Hello World Lab

Introduction

This lab walks you through the development of a simple application to call and process a Watson service. We make use of the Relationship Extraction service available through Bluemix.

You are given a skeleton Eclipse project call HelloWorld that does nothing except display a web page that we will be using for testing. The skeleton app is a Java web app that uses the Liberty Profile as the Bluemix target environment.

Resources

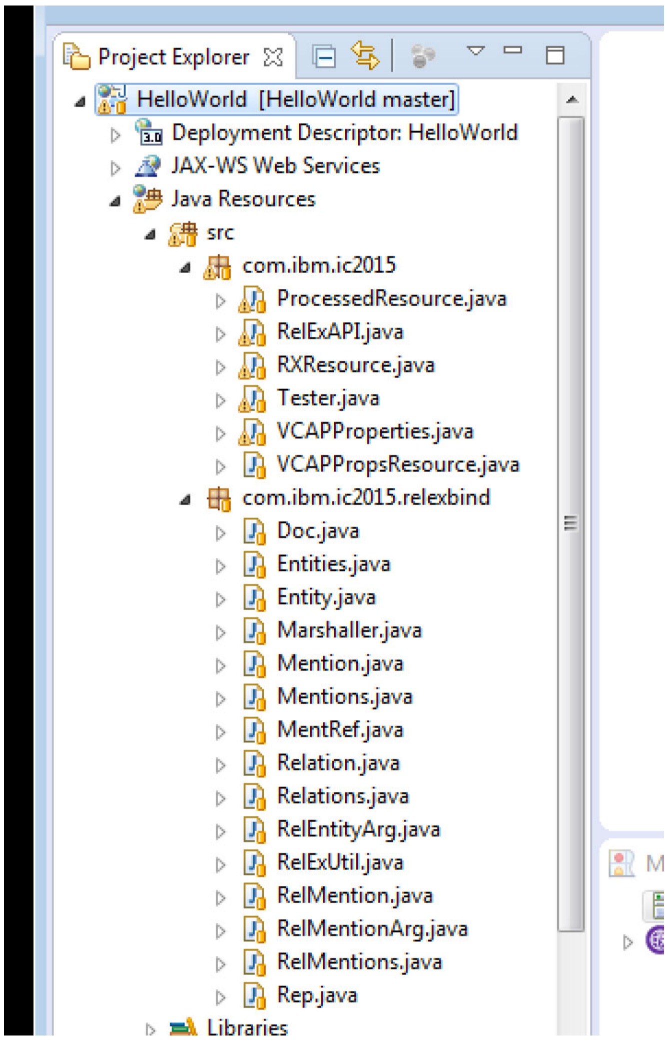
Lab code is on GitHub
https://github.com/WatsonDevCloud/HelloWorld

Bluemix web console
https://console.ng.bluemix.net/

1. Project Overview

1.1 Java Files

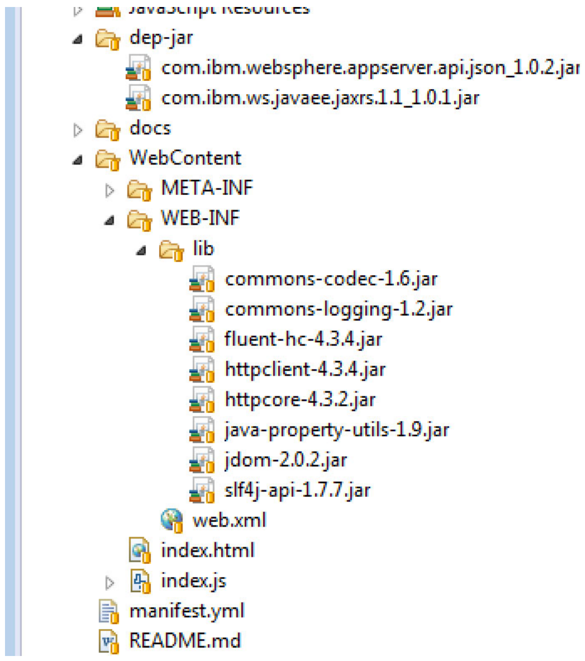
The skeleton app contains the files we will be working on which are contained in the package com.ibm.ic2015. We also provide some helper code in the com.ibm.ic2015.relexbind package. This helper code does things like binding XML to Java objects and mining useful data from the Relationship Extraction results.



```
Project Explorer
HelloWorld [HelloWorld master]
  Deployment Descriptor: HelloWorld
  JAX-WS Web Services
  Java Resources
    src
      com.ibm.ic2015
        ProcessedResource.java
        RelExAPI.java
        RXResource.java
        Tester.java
        VCAPProperties.java
        VCAPPropsResource.java
      com.ibm.ic2015.relexbind
        Doc.java
        Entities.java
        Entity.java
        Marshaller.java
        Mention.java
        Mentions.java
        MentRef.java
        Relation.java
        Relations.java
        RelEntityArg.java
        RelExUtil.java
        RelMention.java
        RelMentionArg.java
        RelMentions.java
        Rep.java
    Libraries
```

1.2 Jar Files

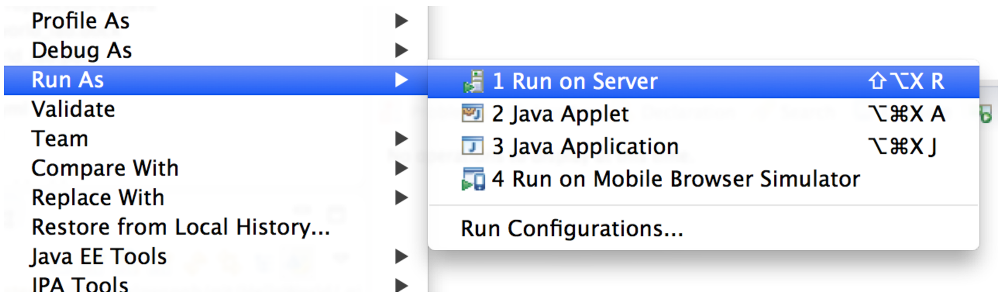
We also provide useful jar libraries. The jar files in the *dep-jar* are part of the Liberty Profile runtime environment and as such don't need to be deployed as part of our app. The jar files in the *WebContent/WEB-INF/lib* directory are deployed as part of our web app.



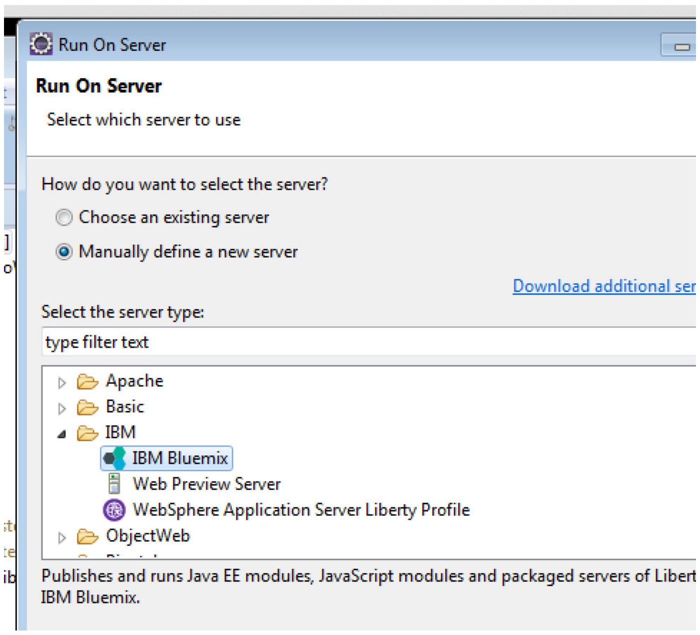
```
▷ JavaScript Resources
◢ dep-jar
    com.ibm.websphere.appserver.api.json_1.0.2.jar
    com.ibm.ws.javaee.jaxrs.1.1_1.0.1.jar
▷ docs
◢ WebContent
  ▷ META-INF
  ◢ WEB-INF
    ◢ lib
        commons-codec-1.6.jar
        commons-logging-1.2.jar
        fluent-hc-4.3.4.jar
        httpclient-4.3.4.jar
        httpcore-4.3.2.jar
        java-property-utils-1.9.jar
        jdom-2.0.2.jar
        slf4j-api-1.7.7.jar
      web.xml
    index.html
  ▷ index.js
  manifest.yml
  README.md
```

2. Deploy to Bluemix

With your mouse right click on the HelloWorld project title, in Package Explorer, and select ***Run As > Run On Server***.



From the Run On Server dialog select the ***Manually define a new server*** option. Then click on the ***IBM/IBM Bluemix*** server type and then click next.


Run On Server

Run On Server

Select which server to use

How do you want to select the server?

( ) Choose an existing server

(●) Manually define a new server

Download additional ser

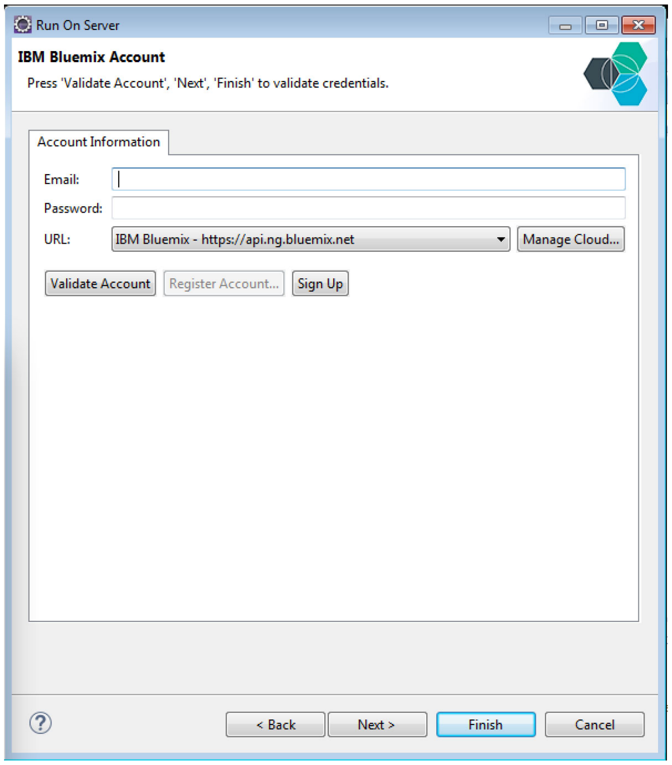
Select the server type:

type filter text

- Apache
- Basic
- IBM
  - IBM Bluemix
  - Web Preview Server
  - WebSphere Application Server Liberty Profile
- ObjectWeb

Publishes and runs Java EE modules, JavaScript modules and packaged servers of Libert
IBM Bluemix.

