[TP J2EE] Notes de révision

Connexion et deconnexion à une base de données

```
public void init() throws DaoException {
  try {
    Class.forName("org.postgresql.Driver").newInstance();
  } catch (Exception e) {
    throw new DaoException("Erreur d'initialisation de la couche DAO : Impossible de chager le driver. " + e, 1);
  }
  String url ="jdbc:postgresql://postgres/clinique?user=login&password=pass" ;
  try {
    connection = DriverManager.getConnection(url);
  } catch (SQLException e) {
    e.printStackTrace();
    System.exit(1);
  }

  try {
    // Prepared statements
    st1 = connection.prepareStatement("SELECT * FROM eleves WHERE id=?");
  } catch (SQLException e) {
    e.printStackTrace();
  }
}

public void destroy(){
   try {
     st1.close();
     st2.close();
     st3.close();
     st4.close();
    connection.close();
  } catch( Exception e ) {
    System.err.println("Problème de fermeture de la connexion à la base de données");
  }
}
```

Utilisation de PreparedStatement

```
@Override
public Eleve getOne(int id) {
  Eleve eleve = null;

  try {
    st1.setInt(1, id);
    ResultSet curs;
    curs = st1.executeQuery();

    int i=0;
    while(curs.next()){
      i++;
      eleve = new Eleve(curs.getInt("id"),curs.getInt("version"),curs.getString("nom"),
          curs.getString("prenom"),curs.getDate("datenaissance"),curs.getBoolean("redoublant"),
          curs.getInt("annee"),curs.getString("filiere"));
    }

    if(i==0)
    {
      throw new DaoException("Eleve non existant", 3);
    }
  } catch (SQLException ex) {
    throw new DaoException("Erreur de recherche dans la couche [dao] : Erreur SQL. "+ex, 2);
  }
  return eleve;
}
```

Execution d'une requète SQL

```
@Override
public Collection<Eleve> getAll() {
  ArrayList<Eleve> res = new ArrayList<>();
  Statement requete;
  try {
    requete = connection.createStatement();
    ResultSet rs = requete.executeQuery("SELECT * FROM eleves;");
    while(rs.next()){
      Eleve e = new Eleve();
      e.setId(rs.getInt("id"));
      e.setVersion(rs.getInt("version"));
      e.setLastName(rs.getString("nom"));
      e.setFirstName(rs.getString("prenom"));
      e.setBirthDate(rs.getDate("datenaissance"));
      e.setRedoublant(rs.getBoolean("redoublant"));
      e.setAnnee(rs.getInt("annee"));
      e.setFiliere(rs.getString("filiere"));
      res.add(e);
    }
  } catch (SQLException e) {
    throw new DaoException("Erreur de recherche dans la couche [dao] : Erreur SQL. "+ e, 2);
  }
  return res;
}
```

Différence executeQuery() et executeUpdate()

executeQuery() attend une réponse de la requète alors que *executeUpdate()* n'attend rien. Pour toutes les requètes sans résultat il faut alors faire *executeUpdate()*. Par exemple : *UPDATE* et *DELETE*.

Tests unitaire

```
public void test() {
  Date date = null;
  int code = 0;
  try {
    date = new SimpleDateFormat("dd/MM/yyyy").parse("18/04/1992");
  } catch (ParseException e1) {
    e1.printStackTrace();
  }

  //Modification élève non existant
  Eleve Nv_eleve1 = new Eleve(100, 1, "B", "B", date, false, 3, "INFO");
  try {
    dao.saveOne(Nv_eleve1);
  } catch(DaoException e) {
    code = e.getCode();
  }
  assertEquals(2, code);

  //Suppression élève non existant
  try {
    dao.deleteOne(100);
  } catch(DaoException e) {
    code = e.getCode();
  }
  assertEquals(2, code);
}
```

Gestion de l'URL

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

  String path = request.getPathInfo();
  if(path==null){
    path = "/list";
    return;
  }

  if(path.equals("/list")){
    ArrayList<Eleve> listeEleves = new ArrayList<Eleve>();
    listeEleves = (ArrayList<Eleve>) service.getAll();

    request.setAttribute("eleves", listeEleves);
    getServletContext().getRequestDispatcher("/WEB-INF/vues/list.jsp").forward(request, response);

    return;
  }

  if(path.equals("/delete")){
    int id = Integer.parseInt(request.getParameter("id"));

    service.deleteOne(id);
    response.sendRedirect("list");

    return;
  }

  if(path.equals("/edit")){
    int id = Integer.parseInt(request.getParameter("id"));
    Eleve eleve = null;

    if(id !=-1){
      eleve = service.getOne(id);
    }else{
      eleve = new Eleve();
      eleve.setId(-1);
      eleve.setVersion(1);
      Date d = new Date();
      eleve.setBirthDate(d);
    }

    request.setAttribute("eleve", eleve);
    getServletContext().getRequestDispatcher("/WEB-INF/vues/edit.jsp").forward(request, response);
    return;
  }


  if(path.equals("/validate")){
    doValidateEleve(request,response);
    return;
  }
}

public void doValidateEleve(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
  String prenom = request.getParameter("prenom").trim();
  String nom = request.getParameter("nom").trim();
  int annee = Integer.parseInt(request.getParameter("annee").trim());
  String filiere = request.getParameter("filiere").trim();
  int id = Integer.parseInt(request.getParameter("id"));
  int version = Integer.parseInt(request.getParameter("version"));
```

