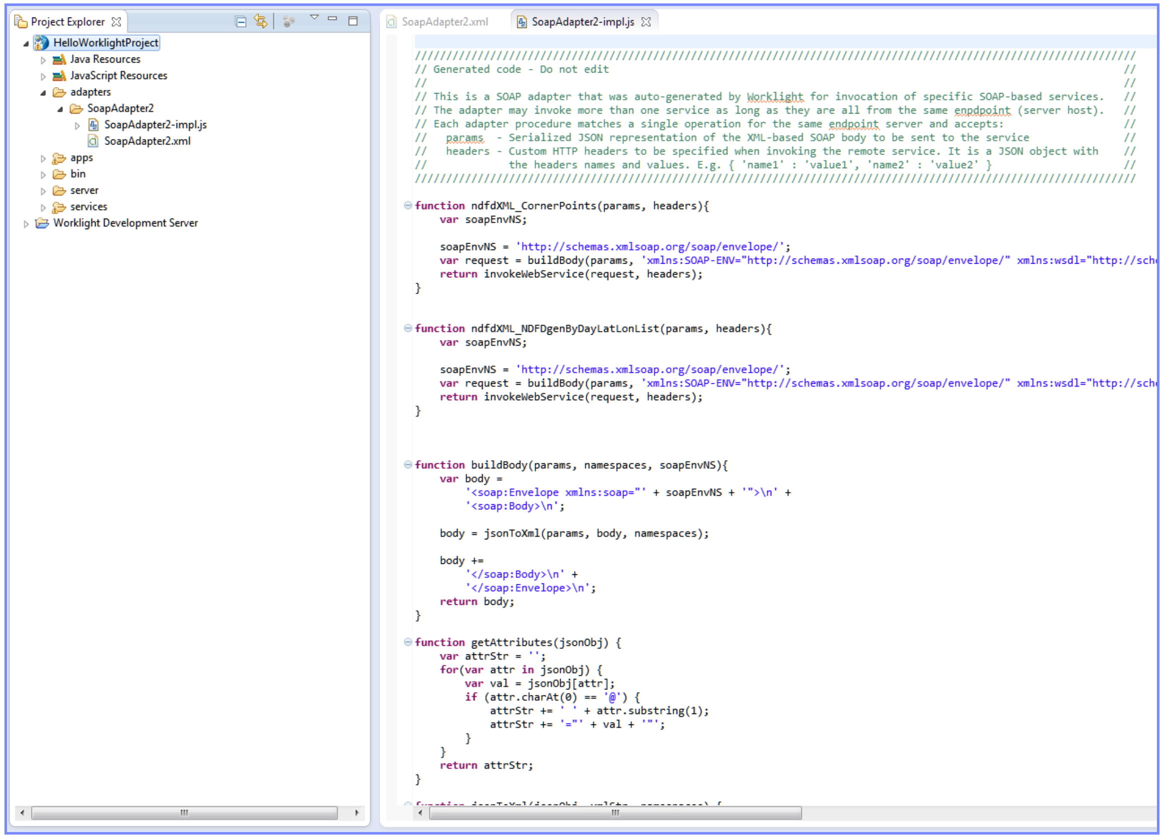
Details

Returns latitude and longitude pairs of the four

CornerPoints

Back-end service discovery – continued

- The end result is an auto-generate adapter:



```
///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Generated code - Do not edit                                                                                      //
//                                                                                                                   //
// This is a SOAP adapter that was auto-generated by Worklight for invocation of specific SOAP-based services.     //
// The adapter may invoke more than one service as long as they are all from the same enpdpoint (server host).     //
// Each adapter procedure matches a single operation for the same endpoint server and accepts:                      //
//   params  - Serialized JSON representation of the XML-based SOAP body to be sent to the service                  //
//   headers - Custom HTTP headers to be specified when invoking the remote service. It is a JSON object with       //
//             the headers names and values. E.g. { 'name1' : 'value1', 'name2' : 'value2' }                         //
///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

function ndfdXML_CornerPoints(params, headers){
    var soapEnvNS;

    soapEnvNS = 'http://schemas.xmlsoap.org/soap/envelope/';
    var request = buildBody(params, 'xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsdl="http://sch
    return invokeWebService(request, headers);
}


function ndfdXML_NDFDgenByDayLatLonList(params, headers){
    var soapEnvNS;