Enter the Bluemix ***Email*** and ***Password,*** using the credentials provided to you earlier, and click ***Next***.


Run On Server

IBM Bluemix Account

Press 'Validate Account', 'Next', 'Finish' to validate credentials.

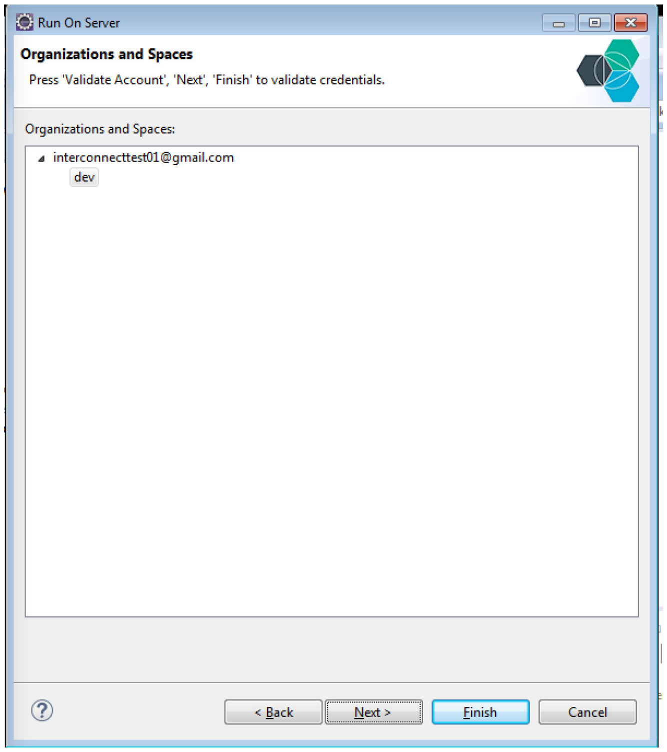
Account Information

Email:

Password:

URL: IBM Bluemix - https://api.ng.bluemix.net

Leave the Organizations and Spaces at the default setting of *dev* and click ***Next***.


Run On Server

Organizations and Spaces

Press 'Validate Account', 'Next', 'Finish' to validate credentials.

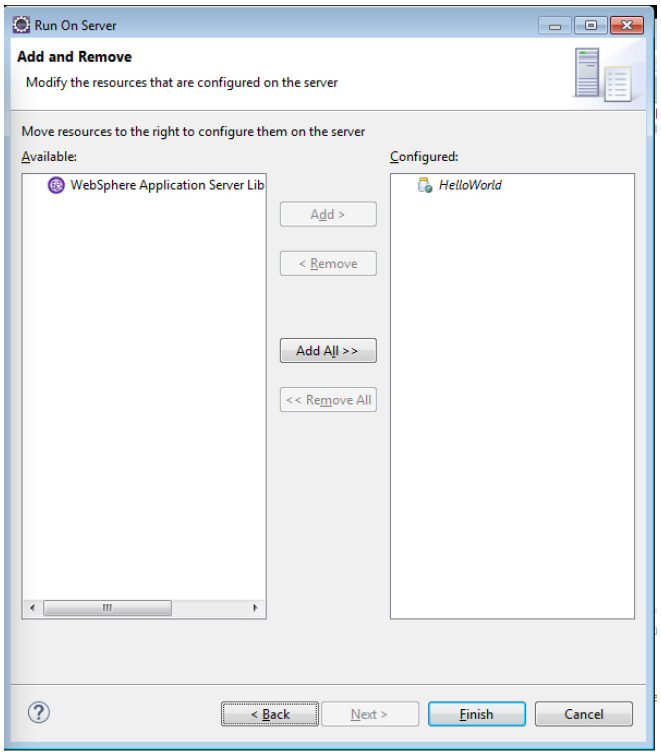
Organizations and Spaces:

interconnecttest01@gmail.com

dev

The *Add and Remove* dialog lists the app we want to deploy in it's *Configured* pane. Click ***Finish***.

Note: The Configured pane also lists any other apps currently running under our Bluemix account. Don't remove them unless you want to delete them completely from Bluemix!


Run On Server

Add and Remove

Modify the resources that are configured on the server

Move resources to the right to configure them on the server

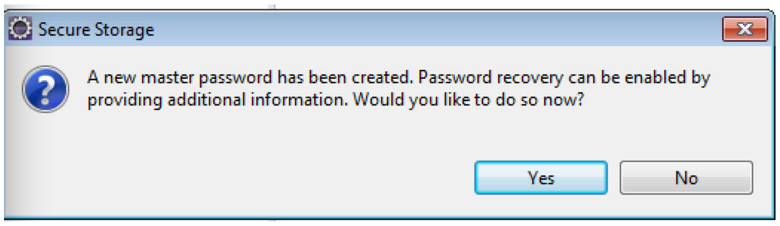
Available:

WebSphere Application Server Lib

Configured:

HelloWorld

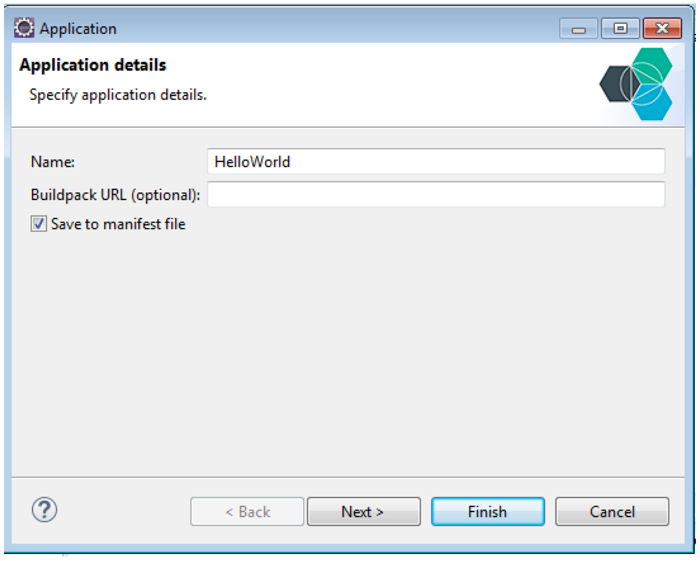
Click the ***No*** button.


Secure Storage

A new master password has been created. Password recovery can be enabled by providing additional information. Would you like to do so now?

Select the ***Save to manifest file*** checkbox then click the ***Next*** button in the *Application details* dialog.

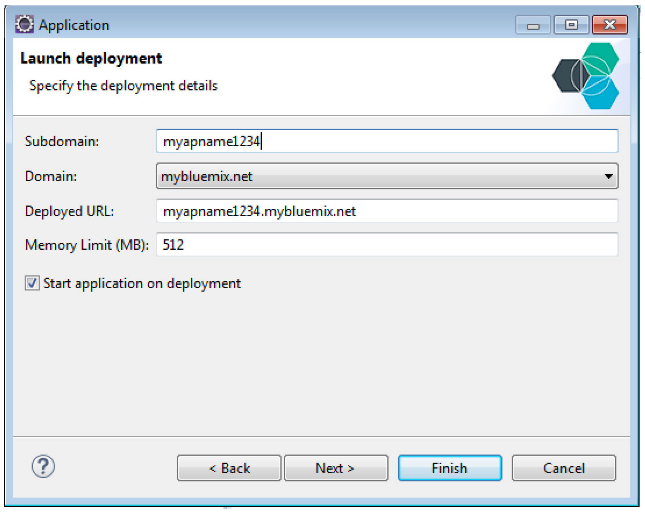
Note: Bluemix uses the manifest.yml file for configuration info when using the command line tools. Even though, we are using the Eclipse Bluemix plugin it is good practice to keep them in sync as the plugin may use manifest.yml in certain cases.



Name: HelloWorld
Buildpack URL (optional):
☑ Save to manifest file

In the *Launch deployment* dialog we will be defining the URL for our application. The URL will take the form of *subdomain.mybluemix.net*. In the *Subdomain* field enter a unique name for your app and click ***Next***.

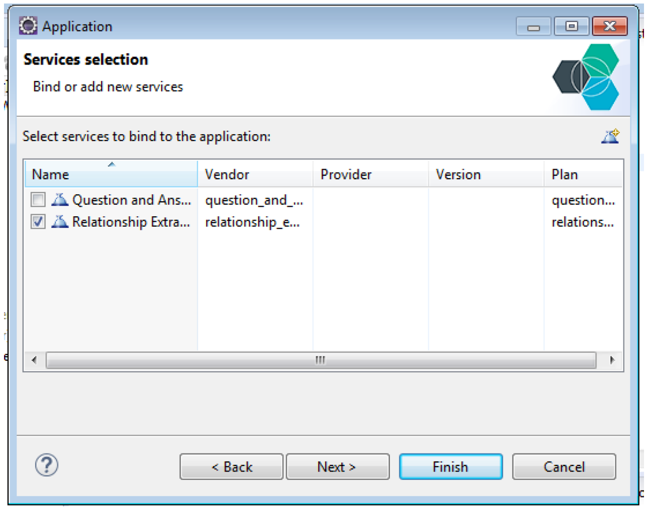
Note: the Subdomain name you give your app must be globally unique in the Bluemix space. It might take several tries to find a unique name. Don't use spaces, periods, or any other character that is not legal for a domain name.



