Java HTTP Adapter

Overview

Java adapters provide free reign over connectivity to your backend. It is therefore your responsibility to ensure best practices regarding performance and other implementation details.
This tutorial covers an example of a Java adapter that connects to an RSS feed by using a Java `HttpClient`.

Prerequisite: Make sure to read the Java Adapters (../) tutorial first.

RSSAdapterApplication

`RSSAdapterApplication` extends `MFPJAXRSApplication` and is a good place to trigger any initialization required by your application.

```
@Override
protected void init() throws Exception {
    RSSAdapterResource.init();
    logger.info("Adapter initialized!");
}
```

RSSAdapterResource

`RSSAdapterResource` is where we handle the requests to your adapter.

```
@Path("/")
public class RSSAdapterResource {
}
```

`@Path("/")` means that the resources will be available at the URL `http(s)://host:port/ProjectName/adapters/AdapterName/`.

HTTP Client

RSSAdapterResource

```
private static CloseableHttpClient client;
private static HttpHost host;

public static void init() {
    client = HttpClients.createDefault();
    host = new HttpHost("developer.ibm.com");
}
```

Because every request to your resource will create a new instance of `RSSAdapterResource`, it is important to reuse objects that may impact performance. In this example we made the Http client a `static` object and initialized it in a static `init()` method, which gets called by the `init()` of

`RSSAdapterApplication` as described above.

Procedure resource

RSSAdapterResource

```
@GET
@Produces("application/json")
public void get(@Context HttpServletResponse response, @QueryParam("tag") String ta
        throws ClientProtocolException, IOException, IllegalStateException, SAXExce
    if(tag!=null && !tag.isEmpty()){
        execute(new HttpGet("/mobilefirstplatform/tag/"+ tag +"/feed"), response);
    }
    else{
        execute(new HttpGet("/mobilefirstplatform/feed"), response);
    }

}
```

Our adapter exposes just one resource URL which allows to retrieve the RSS feed from the backend service.

- `@GET` means that this procedure only responds to `HTTP GET` requests.
- `@Produces("application/json")` specifies the Content Type of the response to send back. We chose to send the response as a `JSON` object to make it easier on the client-side.
- `@Context HttpServletResponse response` will be used to write to the response output stream. This enables us more granularity than returning a simple string.
- `@QueryParam("tag")` String tag enables the procedure to receive a parameter. The choice of `QueryParam` means the parameter is to be passed in the query (`/RSSAdapter/?tag=MobileFirst_Platform`). Other options include `@PathParam`, `@HeaderParam`, `@CookieParam`, `@FormParam`, etc.
- `throws ClientProtocolException, ...` means we are forwarding any exception back to the client. The client code is responsible for handling potential exceptions which will be received as `HTTP 500` errors. Another solution (more likely in production code) is to handle exceptions in your server Java code and decide what to send to the client based on the exact error.
- `execute(new HttpGet("/mobilefirstplatform/feed"), response)`. The actual HTTP request to the backend service is handled by another method defined later.

Depending if you pass a `tag` parameter, `execute` will retrieve a different build a different path and retrieve a different RSS file.

execute()

RSSAdapterResource

```
public void execute(HttpUriRequest req, HttpServletResponse resultResponse)
        throws ClientProtocolException, IOException,
        IllegalStateException, SAXException {
    HttpResponse RSSResponse = client.execute(host, req);
    ServletOutputStream os = resultResponse.getOutputStream();
    if (RSSResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK){
        resultResponse.addHeader("Content-Type", "application/json");
        String json = XML.toJson(RSSResponse.getEntity().getContent());
        os.write(json.getBytes(Charset.forName("UTF-8")));

    }else{
        resultResponse.setStatus(RSSResponse.getStatusLine().getStatusCode());
        RSSResponse.getEntity().getContent().close();
        os.write(RSSResponse.getStatusLine().getReasonPhrase().getBytes());
    }
    os.flush();
    os.close();
}
```

- `HttpResponse RSSResponse = client.execute(host, req)`. We use our static HTTP client to execute the HTTP request and store the response.
- `ServletOutputStream os = resultResponse.getOutputStream()`. This is the output stream to write a response to the client.
- `resultResponse.addHeader("Content-Type", "application/json")`. As mentioned before, we chose to send the response as JSON.
- `String json = XML.toJson(RSSResponse.getEntity().getContent())`. We used `org.apache.wink.json4j.utils.XML` to convert the XML RSS to a JSON string.
- `os.write(json.getBytes(Charset.forName("UTF-8")))` the resulting JSON string is written to the output stream.

