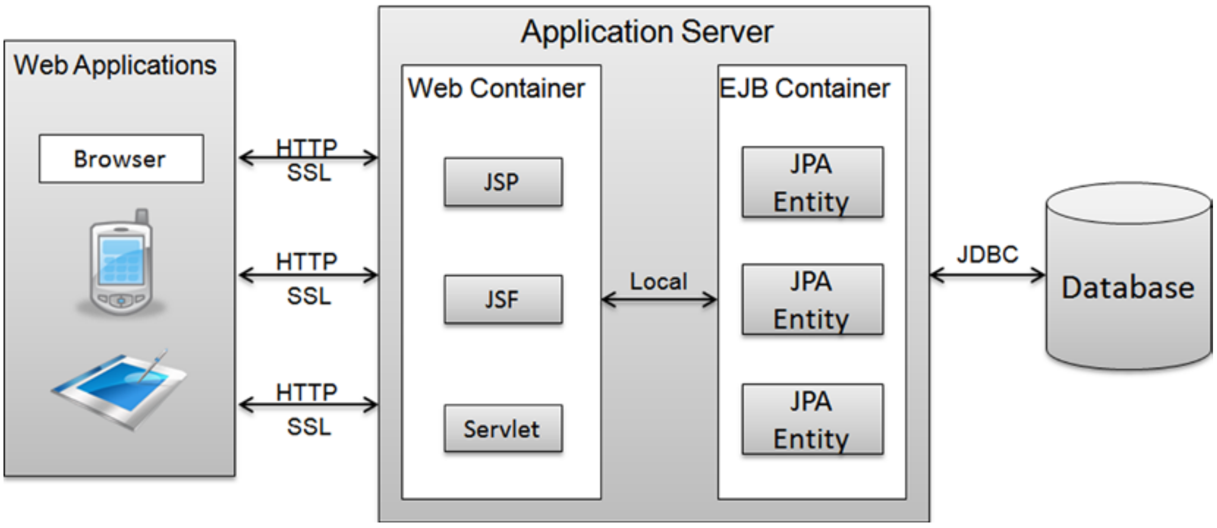
LAB_PROJET

Objectifs : construire une application Java EE avec JPA + EJB 3 + JSF + JAAS



Etape 1 Lab JPA

1-Objet User

```
@Entity
@Table(name = "USERS")
@NamedQuery(name="User.findUserByEmail", query="select u from User u where u.email =
:email")
public class User {

    public static final String FIND_BY_EMAIL = "User.findUserByEmail";

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private int id;

    @Column(unique = true)
    private String email;
    private String password;
    private String name;
    private String role;

//la suite faire des getters et setter

   @Override
    public boolean equals(Object obj) {
        if(obj instanceof User){
            User user = (User) obj;
            return user.getEmail().equals(getEmail());
        }

        return false;
    }
```

2-Objet Dogs

```
@Entity
@Table(name = "DOGS")
public class Dog {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String name;
    private double weight;
//GETTERS ET SETTERS

@Override
    public boolean equals(Object obj) {

        if(obj instanceof Dog){
            Dog dog = (Dog) obj;
            return dog.getId() == getId();
        }

        return false;
    }
```

Remarques :
La classe d'utilisateur a un champ nommé «rôle» qui va stocker le niveau de l'utilisateur de rôle. L'email sera unique; ce sera l'identifiant de connexion. Notez qu'une classe à être déclaré comme entité, il ne doit les annotations suivantes: "Entity" et "Id".