Subdomain: myapname1234
Domain: mybluemix.net
Deployed URL: myapname1234.mybluemix.net
Memory Limit (MB): 512
☑ Start application on deployment

In the ***Services selection*** dialog, check Relationship Extraction and click ***Finish***.

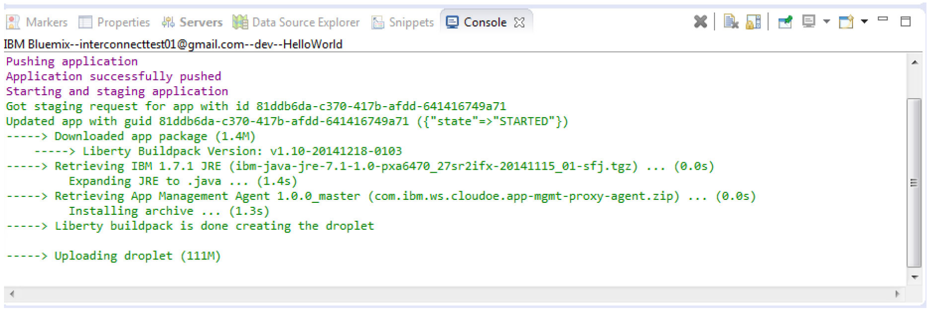
Note: The services in this dialog are those that we added to our test app in the previous lab. Here we are selecting to also bind the Relationship Extraction service to this app. Instead of reusing this service we could add a new one using the Bluemix web console.



Name	Vendor	Provider	Version	Plan
☐ Question and Ans...	question_and_...			question...
☑ Relationship Extra...	relationship_e...			relations...

The Eclipse Console will start displaying the progress of the Bluemix deployment.

Note: When using the Eclipse Bluemix plugin, stdout and stderr app output will display in the console. This can be useful for debugging.

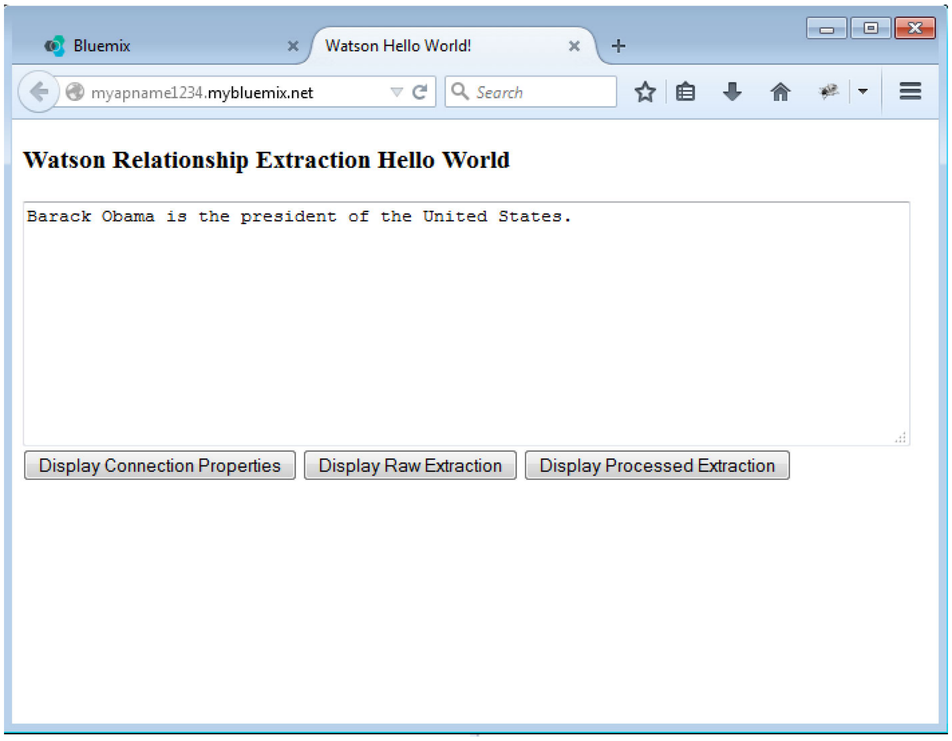


```
IBM Bluemix--interconnecttest01@gmail.com--dev--HelloWorld
Pushing application
Application successfully pushed
Starting and staging application
Got staging request for app with id 81ddb6da-c370-417b-afdd-641416749a71
Updated app with guid 81ddb6da-c370-417b-afdd-641416749a71 ({"state"=>"STARTED"})
-----> Downloaded app package (1.4M)
    -----> Liberty Buildpack Version: v1.10-20141218-0103
-----> Retrieving IBM 1.7.1 JRE (ibm-java-jre-7.1-1.0-pxa6470_27sr2ifx-20141115_01-sfj.tgz) ... (0.0s)
         Expanding JRE to .java ... (1.4s)
-----> Retrieving App Management Agent 1.0.0_master (com.ibm.ws.cloudoe.app-mgmt-proxy-agent.zip) ... (0.0s)
         Installing archive ... (1.3s)
-----> Liberty buildpack is done creating the droplet

-----> Uploading droplet (111M)
```

After your app has successfully deployed, Eclipse will open up the default app web page.

Note: We are using JavaScript to do AJAX calls to the app backend. The Eclipse built-in browser doesn’t handle JavaScript very well so open the URL in Chrome or Firefox instead.

Note: Sometimes the Bluemix plugin doesn’t realize the app is fully deployed. In these cases the web page may not popup in Eclipse. If you think this is the case, verify either by opening the page in Chrome or Firefox or using the Bluemix web console (https://console.ng.bluemix.net/) to check deployment status.


Bluemix
Watson Hello World!
myapname1234.mybluemix.net
Search

Watson Relationship Extraction Hello World

```
Barack Obama is the president of the United States.
```

Display Connection Properties
Display Raw Extraction
Display Processed Extraction

3. Manifest.yml

Below is an example of a manifest.yml file. As mentioned earlier, it holds deployment info used by the command line tools. We will be looking at this more in later labs.

FILE: manifest.yml

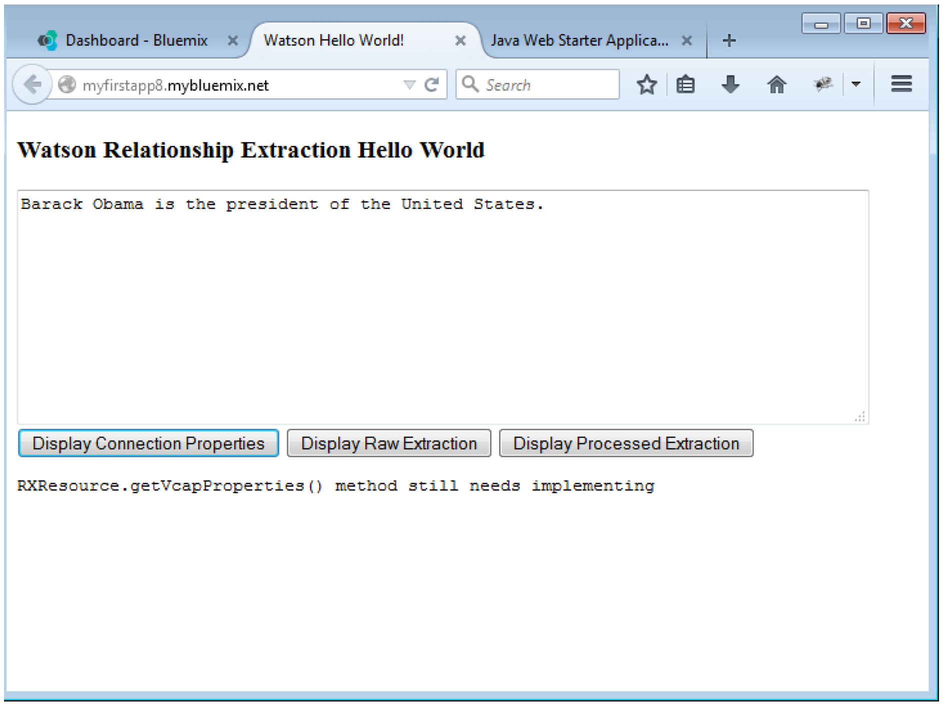
```
applications:
- disk_quota: 1024M
  instances: 1
- name: HelloWorld
  memory: 512M
  host: myapp999
  domain: mybluemix.net
  env: {
    }
  services: {
    }
```

4. The Skeleton App

At this point our app doesn't do much. It has a textarea that holds text that we want to process and some button.

Right now, if you push any of the buttons an AJAX call will be made to the app backend via its REST-like APIs. Currently these APIs just return some text that tells you the name of a method that needs to be implemented. We will be implementing these methods in this lab.

The first button will, when implemented, display the connection information for the Watson Relationship Extraction service.


