```
#
#   Ako na novu komponentu
#
#   Matej Snoha, rev. 1
#



#########
1) Uvod
#########

Komponenty su realizovane ako Java Servlet.
Nevychadzame vsak priamo z triedy HttpServlet, ale z nej zdedenej abstraktnej triedy
 AbstractComponent.

Tato prinasa nastroje na ziskanie konfiguracie z databazy (trieda
ComponentConfiguration),
ziskanie informacii o prihlasenom uzivatelovi a jeho preferenciach (trieda
UserContext) a prislusne funkcie.



########################################
2) Kostra novej komponenty DemoComponent
########################################

Minimalna funkcna komponenta sa ziska zdedenim z AbstractComponent a prepisanim
doGetPost() napriklad takto:

package cz.opendata.tenderstats;

import java.io.IOException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class DemoComponent extends AbstractComponent {

    @Override
    protected void doGetPost(HttpServletRequest request, HttpServletResponse
    response) throws IOException {
        response.getWriter().println("Hello linked data world");
    }
}

Dalej je potrebne komponente poskytnut pristup k relacnej databazi.
Na to je potrebne vytvorit uzivatela v relacnej databazi a ulozit prihlasovacie
udaje
do deployment descriptoru (subor WebContent/WEB-INF/web.xml)

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns=
"http://java.sun.com/xml/ns/javaee" xmlns:web=
"http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" xsi:schemaLocation=
"http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
  <servlet>
    <servlet-name>DemoComponent</servlet-name>
    <servlet-class>cz.opendata.tenderstats.DemoComponent</servlet-class>
```

```
    <init-param>
      <param-name>rdbAddress</param-name>
      <param-value>jdbc:mysql://gd.projekty.ms.mff.cuni.cz:3306/</param-value>
    </init-param>
    <init-param>
      <param-name>rdbDatabase</param-name>
      <param-value>tenderstats</param-value>
    </init-param>
    <init-param>
      <param-name>rdbUsername</param-name>
      <param-value>ts_DemoComponent</param-value>
    </init-param>
    <init-param>
      <param-name>rdbPassword</param-name>
      <param-value>****************</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>DemoComponent</servlet-name>
    <url-pattern>/DemoComponent/*</url-pattern>
  </servlet-mapping>
</web-app>

*/
##############################
3) Pristup k SPARQL endpointu
##############################

Na pracu s RDF datami a SPARQL dotazy mozte pouzit kniznicu Apache Jena.
Adresy endpointov (jeden pre sukromne - nezverejnene zakazy a jeden pre verejne)
ziskate ako

config.getSparqlPrivateQuery()
config.getSparqlPrivateUpdate()
config.getSparqlPublicQuery()
config.getSparqlPublicUpdate()

SPARQL SELECT dotaz na private endpoint by vyzeral napriklad takto:

Query query = QueryFactory.create("SELECT ?s ...");
ResultSet rs = QueryExecutionFactory.sparqlService(config.getSparqlPrivateQuery(),
query).execSelect();
while (rs.hasNext()) {
    QuerySolution row = rs.next();
    // spracovat row.get("s").toString()
}


##############################
4) Pristup k relacnej databazi
##############################

Data v databaze tenderstats, tabulke component_preferences patriace aktualnej
komponente
su spristupnene na citanie cez

config.getPreference(String)
```

Pre prisup pomocou JDBC sluzia nastavenia ulozene vo web.xml, ktore su spristupnene pod

```
config.getRdbAddress()
config.getRdbDatabase()
config.getRdbUsername()
config.getRdbPassword()
```

SQL SELECT dotaz by vyzeral napriklad takto:

```
Connection con = DriverManager.getConnection(config.getRdbAddress() + config.
getRdbDatabase(),
    config.getRdbUsername(), config.getRdbPassword());
PreparedStatement pst = con.prepareStatement("SELECT preference, value FROM
user_preferences WHERE username=?");
pst.setString(1, "demo");
ResultSet rs = pst.executeQuery();
while (rs.next()) {
    // spracuj rs.getString("preference"), rs.getString("value")
}
```

```
####################
5) Demo komponenta
####################
```

Takto by vyzerala jednoducha komponenta, ktora pri HTTP dotaze na DemoComponent/?action=getPrivateContracts
vrati JSON reprezentaciu zakaziek aktualne prihlaseneho uzivatela.