Etape 2 Couche Business DAO :

```
public abstract class GenericDAO<T> {
    private final static String UNIT_NAME = "CrudPU";

    @PersistenceContext(unitName = UNIT_NAME)
    private EntityManager em;

    private Class<T> entityClass;

    public GenericDAO(Class<T> entityClass) {
        this.entityClass = entityClass;
    }

    public void save(T entity) {
        em.persist(entity);
    }
    protected void delete(Object id, Class<T> classe) {
        T entityToBeRemoved = em.getReference(classe, id);

        em.remove(entityToBeRemoved);
    }

    public T update(T entity) {
        return em.merge(entity);
    }

    public T find(int entityID) {
        return em.find(entityClass, entityID);
    }
// Using the unchecked because JPA does not have a
    // em.getCriteriaBuilder().createQuery()<T> method
    @SuppressWarnings({ "unchecked", "rawtypes" })
    public List<T> findAll() {
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
        cq.select(cq.from(entityClass));
        return em.createQuery(cq).getResultList();
    }

    // Using the unchecked because JPA does not have a
    // ery.getSingleResult()<T> method
    @SuppressWarnings("unchecked")
    protected T findOneResult(String namedQuery, Map<String, Object> parameters) {
        T result = null;

        try {
            Query query = em.createNamedQuery(namedQuery);

            // Method that will populate parameters if they are passed not null and empty
            if (parameters != null && !parameters.isEmpty()) {
                populateQueryParameters(query, parameters);
            }

            result = (T) query.getSingleResult();

        } catch (Exception e) {
            System.out.println("Error while running query: " + e.getMessage());
            e.printStackTrace();
        }

        return result;
    }

    private void populateQueryParameters(Query query, Map<String, Object> parameters) {

        for (Entry<String, Object> entry : parameters.entrySet()) {
            query.setParameter(entry.getKey(), entry.getValue());
        }
    }
}
```

ETAPE 3 : mise en place des EJB DAO

```
@Stateless
public class DogDAO extends GenericDAO<Dog> {

    public DogDAO() {
        super(Dog.class);
    }
}
```

et ensuite un autre EJB DAO

```
@Stateless
public class UserDAO extends GenericDAO<User> {

    public UserDAO() {
        super(User.class);
    }

    public User findUserByEmail(String email){
        Map<String, Object> parameters = new HashMap<String, Object>();
        parameters.put("email", email);

        return super.findOneResult(User.FIND_BY_EMAIL, parameters);}
```

CREATION DE LA COUCHE BUSINESS FACADE

```
@Local
public interface DogFacade {

    public abstract void save(Dog dog);

    public abstract Dog update(Dog dog);

    public abstract void delete(Dog dog);

    public abstract Dog find(int entityID);

    public abstract List<Dog> findAll();
}
```

Son implémentation est :

```
@Stateless
public class DogFacadeImp implements DogFacade {
```

```
@EJB
private DogDAO dogDAO;

@Override
public void save(Dog dog) {
   isDogWithAllData(dog);

   dogDAO.save(dog);
}

@Override
public Dog update(Dog dog) {
   isDogWithAllData(dog);

   return dogDAO.update(dog);
}

@Override
public void delete(Dog dog) {
   dogDAO.delete(dog);
}

@Override
public Dog find(int entityID) {
   return dogDAO.find(entityID);
}

@Override
public List<Dog> findAll() {
   return dogDAO.findAll();
}

private void isDogWithAllData(Dog dog){
   boolean hasError = false;

   if(dog == null){
      hasError = true;
   }

   if (dog.getName() == null || "".equals(dog.getName().trim())){
      hasError = true;
   }

   if(dog.getWeight() <= 0){
      hasError = true;
   }

   if (hasError){
```

```
            throw new IllegalArgumentException("The dog is missing data. Check the name and weight,
they should have value.");
        }
    }
}
```

Ensuite nous définissons la façade pour utilisateurs

```
@Local
public interface UserFacade {
    public User findUserByEmail(String email);
}
```

Son implémentation,

```
@Stateless
public class UserFacadeImp implements UserFacade {

    @EJB
    private UserDAO userDAO;

    public User findUserByEmail(String email) {
        return userDAO.findUserByEmail(email);
    }
}
```

Business – Datasource

Définir une dataSource dans JBOSS AS 7 pour MYSQL ou PostGreSQl

1-(voir LAB +création DataSprceMYSQL)
2- https://developer.jboss.org/wiki/JBossAS7-DatasourceConfigurationForPostgresql(voir exemple)

Configuration XML du projet JPA

```
    <persistence-unit name="CrudPU" transaction-type="JTA">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>
        <jta-data-source>java:/CrudDS</jta-data-source>
        <properties>
            <property name="hibernate.hbm2ddl.auto" value="update"/>
        </properties>
    </persistence-unit>
</persistence>
```