Dashboard - Bluemix
Watson Hello World!
Java Web Starter Applica...
myfirstapp8.mybluemix.net
Search

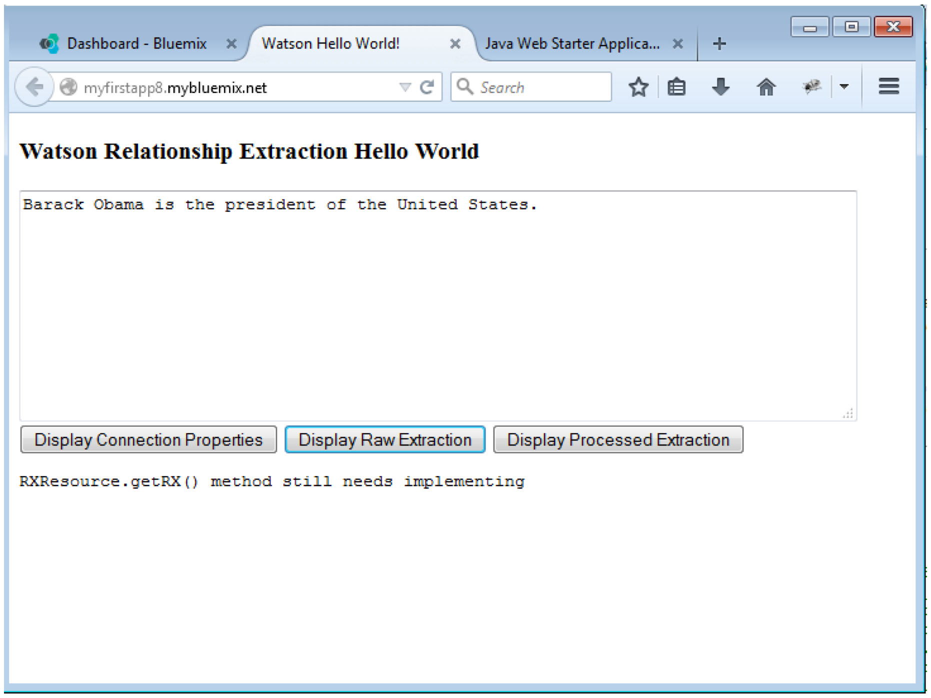
Watson Relationship Extraction Hello World

```
Barack Obama is the president of the United States.
```

Display Connection Properties
Display Raw Extraction
Display Processed Extraction

```
RXResource.getVcapProperties() method still needs implementing
```

The second button will, when implemented, display the raw XML text provided by the Watson Relationship Extraction service.


Dashboard - Bluemix
Watson Hello World!
Java Web Starter Applica...
myfirstapp8.mybluemix.net
Search

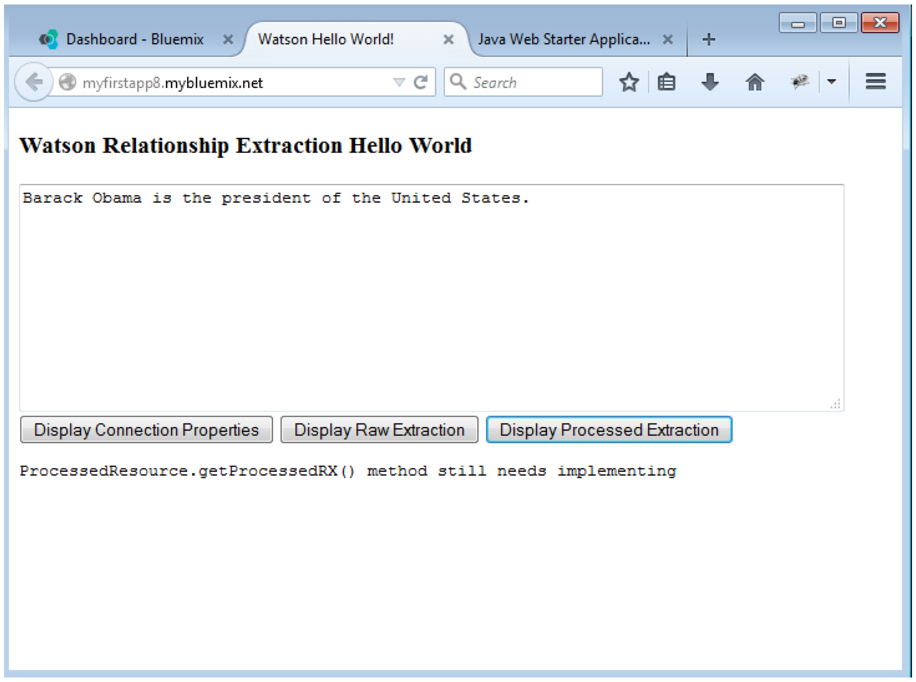
Watson Relationship Extraction Hello World

```
Barack Obama is the president of the United States.
```

Display Connection Properties
Display Raw Extraction
Display Processed Extraction

```
RXResource.getRX() method still needs implementing
```

The third button will, when implemented, extract the entity and relationship information from the raw XML and display it in a human readable format.


Dashboard - Bluemix
Watson Hello World!
Java Web Starter Applica...
myfirstapp8.mybluemix.net
Search

Watson Relationship Extraction Hello World

```
Barack Obama is the president of the United States.
```

Display Connection Properties
Display Raw Extraction
Display Processed Extraction

```
ProcessedResource.getProcessedRX() method still needs implementing
```

5. Coding the App

As mentioned before, we will be creating our app by modifying the skeleton code in the ***com.ibm.ic2015*** package.

Note: The code solution Java files are provided in the ***docs/solutions*** directory.

5.1 Display Connection Properties

In this section we will be writing the code required to implement the Display Connection Properties button. When implemented, it will display the connection information for the Watson Relationship Extraction service.

5.1.1 VCAPProperties.java

As we saw in the previous section, connection properties for Bluemix services are provided via the environment variable *VCAP_ SERVICES*. The properties are represented in the JSON format.

The getVcapServices() method reads the *VCAP_ SERVICES* environment variable and places it in a Java object (JSONObject). The processVCAP_Services() extracts the connection parameters from the JSONObject and sets the member variables.

In Eclipse Package Explorer, double click on the skeleton VCAPProperties.java file in the ***com.ibm.ic2015*** package, to bring it up in the editor.

Update the getVcapServices() and processVCAP_Services() methods with the VCAPProperties.java code below.

FILE: com/ibm/ic2015/VCAPProperties.java

```
package com.ibm.ic2015;

import java.io.IOException;

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

public class VCAPProperties {

      private String serviceName = "<service name>";
      private String baseURL = "<service url>";
      private String username = "<service username>";
      private String password = "<service password>";


      public VCAPProperties(String serviceName) {
            this.serviceName = serviceName;
            processVCAP_Services();
      }


    /**
     * If exists, process the VCAP_SERVICES environment variable in order to get
the
     * username, password and baseURL
     */
    private void processVCAP_Services() {
      System.out.println("Processing VCAP_SERVICES");

        JSONObject sysEnv = getVcapServices();
        if (sysEnv == null) {
            return;
        }

        System.out.println("Looking for: " + serviceName );

        if (sysEnv.containsKey(serviceName)) {
                  JSONArray services = (JSONArray)sysEnv.get(serviceName);
                  JSONObject service = (JSONObject)services.get(0);
                  JSONObject credentials = (JSONObject)service.get("credentials");

                  baseURL = (String)credentials.get("url");
                  username = (String)credentials.get("username");
                  password = (String)credentials.get("password");

                  System.out.println("baseURL  = " + baseURL);
                  System.out.println("username   = " + username);
                  System.out.println("password = " + password);
      } else {
            System.out.println(serviceName + " is not available in VCAP_SERVICES, "
                        + "please bind the service to your application");
        }
    }


    /**
     * Gets the <b>VCAP_SERVICES</b> environment variable and return it
     *  as a JSONObject.
     *
     * @return the VCAP_SERVICES as Json
     */
    private JSONObject getVcapServices() {
        String envServices = System.getenv("VCAP_SERVICES");

        if (envServices == null) {
            return null;
        }

        JSONObject sysEnv = null;
        try {
             sysEnv = JSONObject.parse(envServices);
        } catch (IOException e) {
            // Do nothing, fall through to defaults
            System.out.println("Error parsing VCAP_SERVICES: " + e.getMessage());
        }

        return sysEnv;
    }

      public String getBaseURL() {
            return baseURL;
      }
      public String getUsername() {
            return username;
      }
      public String getPassword() {
            return password;
      }
      public String getServiceName() {
            return serviceName;
      }

}
```

5.1.2 VCAPPropsResource.java

In Eclipse Package Explorer, double click on the skeleton *VCAPPropsResource.java* file in the ***com.ibm.ic2015*** package, to bring it up in the editor***.***

As mentioned earlier, the backend services our app provides to the front end is with the use of a REST-like API. For this lab we are using JAX-RS which is preconfigured in the skeleton app.

Note: You can see the JAX-RS configuration in the *WebContent/WEB-INF/web.xml* file. The endpoint names are defined by the *@Path* annotation in front of the class declaration in *VCAPPropsResource.java* file. Based on the config in web.xml and the annotations in the code, the URL to this resource is http://myappname.mybluemix.net/api/vcapprops.

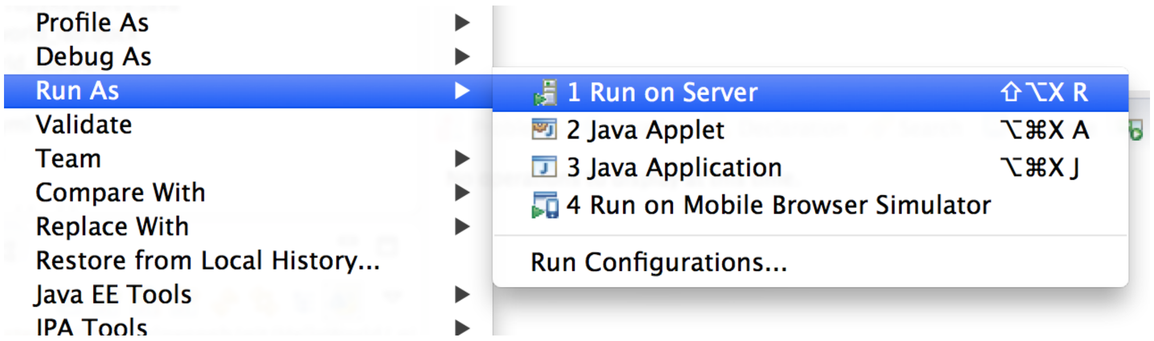
Update the getVcapProperties() method with the *VCAPPropsResource.java* code below.

