introduction to using Watson Services with Java on Bluemix

Patrick Mueller **@pmuellr**, **muellerware.org**
developer advocate for IBM's Bluemix PaaS

http://pmuellr.github.io/slides/2015/02-java-intro-with-watson
http://pmuellr.github.io/slides/ (all slides)

what is Bluemix

- Platform-as-a-Service aka PaaS aka web app hosting platform
- you provide the app, Bluemix hosts the app

deployment process

- you push your application code to Bluemix
- Bluemix stages your app; finds runtimes, libraries your app uses
- Bluemix builds a "droplet"; archive of app code, runtimes, libraries
- Bluemix provisions VM to run the droplet, unpacks droplet, starts it

references

- Bluemix console
- Bluemix documentation
- Eclipse tools for Bluemix

Bluemix Answers

- https://developer.ibm.com/answers/smartspace/bluemix/
- open to the public
- thousands of questions already asked and answered
- please ask any questions here, but no IBM Confidential or IBM internal questions

articles / movies

- Getting Started with IBM Bluemix and DevOps Services using Java
- Developing IBM Bluemix applications in Java with Eclipse and DevOps Services
- Work locally with IBM DevOps Services projects and Git source control
- Video: Develop and manage Java Apps with IBM Bluemix and DevOps Services

sign up for Bluemix and Dev Ops Services

- for Bluemix, register here (click on **SIGN UP**):

  https://bluemix.net

- for Dev Ops Services, register here:

  https://jazz.net/action/register

  use the same userid/password as for Bluemix

supported programming languages

- just about anything
- 1st class support for Java (using Liberty) and node.js
- community support for PHP, Ruby, Python, others

supported programming languages - node.js

- http://node-stuff.mybluemix.net/how-to
  - lists pre-reqs to install
  - sample app with instructions to deploy yourself
- Watson User Modeling sample for node.js available here

supported programming languages - Java

pre-reqs for Java development

- install Eclipse (Luna)
- install Bluemix tools for Eclipse
- install WebSphere Software (in Eclipse Help menu)
- install `cf` command-line tool (optional, but you will probably want it)

supported development environments

- command-line; using text editors or IDEs, and the **`cf`** command-line tool
- Eclipse using **`cf`** command-line tool, or Bluemix plugin for Eclipse
- Dev Ops Services - http://hub.jazz.net; edit, build, deploy all from the web

Watson User Modeling sample for Java

- code / instructions, available here:

https://hub.jazz.net/project/pmuellr/um-java/overview

- (live demo of deploying app using IDS)

Watson User Modeling sample for Java - using Eclipse

- import um-java project using Eclipse git
- deployment options
  - commit to git, let IDS redeploy to Bluemix
  - deploy directly using Eclipse for Bluemix tools

other goodies for Bluemix using Eclipse

- incremental publish
- remote debug

Java code examples

using Watson services from Java

- bind service to app in Bluemix console
- use **`VCAP_SERVICES`** environment variable to get URL and credentials for service
- make REST calls to service

example VCAP_SERVICES

```
{
  "user_modeling": [
    {
      "name": "watson-user-modeling",
      "label": "user_modeling",
      "plan": "user_modeling_free_plan",
      "credentials": {
        "url": "https://gateway.watsonplatform.net/systemu/service/",
        "username": "<secret username>",
        "password": "<secret password>"
      }
    }
  ]
}
```

parsing `VCAP_SERVICES` in Java - libraries

`com.ibm.websphere.appserver.api.json_1.0.2.jar`

- available for local usage in **`um-java`** sample, in **`um-java/lib`** directory
- provided automatically when deploying to Bluemix

parsing VCAP_SERVICES in Java - code

```
import com.ibm.json.java.JSONArray;
import com.ibm.json.java.JSONObject;

JSONObject getVcapServices() {
  String envServices = System.getenv("VCAP_SERVICES");

  if (envServices == null) return null;
  JSONObject sysEnv = null;
  try {
     sysEnv = JSONObject.parse(envServices);
  }
  catch (IOException e) {
    String message = "Error parsing VCAP_SERVICES: ";
    logger.log(Level.SEVERE, message + e.getMessage(), e);
  }
  return sysEnv;
}
```