```
  Boolean redoublant = Boolean.valueOf(request.getParameter("redoublant"));

  Date dateNaissance = null;
  Eleve eleve = null;

  try {
    dateNaissance = new SimpleDateFormat("yyyy-MM-dd").parse(request.getParameter("dateNaissance"));
    eleve = new Eleve(id, version, nom, prenom, dateNaissance,redoublant,annee,filiere);
  } catch (ParseException e) {
    e.printStackTrace();
  }

  service.saveOne(eleve);
  response.sendRedirect("list");
}
```

Spring

student-postgres.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap PUBLIC "-//ibatis.apache.org//DD SQL MAP 2.0//EN"
          "http://ibatis.apache.org/dtd/sql-map-2.dtd">
<sqlMap>
    <!-- alias class [Student] -->
    <typeAlias alias="Student.class" type="ensicaen.obo.mvc.entities.Student"/>
    <!-- mapping table [STUDENT] - objet [Eleves] -->
    <resultMap id="Student.map" class="Student.class">
        <result property="id" column="ID"/>
        <result property="version" column="VERSION"/>
        <result property="name" column="NAME"/>
        <result property="firstname" column="FIRSTNAME"/>
        <result property="birthdate" column="BIRTHDATE"/>
        <result property="redoublers" column="REDOUBLER"/>
        <result property="year" column="YEAR"/>
        <result property="branch" column="BRANCH"/>
    </resultMap>
    <!-- liste de tous les eleves -->
    <select id="Student.getAll" resultMap="Student.map">
        SELECT * FROM rpereira.student
    </select>
    <!-- obtenir un eleve en particuliers -->
    <select id="Student.getOne" parameterClass="int" resultMap="Student.map">
        SELECT * FROM rpereira.student WHERE ID=#value#
    </select>
    <!-- ajouter un eleve -->
    <insert id="Student.insertOne" parameterClass="Student.class">
       <selectKey keyProperty="id">
           SELECT nextval('rpereira.seq_eleves') AS value
        </selectKey>
        INSERT INTO rpereira.student VALUES(#id#,#version#,#name#,#firstname#,#birthdate#,#redoublers#,#year#,#branch#);
    </insert>
    <!-- mettre à jour un eleve -->
    <insert id="Student.updateOne" parameterClass="Student.class">
        UPDATE rpereira.student set VERSION=VERSION+1, NAME=#name#, FIRSTNAME=#firstname#, BIRTHDATE=#birthdate#, REDOUBLER=#redoublers#, YEAR=#year#, BRANCH=#branch# WHERE ID=
    </insert>
    <!-- supprimer un eleve -->
    <insert id="Student.deleteOne" parameterClass="int">
        DELETE FROM rpereira.student where ID=#value#
    </insert>
</sqlMap>
```

sql-map-config-postgres.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMapConfig PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
           "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">
<sqlMapConfig>
    <sqlMap resource="student-postgres.xml"/>
</sqlMapConfig>
```

spring-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans     xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
    <!-- la class dao -->
    <bean id="dao" class="ensicaen.obo.mvc.dao.DaoImpl" init-method="init" destroy-method="destroy"/>
    <!-- la class service -->
    <bean id="service" class="ensicaen.obo.mvc.service.ServiceImpl">
       <property name="dao">
           <ref local="dao"/>
       </property>
    </bean>
    <!-- la source de données utilisant JDBC -->
    <bean id="dataSource" class="org.springframework.jdbc.datasource.SingleConnectionDataSource" destroy-method="destroy">
        <property name="url" value="jdbc:postgresql://postgres/clinique"/>
        <property name="username" value="rpereira"/>
        <property name="password" value="toto"/>
        <property name="driverClassName" value="org.postgresql.Driver"/>
    </bean>
    <!-- SqlMapClient -->
    <bean id="sqlMapClient" class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
        <property name="dataSource">
```

```
            <ref local="dataSource"/>
        </property>
        <property name="configLocation" value="sql-map-config-postgres.xml"/>
    </bean>
    <!-- la classe d'accès à la couche [dao] -->
    <bean id="daoCommon" class="ensicaen.obo.mvc.dao.DaoImplCommon" init-method="init" destroy-method="destroy">
        <property name="sqlMapClient">
            <ref local="sqlMapClient"/>
        </property>
    </bean>
    <!-- gestionnaire de transaction -->
    <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource">
            <ref local="dataSource"/>
        </property>
    </bean>
    <!-- service 2.0  -->
    <bean id="serviceCommon" class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
           <property name="transactionManager">
               <ref local="transactionManager"/>
           </property>
          <property name="target">
               <bean class="ensicaen.obo.mvc.service.ServiceImpl">
                   <property name="daoCommon">
                   <ref local="daoCommon"/>
               </property>
           </bean>
           </property>
           <property name="transactionAttributes">
               <props>
                   <prop key="saveMany">PROPAGATION_REQUIRED</prop>
                   <prop key="deleteMany">PROPAGATION_REQUIRED</prop>
               </props>
           </property>
    </bean>
</beans>
```