FILE: com/ibm/ic2015/ VCAPPropsResource.java

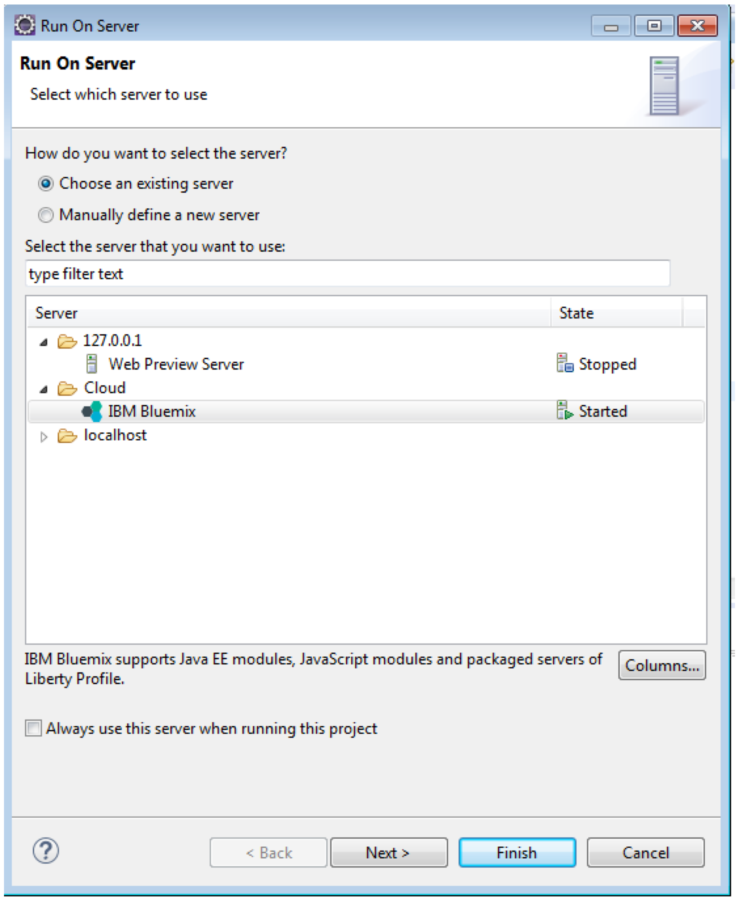
```
package com.ibm.ic2015;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;


@Path("/vcapprops")
public class VCAPPropsResource {

      @GET
      @Produces("text/plain")
      public String getVcapProperties() {

            VCAPProperties vcp = new VCAPProperties("relationship_extraction");

            String props =    "Resource: " + vcp.getServiceName() +
                                    "\nURL: " + vcp.getBaseURL() +
                                    "\nUsername: " + vcp.getUsername() +
                                    "\nPassword: " + vcp.getPassword();

            return props;

      }

}
```

5.1.3 Redeploy the App

With your mouse right click on the HelloWorld project title, in Package Explorer, and select ***Run As > Run On Server***.



Select ***Cloud > IBM Bluemix*** from the server list and click ***Finish***.



Run On Server

Select which server to use

How do you want to select the server?

(●) Choose an existing server

( ) Manually define a new server

Select the server that you want to use:

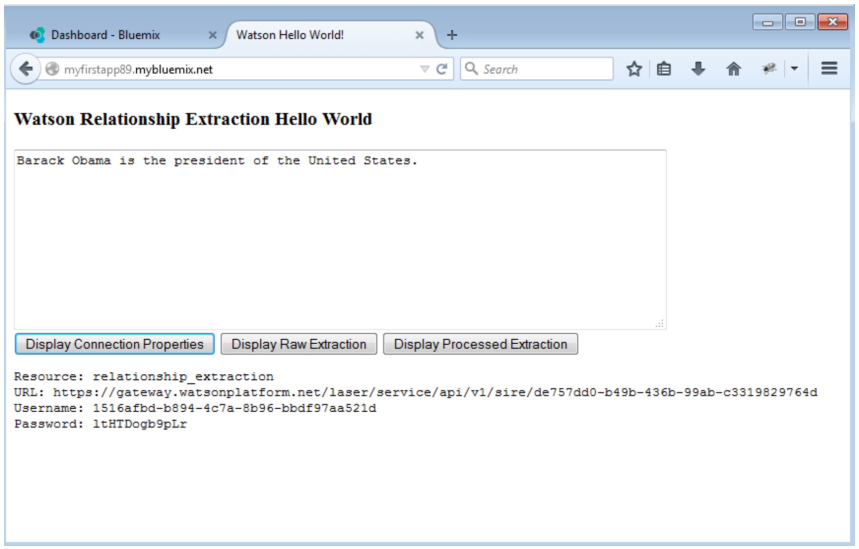
type filter text

Server	State
127.0.0.1	
Web Preview Server	Stopped
Cloud	
IBM Bluemix	Started
localhost	

IBM Bluemix supports Java EE modules, JavaScript modules and packaged servers of Liberty Profile.

[ ] Always use this server when running this project

After the app is deployed, open it in your browser. Refresh the page if necessary. Click on the ***Display Connection Properties*** button. The properties will display.


Dashboard - Bluemix
Watson Hello World!
myfirstapp89.mybluemix.net
Search
Display Connection Properties
Display Raw Extraction
Display Processed Extraction

Watson Relationship Extraction Hello World

```
Barack Obama is the president of the United States.
```

```
Resource: relationship_extraction
URL: https://gateway.watsonplatform.net/laser/service/api/v1/sire/de757dd0-b49b-436b-99ab-c3319829764d
Username: 1516afbd-b894-4c7a-8b96-bbdf97aa521d
Password: 1tHTDogb9pLr
```

5.2 Display Raw Extraction

In this section we will be writing the code required to implement the ***Display Raw Extraction*** button. When implemented, it will send the text on the web interface to the Relationship Extraction service and display the XML returned from the service.

5.2.1 RelExAPI.java

I this Java class we are calling the Watson Relationship Extraction REST service using an HTTP POST. What the Watson Relationship Extraction REST service returns is an XML document. The performExtraction() method returns this XML as a string. We will be using the Apache HTTPClient library to handle the HTTP POST and Basic Auth.

We pass to the performExtraction() method are the connection parameters, as found by the code in the previous sections, and the text we want to analyze. There is one other parameter that we have not discussed before and that is the SID.

The SID instructs the service to use the specified context when processing the text. For this lab we will be using the ***ie-en-news*** SID. ie-en-news handles the given text in the context of news article in the English language. A description of the available SIDs is provided in the VCAP_SERVICES environment variable.

Update the performExtraction() method with the RelExAPI.*java* code below.

FILE: com/ibm/ic2015/ RelExAPI.java

```
package com.ibm.ic2015;

import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.ArrayList;
import java.util.List;

import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.fluent.Executor;
import org.apache.http.client.fluent.Request;
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.entity.ContentType;
import org.apache.http.message.BasicNameValuePair;

import com.ibm.ic2015.relexbind.Marshaller;
import com.ibm.ic2015.relexbind.Rep;

public class RelExAPI {


      /*
       * Using Apache HTTPClient library to call the Relationship Extraction REST
service.
       * Results are an XML document.
       */
      public static String performExtraction(String text, String sid, String url,
String username, String password) {

            System.out.println("text: " + text);
            System.out.println("sid: " + sid);
            System.out.println("url: " + url);
            System.out.println("username: " + username);
            System.out.println("password: " + password);

            List<NameValuePair> queryParams = new ArrayList<NameValuePair>();
            queryParams.add(new BasicNameValuePair("txt", text ));
            queryParams.add(new BasicNameValuePair("sid", sid ));
            queryParams.add(new BasicNameValuePair("rt", "xml" ));

            String response = "ERROR";
      try {
            Executor executor = Executor.newInstance().auth(username, password);
            URI serviceURI = new URI(url).normalize();
          byte[] responseBytes = null;

                  responseBytes = executor.execute(Request.Post(serviceURI)
                      .bodyString(URLEncodedUtils.format(queryParams, "utf-8"),
                              ContentType.APPLICATION_FORM_URLENCODED)
                      ).returnContent().asBytes();

          response = new String(responseBytes, "UTF-8");

            } catch (URISyntaxException e) {
                  e.printStackTrace();
            } catch (ClientProtocolException e) {
                  e.printStackTrace();
            } catch (IOException e) {
                  e.printStackTrace();
            }

      return response;
      }

}
```

5.2.2 Tester.java (step 1 of 3)

Because the debug cycle can be long if we repeatedly have to deploy to Bluemix, we are going to create a test program that allows us to call the service when running locally. Some Bluemix services, not all, have an API that is available outside the Bluemix runtime environment. Watson Relationship Extraction service currently allows this.

In Eclipse Package Explorer, double click on the skeleton Tester.java file in the ***com.ibm.ic2015*** package, to bring it up in the editor*.*

In the Tester main() method, enter code for step 1 of 3 given below. Replace the values for the url, username, and password variables with the values you obtained earlier.

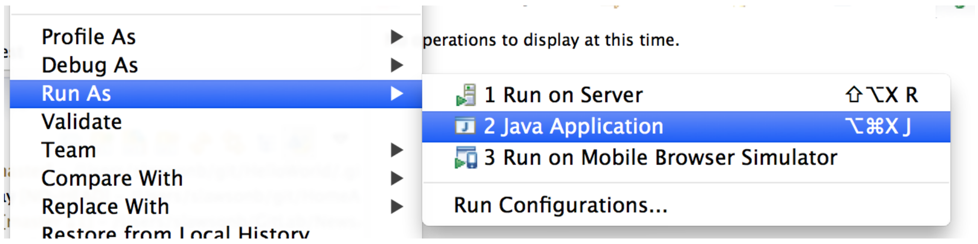
Note: If you don't update the values for the url, username, and password variables, your program will not work.