The output stream is then `flushed` and `closed`.

If `RSSResponse` is not `200 OK`, we write the status code and reason in the response instead.

Results

The adapter should return the RSS feed converted to JSON.

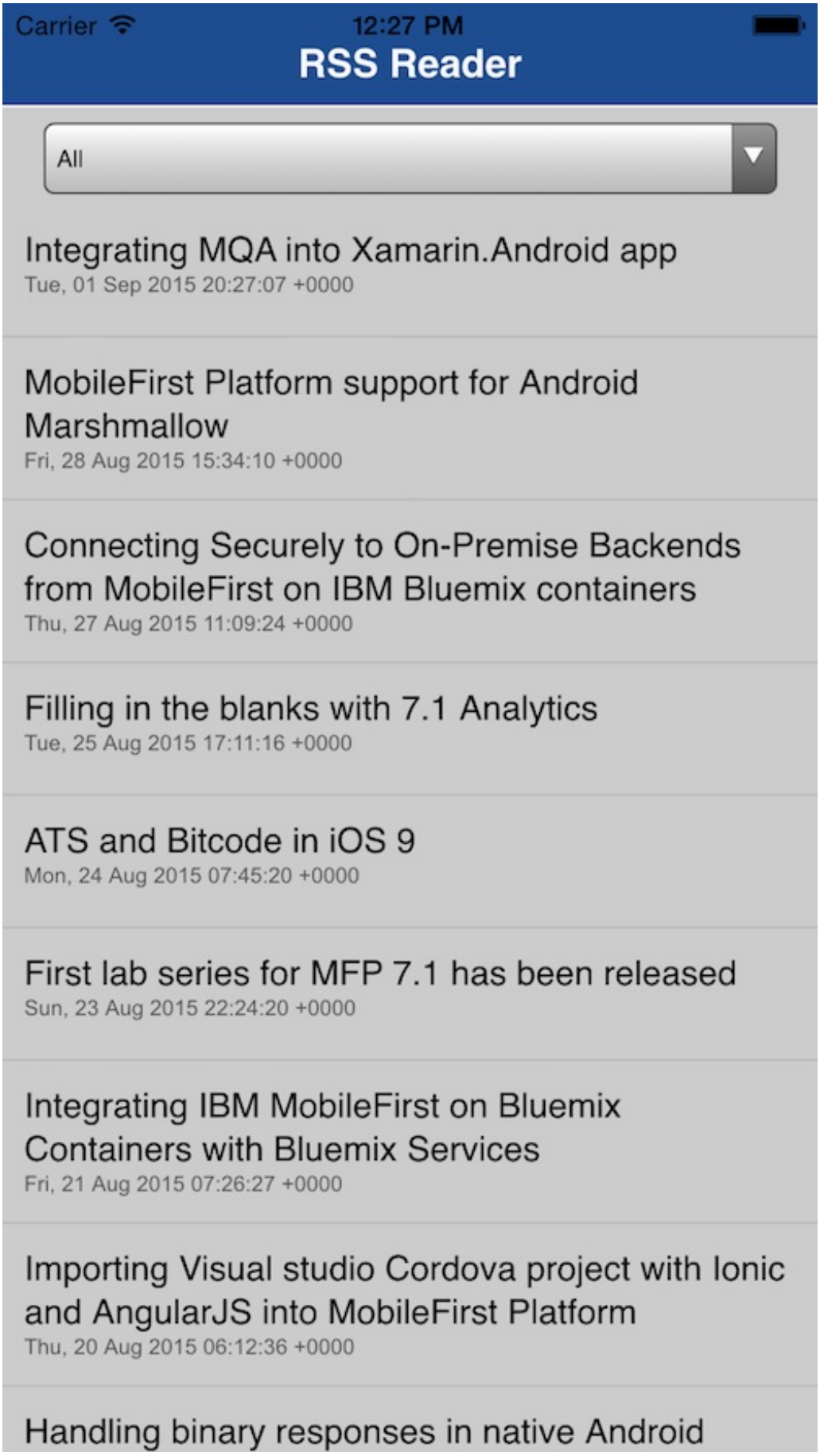

```
{
   "rss": {
      "channel": {
         "description": "Develop, test, manage, and secure your mobile web, native a
         "generator": "http:\/\/wordpress.org\/?v=4.2.4",
         "item": [
            {
               "category": [
                  "Mobile",
                  "android",
                  "Mobile Quality Assurance",
                  "mobile_development",
                  "mobilefirst",
                  "xamarin"
               ],
```

```
            "commentRss": "https:\/\/developer.ibm.com\/mobilefirstplatform\/201
            "comments": [
               "https:\/\/developer.ibm.com\/mobilefirstplatform\/2015\/09\/01\/
               "0"
            ],
            "creator": "Vidyasagar MSC",
            "description": "<p>The post <a rel=\"nofollow\" href=\"https:\/\/dev
            "encoded": "<p>It all startedÂ when I received an email seeking help
            "guid": {
               "content": "https:\/\/developer.ibm.com\/mobilefirstplatform\/?p=
               "isPermaLink": "false"
            },
            "link": "https:\/\/developer.ibm.com\/mobilefirstplatform\/2015\/09\/
            "pubDate": "Tue, 01 Sep 2015 20:27:07 +0000",
            "title": "Integrating MQA into Xamarin.Android app"
         },
         {
            "category": [
               "Uncategorized",
               "MobileFirst_Platform"
            ],
            "commentRss": "https:\/\/developer.ibm.com\/mobilefirstplatform\/201
            "comments": [
               "https:\/\/developer.ibm.com\/mobilefirstplatform\/2015\/08\/19\/
               "0"
            ],
            "creator": "ChethanKumar",
            "description": "<p>The post <a rel=\"nofollow\" href=\"https:\/\/dev
            "encoded": "<p>Contributed By : Chethan Kumar SN (chethankumar.sn@in
            "guid": {
               "content": "https:\/\/developer.ibm.com\/mobilefirstplatform\/?p=
               "isPermaLink": "false"
            },
            "link": "https:\/\/developer.ibm.com\/mobilefirstplatform\/2015\/08\/
            "pubDate": "Wed, 19 Aug 2015 10:36:51 +0000",
            "title": "Try on Bluemix and migrate to on-prem MobileFirst Platform
         }
      ],
      "language": "en-US",
      "lastBuildDate": "Tue, 08 Sep 2015 09:22:53 +0000",
      "link": [
         {
            "href": "https:\/\/developer.ibm.com\/mobilefirstplatform\/feed\/",
            "rel": "self",
            "type": "application\/rss+xml"
         },
         "https:\/\/developer.ibm.com\/mobilefirstplatform"
      ],
      "title": "IBM MobileFirst Platform",
      "updateFrequency": "1",
      "updatePeriod": "hourly"
   },
   "version": "2.0"
}
}
```

