Integrating with Cloudant by using an adapter

Overview

Cloudant is a NoSQL Database based on CouchDB, which is included with the product as a component called IBM MobileFirst Cloudant Local Data Layer Edition. Cloudant is also available as a stand-alone installed product and as a Database-as-a-Service (DBaaS) on IBM Bluemix and `cloudant.com`.

As described in the Cloudant documentation:

> Documents are JSON objects. Documents are containers for your data, and are the basis of the Cloudant database.
>
> All documents must have two fields: a unique `_id` field, and a `_rev` field. The `_id` field is either created by you, or generated automatically as a UUID by Cloudant. The `_rev` field is a revision number, and is essential to the Cloudant replication protocol. In addition to these two mandatory fields, documents can contain any other content expressed as JSON.

The Cloudant API is documented on the IBM Cloudant Documentation (https://docs.cloudant.com/index.html) site.

You can use IBM MobileFirst Platform adapters to communicate with a remote Cloudant database. This tutorial shows you some examples.

This tutorial assumes that you are comfortable with adapters. See JavaScript HTTP Adapters (../../server-side-development/javascript-adapters/js-http-adapter/) or Java Adapters (../../server-side-development/java-adapter/).

Topics:

- JavaScript HTTP adapter
- Java adapter
- Sample

JavaScript HTTP adapter

The Cloudant API can be accessed as a simple HTTP web service.

Using an HTTP Adapter (../../server-side-development/javascript-adapters/js-http-adapter/), you can connect to the Cloudant HTTP service with the `invokeHttp` method.

Authentication

Cloudant supports several forms of authentication. See the Cloudant documentation about authentication at https://docs.cloudant.com/authentication.html (https://docs.cloudant.com/authentication.html). With a JavaScript HTTP adapter, the easiest way is to use **Basic Authentication**.

In your adapter XML file, specify the `domain` for your Cloudant instance, the `port`, and add an `authentication` element of type `basic`. The framework will use those credentials to generate an `Authorization: Basic` HTTP header.

Note: With Cloudant, you can generate unique API keys to use instead of your real username and password.

```
<connectivity>
   <connectionPolicy xsi:type="http:HTTPConnectionPolicyType">
      <protocol>https</protocol>
      <domain>mysubdomain.cloudant.com</domain>
      <port>443</port>
      <connectionTimeoutInMilliseconds>30000</connectionTimeoutInMilliseconds>
      <socketTimeoutInMilliseconds>30000</socketTimeoutInMilliseconds>
      <authentication>
         <basic/>
         <serverIdentity>
            <username>foribloster5er5engthesce</username>
            <password>c2a8c830f3eertyert0729c786644b0badc1</password>
         </serverIdentity>
      </authentication>
      <maxConcurrentConnectionsPerNode>50</maxConcurrentConnectionsPerNode>
      <!-- Following properties used by adapter's key manager for choosing specific certificate from key st
ore<br />
      <sslCertificateAlias></sslCertificateAlias>
      <sslCertificatePassword></sslCertificatePassword>
      -->
 </connectionPolicy>
</connectivity>
```

Procedures

Your adapter procedures use the `invokeHttp` method to send an HTTP request to one of the URLs that are defined by Cloudant.
For example, you can create a new document by sending a `POST` request to `/{your-database}/` with the body being a JSON representation of the document that you wish to store.

```
function addEntry(entry){
   var input = {
      method : 'post',
      returnedContentType : 'json',
      path : DATABASE_NAME + '/',
      body: {
         contentType : 'application/json',
         content : entry
      }
   };

   var response = WL.Server.invokeHttp(input)
;
   if(!response.id){
    response.isSuccessful = false;
   }
   return response;
}
```

The same idea can be applied to all Cloudant functions. See the Cloudant documentation about documents at https://docs.cloudant.com/document.html (https://docs.cloudant.com/document.html).

Java adapter

Cloudant provides a Java client library (https://github.com/cloudant/java-cloudant) for you to easily use all the features of Cloudant.

During the initialization of your Java adapter, set up a `CloudantClient` instance to work with.
Note: With Cloudant, you can generate unique API keys to use instead of your real username and password.

```
cloudant = new CloudantClient(cloudantDomain,cloudantKey,cloudantPassword);
db = cloudant.database(CLOUDANT_DB, false);
```

Using Plain Old Java Objects (https://en.wikipedia.org/wiki/Plain_Old_Java_Object) and standard Java API for RESTful Web Services (JAX-RS), you can create a new document on Cloudant in just a few lines, by sending a JSON representation of the document in the HTTP request.

```
@POST
@Consumes(MediaType.APPLICATION_JSON)
public Response addEntry(User user){
   if(user!=null && user.isValid()){
    db.save(user);
    return Response.ok().build();
   }
   else{
    return Response.status(418).build();
   }
}
```

Sample

The sample contains two adapters, one in JavaScript and one in Java.

1. Create a database in Cloudant and set the database name in the adapter.
2. Create a user (an API key) and make sure that you provide read and write rights for this user.
3. To connect to Cloudant, update the `CloudantAdapter/server/conf/worklight.properties` file with the domain, username, and password at the bottom of the file.

The sample also contains a hybrid application that works with both the Java and JavaScript adapters.

Download Sample (https://github.com/MobileFirst-Platform-Developer-Center/CloudantAdapter)