Metody init() a destroy() je mozne vynechat.
Je vhodne ich vsak pouzit na uchovavanie stavu komponenty alebo spravu zdrojov.

```
package cz.opendata.tenderstats;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

// Google GSON na generovanie JSON
import com.google.gson.Gson;

// Apache Jena na pracu s RDF
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QueryFactory;
import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;

public class DemoComponent extends AbstractComponent {
```

```
    // tuto triedu pouzijeme na rychle generovanie JSON v pozadovanom formate
    class ContractTableRow {

        String contractURI;
        String title;
        String price;
        String currency;

        public ContractTableRow(String contractURI, String title, String price,
        String currency) {
            this.contractURI = contractURI;
            this.title = title;
            this.price = price;
            this.currency = currency;
        }
    }

    private static final long serialVersionUID = 1L;

    @Override
    public void init() throws ServletException {
        super.init(); // zabezpeci inicializaciu statickej premennej config s
        konfiguraciou komponenty z databazy
        // tento kod bude zavolany raz pri starte servletu este pred prvou HTTP
        poziadavkou
    }

    @Override
    public void destroy() {
        // tento kod bude zavolany pri vypnuti servletu, teda napriklad pri
        normalnom vypnuti Servlet containeru
        // (ako Tomcat), pri migracii servletu na iny stroj, pri nahradeni
        beziaceho servletu novou verziou, ...
        super.destroy(); // momentalne nerobi vobec nic, ale ak by v buducnosti
        bolo treba :)
    }

    @Override
    protected void doGetPost(HttpServletRequest request, HttpServletResponse
    response) throws IOException,
            ServletException {
        // tento kod bude zavolany pri HTTP poziadavke (z roznych vlaken pre rozne
        poziadavky)
        // request obsahuje HTTP poziadavku od klienta - URL, parametre, hlavicky,
        cookies, session ...
        // response sluzi na posielanie dat klientovi, ako napr. HTTP status kod,
        ine hlavicky, telo
        if (isUserLoggedIn(request)) {
            String action = request.getParameter("action");
            if (action == null) {
                response.sendError(400);
                return;
            }
            switch (action) {
                case "getPrivateContracts":
                    List<ContractTableRow> contracts = getPrivateContracts(request);
                    Gson gson = new Gson();
```

```
                    response.setContentType("application/json; charset=UTF-8");
                    response.getWriter().println(gson.toJson(contracts));
                    break;
                default:
                    response.sendError(400);
                    break;
            }
        } else {
            response.sendError(403, "No user logged in.");
        }
    }

    // pripoji sa na SPARQL endpoint a vrati zakazky aktualne prihlaseneho uzivatela
    protected List<ContractTableRow> getPrivateContracts(HttpServletRequest request)
     {
        UserContext uc = getUserContext(request);
        Query query = QueryFactory.create(
                "PREFIX gr:       <http://purl.org/goodrelations/v1#>  " +
                "PREFIX pc:       <http://purl.org/procurement/public-contracts#>  "
                 +
                "PREFIX dc:       <http://purl.org/dc/terms/>  " +
                "SELECT ?contractURI ?title ?price ?currency " +
                "WHERE " +
                "{ " +
                "   GRAPH <" + uc.getNamedGraph() + "> " +
                "   { " +
                "       ?contractURI    a                   pc:Contract; " +
                "                       dc:title            ?title; " +
                "                       pc:estimatedPrice   ?priceURI " +
                "       . " +
                "       ?priceURI       gr:hasCurrencyValue ?price; " +
                "                       gr:hasCurrency      ?currency " +
                "   } " +
                "}");
        ResultSet rs = QueryExecutionFactory.sparqlService(config.
        getSparqlPrivateQuery(), query).execSelect();
        List<ContractTableRow> table = new ArrayList<>();
        while (rs.hasNext()) {
            QuerySolution row = rs.next();
            table.add(new ContractTableRow(row.get("contractURI").toString(), row.
            get("title").toString(), row
                    .get("price").toString(), row.get("currency").toString()));
        }
        return table;
    }
}
```

Tato komponenta moze vypisat napriklad:


```
[
   {
      "contractURI":
      "http://ld.opendata.cz/resource/isvzus.cz/public-contract/226193-7202020026193-
      0028c6b1-a0c4-470c-89ff-f33d8fe63871",
      "title":"Software pro reversní inženýrství ve strojírenství@cs",
      "price":"399200^^http://www.w3.org/2001/XMLSchema#float",
      "currency":"CZK"
```

```
   },
   {
      "contractURI":
      "http://ld.opendata.cz/resource/isvzus.cz/public-contract/226051-7202012033006-
      b52b09da-16d1-4b64-88db-0c853c011f6e",
      "title":"TČ – práce s motorovou pilou, polesí č. 43 Višňová@cs",
      "price":"80000,00^^http://www.w3.org/2001/XMLSchema#float",
      "currency":"CZK"
   }
]
```


Na spracovanie v prehliadaci je napriklad jQuery.getJSON()
http://api.jquery.com/jQuery.getJSON/