Sample

Click to download (https://github.com/MobileFirst-Platform-Developer-Center/JavaAdapters) the MobileFirst project.

The attached sample includes an adapter called `RSSAdapter` and a hybrid application called `RSSReader` to test the adapter inside an application.


Carrier
12:27 PM
RSS Reader
All
Integrating MQA into Xamarin.Android app
Tue, 01 Sep 2015 20:27:07 +0000
MobileFirst Platform support for Android Marshmallow
Fri, 28 Aug 2015 15:34:10 +0000
Connecting Securely to On-Premise Backends from MobileFirst on IBM Bluemix containers
Thu, 27 Aug 2015 11:09:24 +0000
Filling in the blanks with 7.1 Analytics
Tue, 25 Aug 2015 17:11:16 +0000
ATS and Bitcode in iOS 9
Mon, 24 Aug 2015 07:45:20 +0000
First lab series for MFP 7.1 has been released
Sun, 23 Aug 2015 22:24:20 +0000
Integrating IBM MobileFirst on Bluemix Containers with Bluemix Services
Fri, 21 Aug 2015 07:26:27 +0000
Importing Visual studio Cordova project with Ionic and AngularJS into MobileFirst Platform
Thu, 20 Aug 2015 06:12:36 +0000
Handling binary responses in native Android