    soapEnvNS = 'http://schemas.xmlsoap.org/soap/envelope/';
    var request = buildBody(params, 'xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsdl="http://sch
    return invokeWebService(request, headers);
}



function buildBody(params, namespaces, soapEnvNS){
    var body =
        '<soap:Envelope xmlns:soap="' + soapEnvNS + '">\n' +
        '<soap:Body>\n';

    body = jsonToXml(params, body, namespaces);

    body +=
        '</soap:Body>\n' +
        '</soap:Envelope>\n';
    return body;
}

function getAttributes(jsonObj) {
    var attrStr = '';
    for(var attr in jsonObj) {
        var val = jsonObj[attr];
        if (attr.charAt(0) == '@') {
            attrStr += ' ' + attr.substring(1);
            attrStr += '="' + val + '"';
        }
    }
    return attrStr;
}
```

Agenda

- What is it?
- How does it work?
- Creating the adapter
- Using SOAP
  - Creating SOAP-based service request
  - Service request invocation
- Back-end service discovery
- Exercise

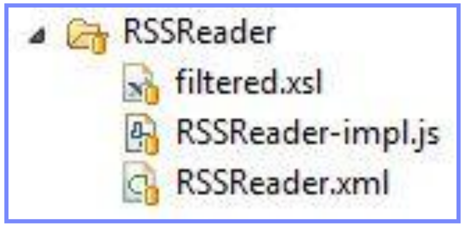
Exercise

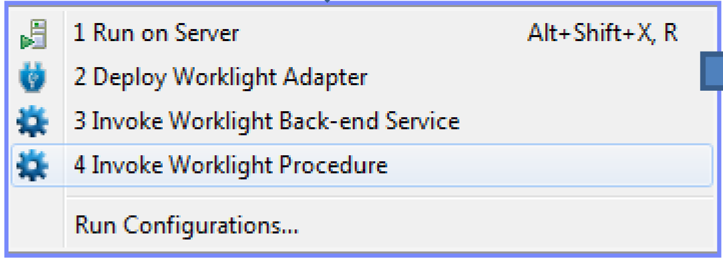
Engadget RSS Reader

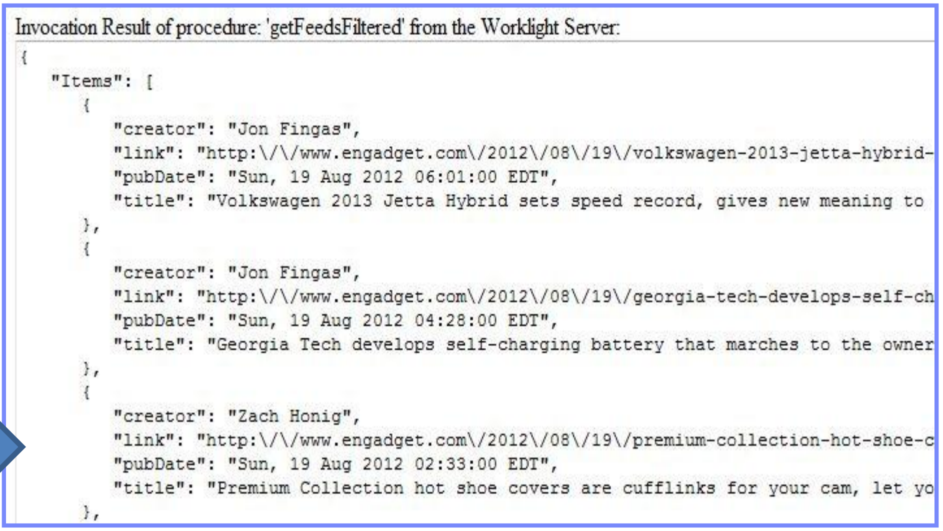
- Create a Worklight HTTP adapter called **RSSReader**.
- Connect to the Engadget RSS feed at http://engadget.com/rss.xml
- Declare and implement a `getFeeds` procedure, which retrieves the RSS feed.
- Declare and implement a `getFeedsFiltered` procedure, which does the same as `getFeeds`, but returns only the `title`, the `creator`, the `pubDate`, and the `link` fields.
- Deploy the adapter and use Worklight Studio to test your procedures as described in this module.