CREATION DES VUES

Dans le projet WEB, nous aurons les caractéristiques suivantes

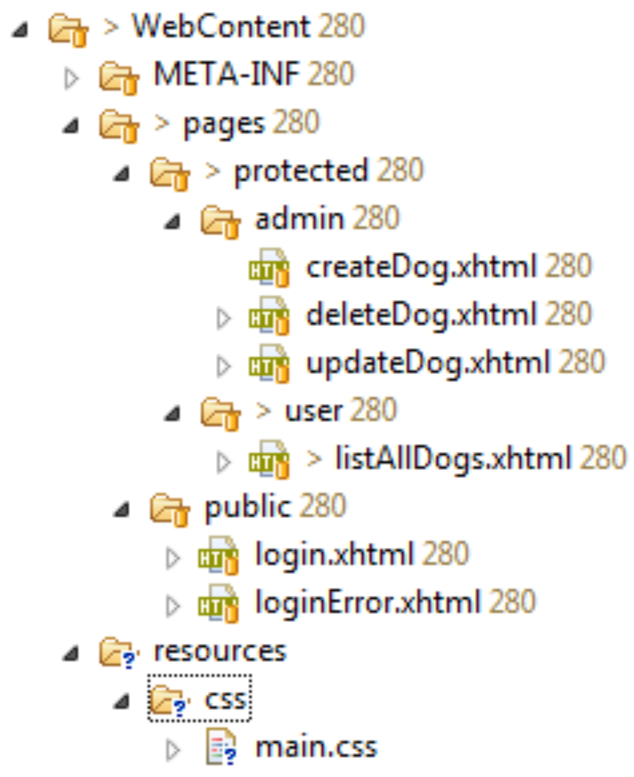
1-Le managedBean DOG

```
@ManagedBean
@RequestScoped
public class DogMB {

    @EJB
    private DogFacade dogFacade;

    private static final String CREATE_DOG = "createDog";
    private static final String DELETE_DOG = "deleteDog";
    private static final String UPDATE_DOG = "updateDog";
    private static final String LIST_ALL_DOGS = "listAllDogs";
    private static final String STAY_IN_THE_SAME_PAGE = null;

    private Dog dog;

    public Dog getDog() {

        if(dog == null){
            dog = new Dog();
        }

        return dog;
    }

    public void setDog(Dog dog) {
        this.dog = dog;
    }

    public List<Dog> getAllDogs() {
        return dogFacade.findAll();
    }

    public String updateDogStart(){
        return UPDATE_DOG;
    }

    public String updateDogEnd(){
        try {
            dogFacade.update(dog);
        } catch (EJBException e) {
            sendErrorMessageToUser("Error. Check if the weight is above 0 or call the adm");
            return STAY_IN_THE_SAME_PAGE;
        }

        sendInfoMessageToUser("Operation Complete: Update");
        return LIST_ALL_DOGS;
    }

    public String deleteDogStart(){
        return DELETE_DOG;
    }

    public String deleteDogEnd(){
        try {
            dogFacade.delete(dog);
        } catch (EJBException e) {
            sendErrorMessageToUser("Error. Call the ADM");
            return STAY_IN_THE_SAME_PAGE;
        }

        sendInfoMessageToUser("Operation Complete: Delete");

        return LIST_ALL_DOGS;
    }

    public String createDogStart(){
        return CREATE_DOG;
    }

    public String createDogEnd(){
        try {
            dogFacade.save(dog);
        } catch (EJBException e) {
            sendErrorMessageToUser("Error. Check if the weight is above 0 or call the adm");

            return STAY_IN_THE_SAME_PAGE;
        }

        sendInfoMessageToUser("Operation Complete: Create");

        return LIST_ALL_DOGS;
    }

    public String listAllDogs(){
        return LIST_ALL_DOGS;
    }

    private void sendInfoMessageToUser(String message){
        FacesContext context = getContext();
        context.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_INFO, message,
message));
    }

    private void sendErrorMessageToUser(String message){
        FacesContext context = getContext();
        context.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, message,
message));
    }

    private FacesContext getContext() {
        FacesContext context = FacesContext.getCurrentInstance();
        return context;
    }
}
```