FILE: com/ibm/ic2015/ Tester.java (step 1 of 3)

```
package com.ibm.ic2015;

import java.util.List;

import com.ibm.ic2015.relexbind.Marshaller;
import com.ibm.ic2015.relexbind.RelExUtil;
import com.ibm.ic2015.relexbind.Rep;

public class Tester {


      /*
       *
       * Tester for Relationship Extraction API so we don't have to deploy to
Bluemix to test.
       *
       */
      public static void main(String[] args) {
            String text = "Barack Obama is the president of the United States.";

            String url =
"https://gateway.watsonplatform.net/laser/service/api/v1/sire/cf997efc-3c5d-440d-
9c54-885b04d36c4e";
            String username = "ebe462b4-16d8-4ed7-9653-ad70999b34c";
            String password = "htbNjXFzHpG";


            String xml = RelExAPI.performExtraction(text,
                                          "ie-en-news",
                                          url,
                                          username,
                                          password);

            System.out.println(xml);

      }

}
```

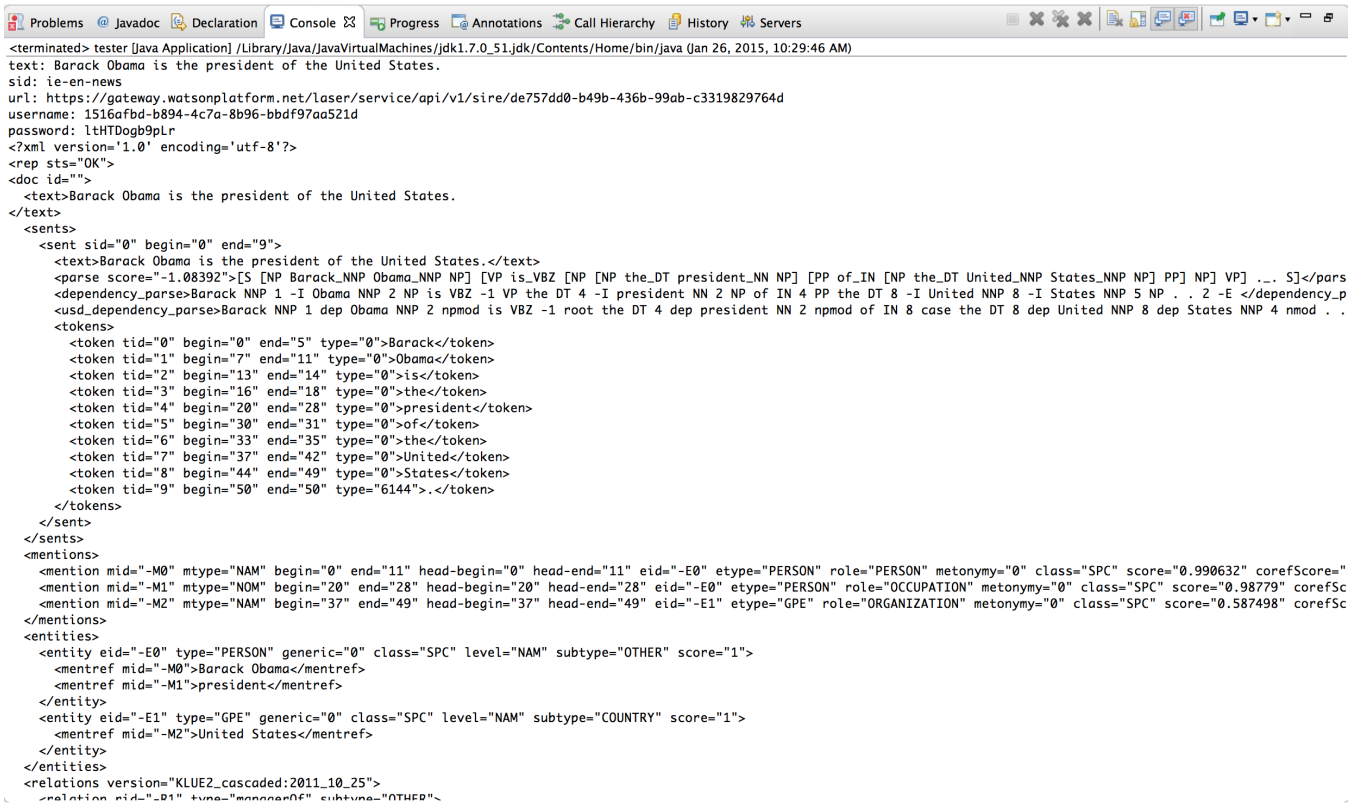
5.2.3 Run the test program (step 1 of 3)

Right click on the Tester.java file, in Package Explorer, and select Run As > Java Application.



If as is well, the XML from the Watson Relationship Extraction service will display in the console, as shown below.

Note: Fix any bugs and before you continue with the lab.



```
<terminated> tester [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_51.jdk/Contents/Home/bin/java (Jan 26, 2015, 10:29:46 AM)
text: Barack Obama is the president of the United States.
sid: ie-en-news
url: https://gateway.watsonplatform.net/laser/service/api/v1/sire/de757dd0-b49b-436b-99ab-c3319829764d
username: 1516afbd-b894-4c7a-8b96-bbdf97aa521d
password: ltHTDogb9pLr
<?xml version='1.0' encoding='utf-8'?>
<rep sts="OK">
<doc id="">
  <text>Barack Obama is the president of the United States.
</text>
  <sents>
    <sent sid="0" begin="0" end="9">
      <text>Barack Obama is the president of the United States.</text>
      <parse score="-1.08392">[S [NP Barack_NNP Obama_NNP NP] [VP is_VBZ [NP [NP the_DT president_NN NP] [PP of_IN [NP the_DT United_NNP States_NNP NP] PP] NP] VP] ._. S]</pars
      <dependency_parse>Barack NNP 1 -I Obama NNP 2 NP is VBZ -1 VP the DT 4 -I president NN 2 NP of IN 4 PP the DT 8 -I United NNP 8 -I States NNP 5 NP . . 2 -E </dependency_p
      <usd_dependency_parse>Barack NNP 1 dep Obama NNP 2 npmod is VBZ -1 root the DT 4 dep president NN 2 npmod of IN 8 case the DT 8 dep United NNP 8 dep States NNP 4 nmod . .
      <tokens>
        <token tid="0" begin="0" end="5" type="0">Barack</token>
        <token tid="1" begin="7" end="11" type="0">Obama</token>
        <token tid="2" begin="13" end="14" type="0">is</token>
        <token tid="3" begin="16" end="18" type="0">the</token>
        <token tid="4" begin="20" end="28" type="0">president</token>
        <token tid="5" begin="30" end="31" type="0">of</token>
        <token tid="6" begin="33" end="35" type="0">the</token>
        <token tid="7" begin="37" end="42" type="0">United</token>
        <token tid="8" begin="44" end="49" type="0">States</token>
        <token tid="9" begin="50" end="50" type="6144">.</token>
      </tokens>
    </sent>
  </sents>
  <mentions>
    <mention mid="-M0" mtype="NAM" begin="0" end="11" head-begin="0" head-end="11" eid="-E0" etype="PERSON" role="PERSON" metonymy="0" class="SPC" score="0.990632" corefScore="
    <mention mid="-M1" mtype="NOM" begin="20" end="28" head-begin="20" head-end="28" eid="-E0" etype="PERSON" role="OCCUPATION" metonymy="0" class="SPC" score="0.98779" corefSc
    <mention mid="-M2" mtype="NAM" begin="37" end="49" head-begin="37" head-end="49" eid="-E1" etype="GPE" role="ORGANIZATION" metonymy="0" class="SPC" score="0.587498" corefSc
  </mentions>
  <entities>
    <entity eid="-E0" type="PERSON" generic="0" class="SPC" level="NAM" subtype="OTHER" score="1">
      <mentref mid="-M0">Barack Obama</mentref>
      <mentref mid="-M1">president</mentref>
    </entity>
    <entity eid="-E1" type="GPE" generic="0" class="SPC" level="NAM" subtype="COUNTRY" score="1">
      <mentref mid="-M2">United States</mentref>
    </entity>
  </entities>
  <relations version="KLUE2_cascaded:2011_10_25">
    <relation rid="-R1" type="managerOf" subtype="OTHER">
```

5.2.4 RXResource.java

Now that we have tested our app we can create a way to call it with a REST call.

In Eclipse Package Explorer, double click on the skeleton RXResource.java file in the ***com.ibm.ic2015*** package, to bring it up in the editor***.***

Update the getRX() methods with the RXResource.java code below. This will give us an endpoint of http://myappname.mybluemix.net/api/rx.