Exercise
Solution

- The sample for this training module can be found on the Getting Started page of the Worklight documentation website at http://www.ibm.com/mobile-docs.




1 Run on Server Alt+Shift+X, R
2 Deploy Worklight Adapter
3 Invoke Worklight Back-end Service
4 Invoke Worklight Procedure
Run Configurations...


Invocation Result of procedure: 'getFeedsFiltered' from the Worklight Server:

```
{
   "Items": [
      {
         "creator": "Jon Fingas",
         "link": "http:\/\/www.engadget.com\/2012\/08\/19\/volkswagen-2013-jetta-hybrid-
         "pubDate": "Sun, 19 Aug 2012 06:01:00 EDT",
         "title": "Volkswagen 2013 Jetta Hybrid sets speed record, gives new meaning to
      },
      {
         "creator": "Jon Fingas",
         "link": "http:\/\/www.engadget.com\/2012\/08\/19\/georgia-tech-develops-self-ch
         "pubDate": "Sun, 19 Aug 2012 04:28:00 EDT",
         "title": "Georgia Tech develops self-charging battery that marches to the owner
      },
      {
         "creator": "Zach Honig",
         "link": "http:\/\/www.engadget.com\/2012\/08\/19\/premium-collection-hot-shoe-c
         "pubDate": "Sun, 19 Aug 2012 02:33:00 EDT",
         "title": "Premium Collection hot shoe covers are cufflinks for your cam, let yo
      },
```

Quiz

Test your knowledge. Answers are in the following slide.

- HTTP adapters can be used to:
  - Work with RESTful services.
  - Work with SOAP services.
  - Issue GET and POST requests.
  - All of the above.
- What format of data can the HTTP adapter retrieve and automatically parse?
  - XML.
  - JSON.
  - Plain text.
  - All of the above.
- Can you use the HTTP adapter with non-standard HTTP ports?
  - You must use port 80 for HTTP and port 443 for HTTPS.
  - You can use any port for HTTP but only port 443 for HTTPS.
  - You must use port 80 for HTTP but can use any port for HTTPS.
  - It is possible to use any port for both HTTP and HTTPS.

Quiz - answers

- HTTP adapters can be used to:
  - Work with RESTful services.
  - Work with SOAP services.
  - Issue GET and POST requests.
  - $\boxed{\text{All of the above.}}$
- What format of data can the HTTP adapter retrieve and automatically parse?
  - XML.
  - JSON.
  - Plain text.
  - $\boxed{\text{All of the above.}}$
- Can you use the HTTP adapter with non-standard HTTP ports?
  - You must use port 80 for HTTP and port 443 for HTTPS.
  - You can use any port for HTTP but only port 443 for HTTPS.
  - You must use port 80 for HTTP but can use any port for HTTPS.
  - $\boxed{\text{It is possible to use any port for both HTTP and HTTPS.}}$

Notices

Support and comments

- For the entire IBM Worklight documentation set, training material and online forums where you can post questions, see the IBM website at:
  - http://www.ibm.com/mobile-docs
- **Support**
  - Software Subscription and Support (also referred to as Software Maintenance) is included with licenses purchased through Passport Advantage and Passport Advantage Express. For additional information about the International Passport Advantage Agreement and the IBM International Passport Advantage Express Agreement, visit the Passport Advantage website at:
    - http://www.ibm.com/software/passportadvantage
  - If you have a Software Subscription and Support in effect, IBM provides you assistance for your routine, short duration installation and usage (how-to) questions, and code-related questions. For additional details, consult your IBM Software Support Handbook at:
    - http://www.ibm.com/support/handbook
- **Comments**
  - We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.
  - For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.
  - When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state.
  - Thank you for your support.
  - Submit your comments in the IBM Worklight Developer Edition support community at:
    - https://www.ibm.com/developerworks/mobile/worklight/connect.html
  - If you would like a response from IBM, please provide the following information:
    - Name
    - Address
    - Company or Organization
    - Phone No.
    - Email address

Thank You