

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/2015/02-java-intro-with-watson/java-intro-with-watson.pdf>

<http://pmuellr.github.io/slides/> (all of Patrick's slides)

agenda

- Bluemix intro
- deploy a sample app
- code snippets
- Watson services

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
- IBMers: do not ask questions containing sensitive IBM internal information

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 DevOps Services

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

<https://bluemix.net>

- for DevOps Services, register here, after registering at Bluemix (click on **LOG IN**):

<https://hub.jazz.net/>

use the same userid/password as for Bluemix

sign up for Bluemix and DevOps Services (more)

- IBMers: use your IBM email address when registering
- if you have problems registering, send an email to id@bluemix.net

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
- DevOps 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>

- a live version of this application here:

<http://watson-um-demo.mybluemix.net/>

- (show deployment of the app using DevOps Services)

Watson User Modeling sample for Java - using Eclipse

- import um-java project using Eclipse git
- deployment options
 - commit to git, let DevOps Services 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 vcap = System.getenv("VCAP_SERVICES");

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

getting service credentials from parsed VCAP_SERVICES in Java

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

    for (Object labelTest : sysEnv.keySet()) {
        String labelString = (String) labelTest;
        if (labelString.startsWith(label)) {
            JSONArray services = (JSONArray) sysEnv.get(labelTest);
            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 - Concept Insights

Explores information based on the concepts behind your input

- input: content and queries, in text and HTML
- output: a list of content that is relevant to your queries
- dataset: also uses English Wikipedia

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
- dataset: domain optimized for news articles

Watson - Speech to Text

Transcribes English speech to text with low latency

- input: streamed or recorded audio
- output: text transcriptions of the recognized words
- dataset: intelligible english speech

Watson - Text to Speech

Synthesizes natural-sounding speech from English or Spanish text

- input: English or Spanish text
- output: synthesized audio based on the input text
- dataset: English or Spanish text

Watson - Tradeoff Analytics

Helps users make better choices to best meet multiple conflicting goals

- input: decision problem in a JSON document
- output: problem and its resolution in a JSON document
- dataset: self-contained in input decision problem

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 characteristics in JSON and visualizations using HTML and SVG
- input should be at least 1000 words of text written by one individual

Watson - Visual Recognition

Analyzes the visual content of images and video frames to understand the content directly

- input: JPEG images
- output: a set of labels and likelihood scores
- dataset: large number of classified pictures

Watson - more information

- [Watson Developer Cloud](#) web site
- one stop shopping for
 - getting started information
 - reference documentation
 - app gallery
 - sample code

fin