EXEMPLE DE RECHERCHE DE BEAN PAR LOOK UP

```
@Stateless
@LocalBinding(jndiBinding="MyBean")
public class MyBeanImp implements MyBean{
    @Override
    public String hello() {
        return "Value From EJB";
    }
}
// In your Servlet class you would lookup like the code bellow:
public class Inject extends HttpServlet {

    private MyBean local;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        try {
            InitialContext iniCtx = new InitialContext();
            local = (MyBean) iniCtx.lookup("MyBean");
        } catch (NamingException e) {
            e.printStackTrace();
        }

        System.out.println(local.hello());
        request.getRequestDispatcher("/finish.jsp").forward(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    }
}
```

ECRIRE VOS VUES
Ceci n'est pas forcement votre structure de projet mais un aperçu de ce que doit être à ce stade votre projet.



```
WebContent 280
  META-INF 280
  pages 280
    protected 280
      admin 280
        createDog.xhtml 280
        deleteDog.xhtml 280
        updateDog.xhtml 280
      user 280
        listAllDogs.xhtml 280
    public 280
      login.xhtml 280
      loginError.xhtml 280
  resources
    css
      main.css
```

Création de votre CSS principale

```
.table {
   border-collapse: collapse;
}

.tableColumnsHeader {
   text-align: center;
   background: none repeat scroll 0 0 #E5E5E5;
   border-bottom: 1px solid #BBBBBB;
   padding: 16px;
}

.tableFirstLine {
   text-align: center;
   background: none repeat scroll 0 0 #F9F9F9;
   border-top: 1px solid #BBBBBB;
}

.tableNextLine {
   text-align: center;
   background: none repeat scroll 0 0 #FFFFFFF;
   border-top: 1px solid #BBBBBB;
}

.panelGrid {
   border: 1px solid;
}

.panelFirstLine {
   text-align: center;
   border-top: 1px solid #BBBBBB;
}

.panelNextLine {
   text-align: center;
   border-top: 1px solid #BBBBBB;
}
```

Les vues :

Login.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:ui="http://java.sun.com/jsf/facelets">
<h:head>
   <h:outputStylesheet library="css" name="main.css" />
</h:head>
<h:body>
   <p>Login to access secure pages:</p>
   <form method="post" action="j_security_check">
      <h:messages layout="table" errorStyle="background: #AFEEEE;"
         infoStyle="background: #AFEEEE;" globalOnly="true" />
      <h:panelGrid columns="2">
         <h:outputLabel value="Username: " />
         <input type="text" id="j_username" name="j_username" />
         <h:outputLabel value="Password: " />
         <input type="password" id="j_password" name="j_password" />
         <h:outputText value="" />
         <h:panelGrid columns="1">
            <input type="submit" name="submit" value="Login" />
         </h:panelGrid>
      </h:panelGrid>
      <br />
   </form>
</h:body>
</html>
```

LoginError.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:ui="http://java.sun.com/jsf/facelets">
<h:head>
   <h:outputStylesheet library="css" name="main.css" />
</h:head>
<h:body>
   #{msgs.loginErrorMessage}
</h:body>
</html>
```

ListAllDogs.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:ui="http://java.sun.com/jsf/facelets">
<h:head>
   <h:outputStylesheet library="css" name="main.css" />
</h:head>
<h:body>
   <h:form>
      <h3>#{msgs.loginHello}: #{userMB.user.name} || <h:commandLink
action="#{userMB.logOut()}" value="#{msgs.logout}" /> </h3>

      <h:messages />
      <h:dataTable value="#{dogMB.allDogs}" var="dog" styleClass="table"
headerClass="tableColumnsHeader" rowClasses="tableFirstLine,tableNextLine" >
         <h:column>
            <f:facet name="header">
               #{msgs.dogName}
            </f:facet>

            #{dog.name}
         </h:column>
         <h:column>
            <f:facet name="header">
               #{msgs.dogWeight}
            </f:facet>

            #{dog.weight}
         </h:column>
```

```
            <h:column>
                <h:panelGrid columns="2">
                    <!-- Always save the id as hidden when you use a request scope MB -->
                    <h:inputHidden value="#{dog.id}" />

