Using Jena in a Java EE WebApp – Live Demonstrations

(Live demonstrations done using Eclipse for Java EE 4.4 and WildFly 8.x)

The instructions below assume you have already set up Eclipse with WildFly and a database, following earlier instructions from the DWWS course.

Start from an existing project – CDI Travel:

1. Visit http://www.inf.ufes.br/~vitorsouza/en/downloads/ and download "CDITravel example" (http://www.inf.ufes.br/~vitorsouza/wp-content/uploads/java-en-tutorial-web-example-cditravel.zip);
2. Unpack it and import it to Eclipse as a project, using File > Import > Existing Projects into Workspace;
3. Right-click the project, open its Properties, go to the Targeted Runtimes section and select your WildFly server, then click OK;
4. Open JPA Content > persistence.xml, copy the name of the `jta-data-source` and make sure one is properly configured in WildFly's settings, pointing to a database that is currently running;
5. Deploy the WebApp and verify that it works.

Add Jena to the Project:

1. Right-click the CDITravel project and select Configure > Convert to Maven Project. Follow the wizard accepting the default options;
2. Open the `pom.xml` file and add the Jena dependencies according to the instructions of its website: https://jena.apache.org/download/index.cgi

```
<!-- Add this after </build> -->
  <dependencies>
   <dependency>
     <groupId>org.apache.jena</groupId>
     <artifactId>apache-jena-libs</artifactId>
     <type>pom</type>
     <version>3.3.0</version>
   </dependency>
  </dependencies>
```

3. Open WebContents/WEB-INF/web.xml and change the <web-app> opening tag to the following:

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.jcp.org/xml/ns/javaee"
   xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
   id="CDITravel" version="3.1">
```

4. Save and update the project if needed. Check that Jena JAR files were added to Java Resources > Libraries > Maven Dependencies.

Consume Linked Data:

1. Open WebContents/addPackage.xhtml and add an AJAX event to the name field:

```
<h:inputText id="name" value="#{addPackage.pack.name}" size="30">
  <f:ajax event="blur" listener="#{addPackage.suggestDescription}"
execute="@this" render="description" />
</h:inputText>
```

2. Implement the `suggestDescription()` method in the `br.ufes.inf.nemo.dev.cditravel.beans.AddPackage` controller class:

```
public void suggestDescription() {
   String name = pack.getName();
   if (name != null && name.length() > 3) {
      String query = "PREFIX dbo: <http://dbpedia.org/ontology/> " +
          "SELECT ?desc " +
          "WHERE { " +
            "?x a dbo:Place ; " +
            "rdfs:label ?name ; " +
            "dbo:abstract ?desc . " +
            "FILTER (?name = \"" + name + "\"@en) " +
            "FILTER (langMatches(lang(?desc), \"EN\")) " +
          "}";

      QueryExecution queryExecution =
QueryExecutionFactory.sparqlService("http://dbpedia.org/sparql", query);
      ResultSet results = queryExecution.execSelect();

      if (results.hasNext()) {
         QuerySolution querySolution = results.next();
         Literal literal = querySolution.getLiteral("desc");
         pack.setDescription("" + literal.getValue());
      }
   }
}
```

3. Add a progress indicator:
   a. Download an animated gif from http://preloaders.net/en/circular, place it under WebContent/files/images/ajaxloader.gif;
   b. Add it to the decorator at the end of `<div id="header">`, together with a script that turns it on/off:

```
<script>
jsf.ajax.addOnEvent(function(data) {
        var ajaxstatus = data.status;
        var ajaxloader = document.getElementById("ajaxloader");
        switch (ajaxstatus) {
            case "begin":
                ajaxloader.style.display = 'block';
                break;
            case "complete":
                ajaxloader.style.display = 'none';
                break;
            case "success":
                break;
        }
    });
</script>
<img id="ajaxloader"
src="#{facesContext.externalContext.requestContextPath}/files/images/ajaxloader
.gif" style="display: none; float: right;" />
```

4. Deploy the application again and test it.

Publish Linked Data:

1. Add a link to the addPackages.xhtml page, right before the <h:dataTable /> tag:

```
<a style="float: right;"
href="#{facesContext.externalContext.requestContextPath}/data/tourpackages">RDF
/XML</a>
```

2. Implement the Servlet that will produce the RDF/XML output:

```
@WebServlet(urlPatterns = { "/data/tourpackages" })
public class ListPackagesInRdfServlet extends HttpServlet {
   private static final DateFormat df = new SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss");

   @EJB
   private TourPackageDAO tourPackageDAO;

   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
      resp.setContentType("text/xml");

      List<TourPackage> packs = tourPackageDAO.retrieveAll();

      Model model = ModelFactory.createDefaultModel();
      String myNS = "http://localhost:8080/CDITravel/data/TourPackage/";
      String grNS = "http://purl.org/goodrelations/v1#";
      model.setNsPrefix("gr", grNS);
      Resource grOffering = ResourceFactory.createResource(grNS + "Offering");
      Resource grPriceSpecification = ResourceFactory.createResource(grNS +
"PriceSpecification");
      Property gravailabilityStarts = ResourceFactory.createProperty(grNS +
"availabilityStarts");
      Property gravailabilityEnds = ResourceFactory.createProperty(grNS +
"availabilityEnds");
      Property grhasPriceSpecification = ResourceFactory.createProperty(grNS +
"hasPriceSpecification");
      Property grhasCurrencyValue = ResourceFactory.createProperty(grNS +
"hasCurrencyValue");

      for (TourPackage pack : packs) {
         model.createResource(myNS + pack.getId())
            .addProperty(RDF.type, grOffering)
            .addProperty(RDFS.label, pack.getName())
            .addProperty(RDFS.comment, pack.getDescription())
            .addLiteral(gravailabilityStarts,
ResourceFactory.createTypedLiteral(df.format(pack.getBegin()),
XSDDatatype.XSDdateTime))
            .addLiteral(gravailabilityEnds,
ResourceFactory.createTypedLiteral(df.format(pack.getEnd()),
XSDDatatype.XSDdateTime))
            .addProperty(grhasPriceSpecification, model.createResource()
               .addProperty(RDF.type, grPriceSpecification)
               .addLiteral(grhasCurrencyValue, pack.getPrice().floatValue()));
      }

      try (PrintWriter out = resp.getWriter()) {
         model.write(out, "RDF/XML");
      }
   }

}
```