getting service credentials from parsed VCAP_SERVICES in Java

```
// serviceName = "user_modeling";
private void processVCAP_Services(serviceName) {
  JSONObject sysEnv = getVcapServices();
  if (sysEnv == null) return;

  for (Object key : sysEnv.keySet()) {
    String keyString = (String) key;
    if (keyString.startsWith(serviceName)) {
      JSONArray services = (JSONArray)sysEnv.get(key);
      JSONObject service = (JSONObject)services.get(0);
      JSONObject credentials;

      credentials = (JSONObject)service.get("credentials");
      baseURL  = (String)credentials.get("url");
      username = (String)credentials.get("username");
      password = (String)credentials.get("password");
    }
  }
}
```

accessing a RESTy service in Java - libraries

- use Apache HttpComponents for RESTy libraries
- provided with Bluemix libraries for Eclipse
- provided automatically when deploying to Bluemix

issuing REST request in Java

```
Executor ex = Executor.newInstance().auth(username, password);
URI profileURI = new URI(baseURL + "api/v2/profile").normalize();

Request profileRequest = Request.Post(profileURI)
  .addHeader("Accept", "application/json")
  .bodyString(content.toString(), ContentType.APPLICATION_JSON);
String profileString = ex.execute(profileRequest)
  .handleResponse(new ResponseHandler<String>() {
    @Override
    public String handleResponse(HttpResponse r)
        throws ClientProtocolException, IOException {
      int statusCode = r.getStatusLine().getStatusCode();
      if (statusCode != HttpStatus.SC_OK) {
        req.setAttribute("error", handleError(r));
        return null;
      }
      return EntityUtils.toString(r.getEntity());
    }
});
```

input and output of REST request

- in previous example, **`content`** was the input, and **`profileString`** was the output, **`baseURL`**, **`username`**, **`password`** came from **`VCAP_SERVICES`**
- input and output will often be JSON format
- parse like **`VCAP_SERVICES`** example
- JSON utilities can also be used to generate correctly formatted JSON for input, from Java data structures

overview of Watson services

Watson - Concept Expansion

Maps euphemisms or colloquial terms to more commonly understood phrases

- input: starting point word, a few terms that are examples of that word, and a data set to analyze
- output: a ranked list of terms with contextually similarity to the starting word
- data sets: periodically updated random tweets, Medical transcript samples from MTSamples

Watson - Language Identification

Identifies the language in which text is written

- supports: Arabic; Chinese (Simplified); Chinese (Traditional); Cyrillic; Danish; Dutch; English; Farsi; Finnish; French; German; Greek; Hebrew; Hindi; Icelandic; Italian; Japanese; Korean; Norwegian (Bokmal); Norwegian (Nynorsk); Portuguese; Spanish; Swedish; Turkish; Urdu.
- input: text
- output: 5-letter ISO language code; eg, "**`en-US`**"

Watson - Machine Translation

Translate text from one language to another

- supports: English, Brazilian Portuguese, Spanish, French and Arabic
- input: text to be translated
- output: translated text

Watson - Message Resonance

Communicate with people with a style and words that suits them

- input: term to evaluate and community to measure against
- output: score ranking of how well term will be received by community
- communities: **"cloud"** twitter messages or **"big data"** twitter messages

Watson - Question and Answer

Direct responses to user inquiries fueled by primary document sources

- input: questions and which data set to query
- output: multiple answers with confidence scores and links to evidence
- data sets: **Healthcare data** (including Healthfinder and CDC Health Topics) or **Travel data** (including Wikivoyage, TSA, and CDC Travel)

Watson - Relationship Extraction

Intelligently finds relationships between sentence components (nouns, verbs, subjects, objects, etc.)

- input: text news articles
- output: entities from text and relationships in XML data structure
- input is processed over a domain optimized for news articles

Watson - User Modeling

Improves understanding of people's preferences to help engage users on their own terms

- input: text from an individual
- output: tree of social characteristcs in JSON and visualizations using HTML and SVG
- input should be at least 1000 words of text written by one individual

Watson - Visualization Rendering

Graphical representations of data analysis for easier understanding

- This service is an SDK that can be used to visualize any numeric data
- aka RAVE
- supports iOS, Android, Java and JavaScript

Watson - more services!

- new Watson services will be released over time
- check the Watson Developer Cloud for updates

fin