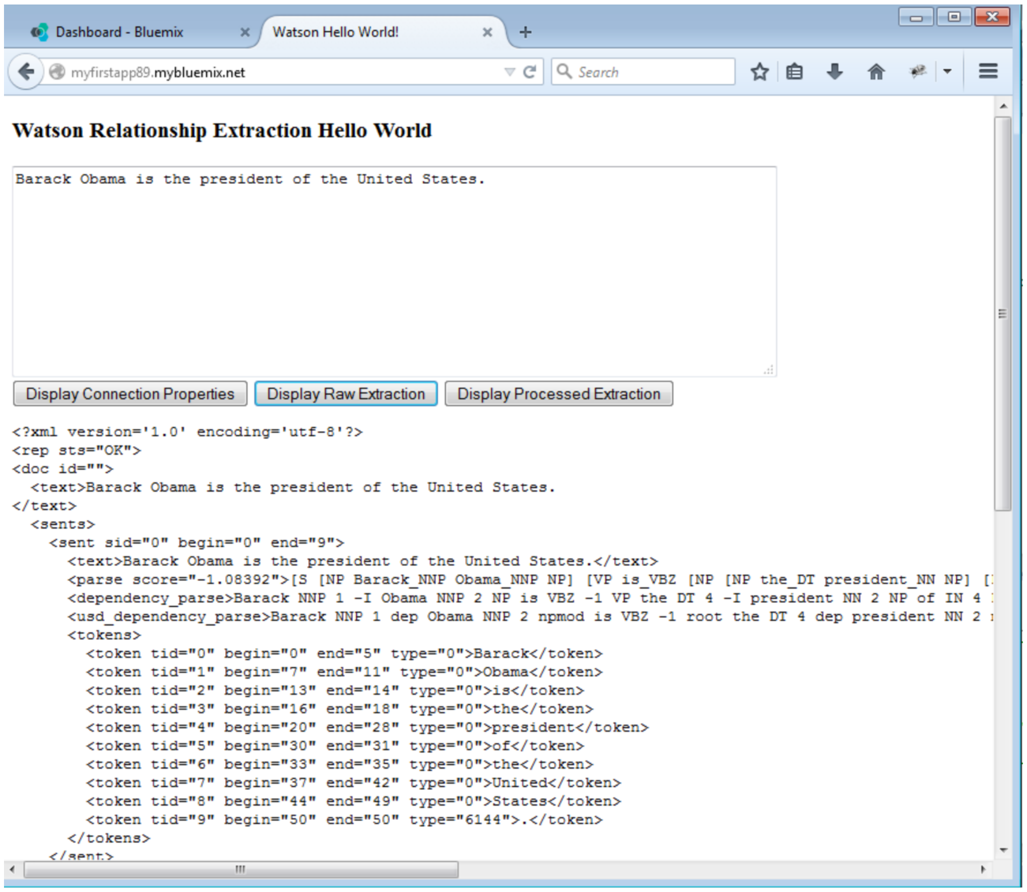
FILE: com/ibm/ic2015/ RXResource.java

```
package com.ibm.ic2015;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;


@Path("/vcapprops")
public class VCAPPropsResource {

	@GET
	@Produces("text/plain")
	public String getVcapProperties() {

		VCAPProperties vcp = new VCAPProperties("relationship_extraction");

		String props =	"Resource: " + vcp.getServiceName() +
						"\nURL: " + vcp.getBaseURL() +
						"\nUsername: " + vcp.getUsername() +
						"\nPassword: " + vcp.getPassword();

		return props;

	}

}
```

5.2.4 Deploy the App

Follow the same steps as in section 5.1.3 to deploy the app.

After the app is deployed, open it in your browser. Refresh the page if necessary. Click on the ***Display Raw Extraction*** button. The XML will display.


Dashboard - Bluemix
Watson Hello World!
myfirstapp89.mybluemix.net
Search

Watson Relationship Extraction Hello World

```
Barack Obama is the president of the United States.
```

Display Connection Properties
Display Raw Extraction
Display Processed Extraction

```
<?xml version='1.0' encoding='utf-8'?>
<rep sts="OK">
<doc id="">
  <text>Barack Obama is the president of the United States.
</text>
  <sents>
    <sent sid="0" begin="0" end="9">
      <text>Barack Obama is the president of the United States.</text>
      <parse score="-1.08392">[S [NP Barack_NNP Obama_NNP NP] [VP is_VBZ [NP [NP the_DT president_NN NP] [
      <dependency_parse>Barack NNP 1 -I Obama NNP 2 NP is VBZ -1 VP the DT 4 -I president NN 2 NP of IN 4 
      <usd_dependency_parse>Barack NNP 1 dep Obama NNP 2 npmod is VBZ -1 root the DT 4 dep president NN 2 
      <tokens>
        <token tid="0" begin="0" end="5" type="0">Barack</token>
        <token tid="1" begin="7" end="11" type="0">Obama</token>
        <token tid="2" begin="13" end="14" type="0">is</token>
        <token tid="3" begin="16" end="18" type="0">the</token>
        <token tid="4" begin="20" end="28" type="0">president</token>
        <token tid="5" begin="30" end="31" type="0">of</token>
        <token tid="6" begin="33" end="35" type="0">the</token>
        <token tid="7" begin="37" end="42" type="0">United</token>
        <token tid="8" begin="44" end="49" type="0">States</token>
        <token tid="9" begin="50" end="50" type="6144">.</token>
      </tokens>
    </sent>
```

5.2.5 Tester.java (step 2 of 3)

We are now going to update the Tester class to use some of our helper code provided in the ***com.ibm.ic2015.relexbind*** package. In this step, we are binding the XML to Java objects using JAXB.

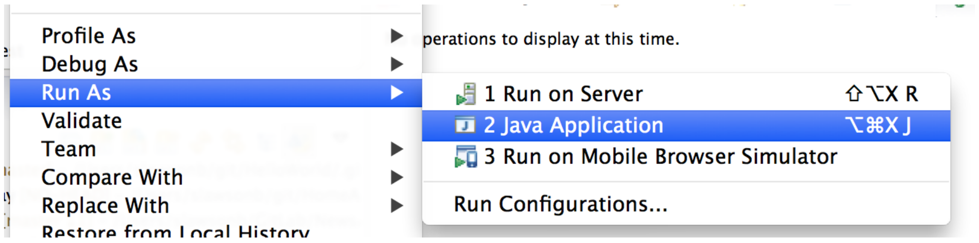
In the Tester.java main() method, add the 2 additional lines of code shown below.

FILE: com/ibm/ic2015/ Tester.java (step 2 of 3)

```
package com.ibm.ic2015;

import java.util.List;

import com.ibm.ic2015.relexbind.Marshaller;
import com.ibm.ic2015.relexbind.RelExUtil;
import com.ibm.ic2015.relexbind.Rep;

public class Tester {


      /*
       *
       * Tester for Relationship Extraction API so we don't have to deploy to
Bluemix to test.
       *
       */
      public static void main(String[] args) {
            String text = "Barack Obama is the president of the United States.";

            String url =
"https://gateway.watsonplatform.net/laser/service/api/v1/sire/cf997efc-3c5d-440d-
9c54-885b04d56c4e";
            String username = "ebe462b4-16d8-4ed7-9653-ad70999bb34c";
            String password = "htbNjBXFzHpG";


            String xml = RelExAPI.performExtraction(text,
                                          "ie-en-news",
                                          url,
                                          username,
                                          password);

            System.out.println(xml);

            Rep rep = Marshaller.marshallXml(xml);

            System.out.println(rep);

      }

}
```

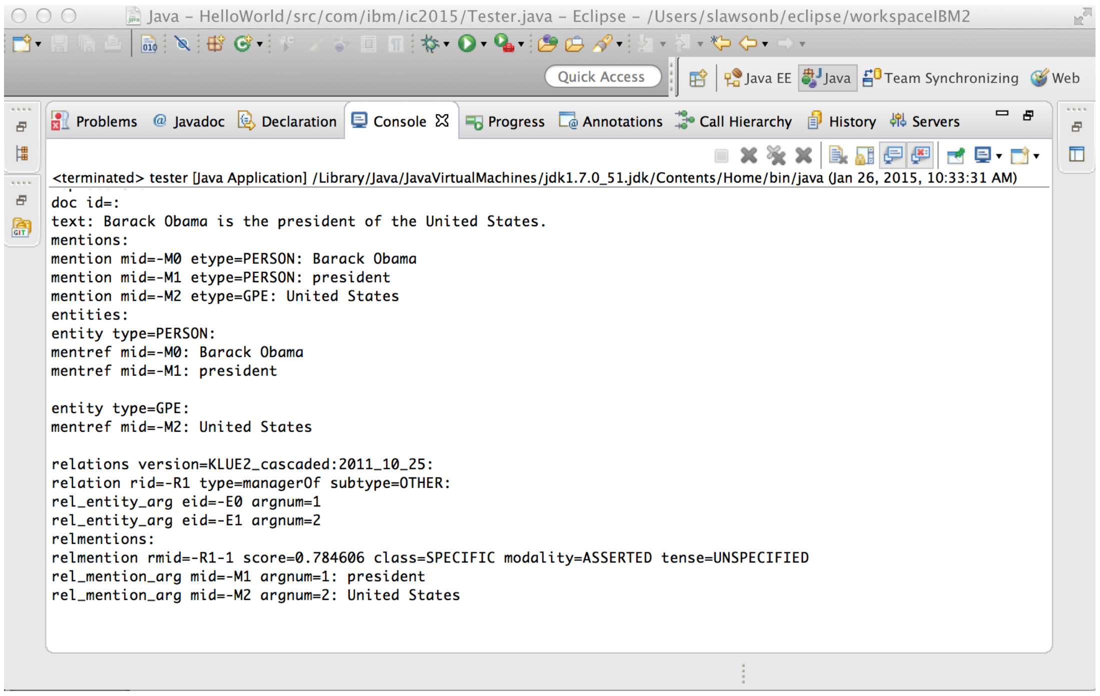
5.2.6 Run the test program (step 2 of 3)

Right click on the Tester.java file, in Package Explorer, and select Run As > Java Application.



If as is well, you will see what is displayed below.

Note: Fix any bugs and before you continue with the lab.



```
<terminated> tester [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_51.jdk/Contents/Home/bin/java (Jan 26, 2015, 10:33:31 AM)
doc id=:
text: Barack Obama is the president of the United States.
mentions:
mention mid=-M0 etype=PERSON: Barack Obama
mention mid=-M1 etype=PERSON: president
mention mid=-M2 etype=GPE: United States
entities:
entity type=PERSON:
mentref mid=-M0: Barack Obama
mentref mid=-M1: president

entity type=GPE:
mentref mid=-M2: United States

relations version=KLUE2_cascaded:2011_10_25:
relation rid=-R1 type=managerOf subtype=OTHER:
rel_entity_arg eid=-E0 argnum=1
rel_entity_arg eid=-E1 argnum=2
relmentions:
relmention rmid=-R1-1 score=0.784606 class=SPECIFIC modality=ASSERTED tense=UNSPECIFIED
rel_mention_arg mid=-M1 argnum=1: president
rel_mention_arg mid=-M2 argnum=2: United States
```

5.2.7 Tester.java (step 3 of 3)

We are now going to update the Tester class to use some more of our helper code provided in the ***com.ibm.ic2015.relexbind*** package. In this step, we are using some utilities to extract and display human readable results .