                    <h:commandButton action="#{dogMB.updateDogStart()}" value="#{msgs.update}"
rendered="#{userMB.userAdmin}" >
                        <f:setPropertyActionListener target="#{dogMB.dog}" value="#{dog}" />
                    </h:commandButton>
                    <h:commandButton action="#{dogMB.deleteDogStart()}" value="#{msgs.delete}"
rendered="#{userMB.userAdmin}" >
                        <f:setPropertyActionListener target="#{dogMB.dog}" value="#{dog}" />
                    </h:commandButton>
                </h:panelGrid>
            </h:column>
        </h:dataTable>
        <!-- This button is displayed to the user, just to you see the error msg  -->
        <h:commandButton action="createDog" value="#{msgs.create} #{msgs.dog}" />
    </h:form>
</h:body>
</html>
```

CreateDog.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:ui="http://java.sun.com/jsf/facelets">
<h:head>
    <h:outputStylesheet library="css" name="main.css" />
</h:head>
<h:body>
    <h:form>
        <h:messages/>

        <h3>${msgs.dogCreateHeader}</h3>
        <h:panelGrid columns="2" styleClass="panelGrid" rowClasses="panelFirstLine,panelNextLine"
>
            <h:outputLabel for="dogName" value="#{msgs.dogName}" />
            <h:inputText id="dogName" value="#{dogMB.dog.name}" required="true"
requiredMessage="#{msgs.dogNameRequired}" />

            <h:outputLabel for="dogWeight" value="#{msgs.dogWeight}" />
            <h:inputText id="dogWeight" value="#{dogMB.dog.weight}" required="true"
requiredMessage="#{msgs.dogWeightRequired}" >
                <f:convertNumber />
            </h:inputText>
        </h:panelGrid>
        <h:panelGrid columns="2">
            <h:commandButton action="#{dogMB.createDogEnd()}" value="#{msgs.create}" />
            <h:commandButton action="#{dogMB.listAllDogs()}" value="#{msgs.cancel}"
immediate="true" />
        </h:panelGrid>
        <br/>
    </h:form>
</h:body>
</html>
```

DeleteDog.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets">
<h:head>
      <h:outputStylesheet library="css" name="main.css" />
</h:head>
<h:body>
      <h:form>
            <h:messages/>

            <h3>#{msgs.dogDeleteHeader}: #{dogMB.dog.name}?</h3>
            <h:inputHidden value="#{dogMB.dog.id}" />
            <h:panelGrid columns="2">
                  <h:commandButton action="#{dogMB.deleteDogEnd()}"
value="#{msgs.delete}" />
                  <h:commandButton action="#{dogMB.listAllDogs()}"
value="#{msgs.cancel}" immediate="true" />
            </h:panelGrid>
            <br/>
      </h:form>
</h:body>
</html>
```

UpdateDog.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets">
<h:head>
      <h:outputStylesheet library="css" name="main.css" />
</h:head>
```

```
<h:body>
      <h:form>
            <h:messages/>

            <h3>#{msgs.dogUpdateHeader}: #{dogMB.dog.name}</h3>
            <h:inputHidden value="#{dogMB.dog.id}" />
            <h:panelGrid columns="2" styleClass="panelGrid"
rowClasses="panelFirstLine,panelNextLine" >
                  <h:outputLabel for="dogName" value="#{msgs.dogName}" />
                  <h:inputText id="dogName" value="#{dogMB.dog.name}" required="true"
requiredMessage="#{msgs.dogNameRequired}" />

                  <h:outputLabel for="dogWeight" value="#{msgs.dogWeight}" />
                  <h:inputText id="dogWeight" value="#{dogMB.dog.weight}" required="true"
requiredMessage="#{msgs.dogWeightRequired}" >
                        <f:convertNumber />
                  </h:inputText>
            </h:panelGrid>
            <h:panelGrid columns="2">
                  <h:commandButton action="#{dogMB.updateDogEnd()}" value="#{msgs.update}" />
                  <h:commandButton action="#{dogMB.listAllDogs()}" value="#{msgs.cancel}"
immediate="true" />
            </h:panelGrid>
            <br/>
      </h:form>
</h:body>
</html>
```

A ce stade vous devez avoir mis en place votre projet Entity avec JPA

- **Mis en place vos Daos,**
- **Mis en place vis services avec EJB**
- **Mis en place votre projet Web**
- **Et surtout votre projet EAR**