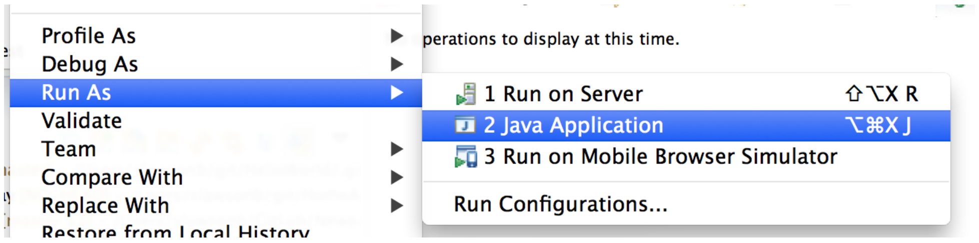
In the Tester.java main() method, add the 5 additional lines of code shown below.

FILE: com/ibm/ic2015/ Tester.java (step 3 of 3)

```
package com.ibm.ic2015;

import java.util.List;

import com.ibm.ic2015.relexbind.Marshaller;
import com.ibm.ic2015.relexbind.RelExUtil;
import com.ibm.ic2015.relexbind.Rep;

public class Tester {


	/*
	 *
	 * Tester for Relationship Extraction API so we don't have to deploy to
Bluemix to test.
	 *
	 */
	public static void main(String[] args) {
		String text = "Barack Obama is the president of the United States.";

		String url =
"https://gateway.watsonplatform.net/laser/service/api/v1/sire/cf997efc-3c5d-440d-
9c54-885b04d56c4e";
		String username = "ebe462b4-16d8-4ed7-9653-ad70999bb34c";
		String password = "htbNjBXFzHpG";


		String xml = RelExAPI.performExtraction(text,
		                                  "ie-en-news",
		                                  url,
		                                  username,
		                                  password);

		System.out.println(xml);

		Rep rep = Marshaller.marshallXml(xml);

		System.out.println(rep);

		RelExUtil util = new RelExUtil(rep);

		List<String> entities = util.getEntitiesList();
		List<String> rels = util.getRelationsList();

		System.out.println(RelExUtil.stringFromList(entities));
		System.out.println(RelExUtil.stringFromList(rels));

	}
}
```

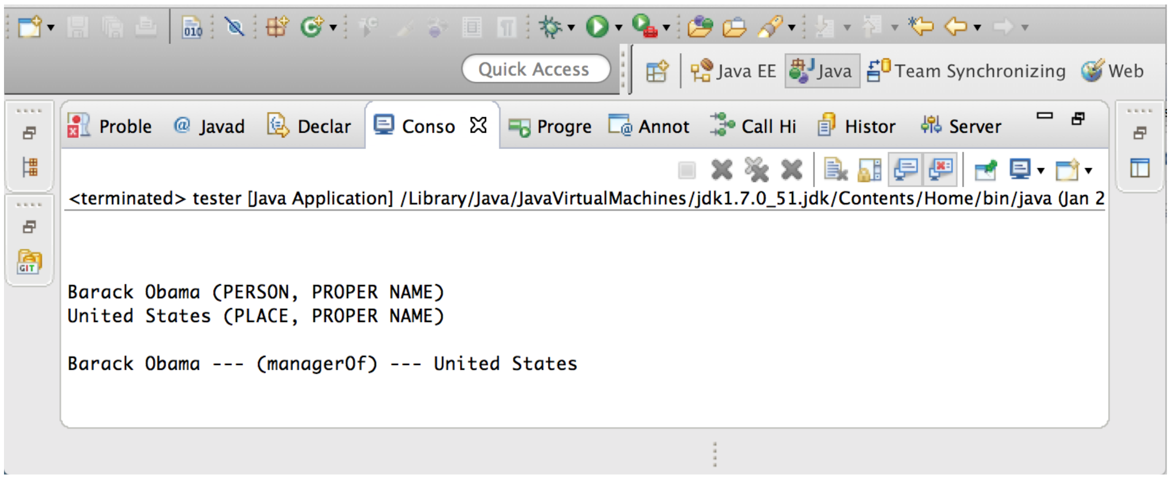
5.2.8 Run the test program (step 3 of 3)

Right click on the Tester.java file, in Package Explorer, and select Run As > Java Application.



If as is well, you will see what is displayed below.

Note: Fix any bugs and before you continue with the lab.



```
<terminated> tester [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_51.jdk/Contents/Home/bin/java (Jan 2


Barack Obama (PERSON, PROPER NAME)
United States (PLACE, PROPER NAME)

Barack Obama --- (managerOf) --- United States
```

5.3 Display Processed Extraction

In this section we will be writing the code required to implement the Display Processed Extraction button.

5.3.1 ProcessedResource.java

Now that we have tested our app we can create a way to call it with a REST call.

In Eclipse Package Explorer, double click on the skeleton ProcessedResource.java file in the ***com.ibm.ic2015*** package, to bring it up in the editor***.***

Update the getProcessedRX () method with the ProcessedResource.java code below. This will give us an endpoint of http://myappname.mybluemix.net/api/processed.

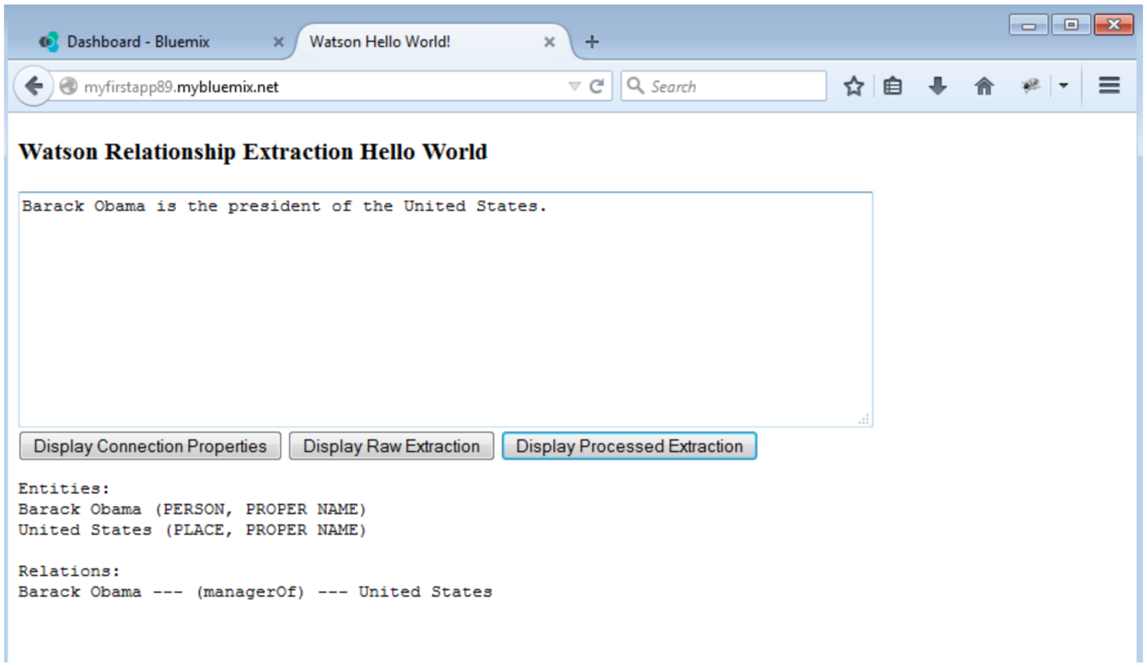
FILE: com/ibm/ic2015/ ProcessedResource.java

```
package com.ibm.ic2015;

import java.util.List;

import javax.ws.rs.FormParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

import com.ibm.ic2015.relexbind.Marshaller;
import com.ibm.ic2015.relexbind.RelExUtil;
import com.ibm.ic2015.relexbind.Rep;


@Path("/processed")
public class ProcessedResource {

      @POST
      @Produces("text/plain")
      public String getProcessedRX(@FormParam("text") String text) {

            System.out.println("Form text: " + text);

            VCAPProperties vcp = new VCAPProperties("relationship_extraction");

            String xml = RelExAPI.performExtraction(text,
                                                    "ie-en-news",
                                                    vcp.getBaseURL(),
                                                    vcp.getUsername(),
                                                    vcp.getPassword());

            Rep rep = Marshaller.marshallXml(xml);

            RelExUtil util = new RelExUtil(rep);
            List<String> entities = util.getEntitiesList();
            List<String> rels = util.getRelationsList();

            return "Entities:\n" +
                  RelExUtil.stringFromList(entities) +
                  "\nRelations:\n" +
                  RelExUtil.stringFromList(rels);

      }

}
```

5.3.2 Deploy the App

Follow the same steps as in section 5.1.3 to deploy the app.

After the app is deployed, open it in your browser. Refresh the page if necessary. Click on the ***Display Processed Extraction*** button. The formatted data will display.


Dashboard - Bluemix
Watson Hello World!
myfirstapp89.mybluemix.net
Search
Display Connection Properties
Display Raw Extraction
Display Processed Extraction

Watson Relationship Extraction Hello World

```
Barack Obama is the president of the United States.
```

```
Entities:
Barack Obama (PERSON, PROPER NAME)
United States (PLACE, PROPER NAME)

Relations:
Barack Obama --- (managerOf) --- United States
```