Première coupe : Introduction à la plateforme Java EE

CREATION : 2010/04/22
MISE A JOUR : 2010/07/01
GLASSFISH:3.0.1
NETBEANS : 6.8

Ce TP de découverte de la plateforme Java EE est une traduction **adaptée** de « Your First Cup :An Introduction to the Java EE » publié le 20 octobre 2009. Le document de référence a été mis à jour pour la sortie officielle de Glassfish 3.0.1. Les commentaires par rapport aux anciennes versions du document d'origine et de netbeans (quand je les ai testées) sont données en orange encadré.

Chapitre 1 : Préparation de l'environnement

Il n'est plus nécessaire d'activer le support d'**Enterprise Server v3** dans **Netbeans**.

Ajouter Enterprise Server v3 comme serveur dans **Netbeans**

- (M) Tools > Servers
- (B) Add Server,
- Dans Server, choisir GlassFish v3, suivant
- Dans Platform location, sélectionner l'emplacement de l'installation, suivant
- Dans Domain, choisir un nom de domaine existant ou en créer un nouveau
- (B) Finish

Il n'est pas nécessaire d'installer le plugin Facelets 1.1.4 (JSF 1.2) à partir du menu Tools > Plugins. Nous utiliserons JSF 2.0.

Obtenir les dernières mises à jour du tutorial (Attention, dans ce cas, ce document n'est peut-être plus une bonne adaptation)

- Ouvrir l'onglet Services et dérouler Servers
- Cliquer avec le bouton droit sur l'instance de GlassFish, Sélectionner View Update Center
- Explorer les menus pour la culture. Le tutorial est considéré comme un package.

Chapitre 2 : Architecture de l'application

L'application à développer est constituée de 4 modules

1. DukesAgeResource est une ressource JAX-RS (service Web RESTful) qui calcule l'âge de Duke, la mascotte Java. Duke est né le 23 mai 1995 lorsque la première démo de la technologie Java a été publiquement publiée.

2. DukesBirthdayBean est un *bean* de session sans état, local, avec vue sans interface qui calcule la différence d'âge entre l'utilisateur et Duke. Les informations sur l'utilisateur sont stockées dans une entité grâce à l'API Java de persistance.
3. FirstcupUser est une entité de persistance Java qui représente la date de naissance d'un utilisateur particulier. Elle est stockée dans une table d'une base de données **JavaDB** et géré par les méthodes métier de DukesBirthdayBean.
4. L'application web firstcup est une application JavaServer Faces Facelets qui utilise DukesAgeResource pour afficher l'âge de Duke, demande l'âge de l'utilisateur, détermine qui est le plus vieux des deux grâce à DukesBirthdayBean, puis affiche le nombre d'année(s) de différence entre Duke et l'utilisateur et la différence d'âge moyenne de tous les utilisateurs

L'application web firstcup se compose de :

- greetings.xhtml : une page xhtml qui utilise les bibliothèques de balises (*tags*) JSF Facelets. Les utilsateurs peuvent saisir leur date de naissance et la soumettre pour la comparer avec celle de Duke.
- response.xhtml : une page xhtml qui affiche si l'utilisateur est plus vieux que Duke, ou pas. Cette page affiche également la différence d'âge.
- DukesBday.java : un bean JSF géré qui définit des propriétés pour se souvenir de la date de naissance de l'utilisateur, obtient l'âge de Duke grâce au web service DukesAgeResource, et donne la différence d'âge entre les deux.
- web.xml : le descripteur de déploiement de l'application, utilisé pour configurer certains aspects de l'application web lorsqu'elle est installée. Il fournit ici un *mapping* pour l'instance Facelets de l'application, qui accepte les demandes entrantes, les transmet au cycle de vie pour traitement et initialise les ressources.
- WebMessages.properties et WebMessages_fr.properties contiennent les chaînes localisées des fichiers xhtml.
- DukesBirthdayBean.java : le bean entreprise de l'application

L'application contient un composant pour le tiers Web, trois composants pour le tiers métier et accède au système d'information de l'entreprise. Le tiers client est le navigateur de l'utilisateur.

L'application va vous faire créer différents projets dans **Netbeans** que vous pouvez placer dans un répertoire maître particulier.

Chapitre 3 : Service web DukesAge

DukesAgeResource est le point d'accès (*endpoint*) d'un service RESTful simple. Les services RESTful sont souvent opposés aux services web SOAP car ils sont plus simples et plus rapides. Les services web RESTful (*representational state transfer*) utilisent le protocole HTTP pour accéder, modifier ou effacer de l'information contenue au sein d'une ressource (méthodes GET ou POST). L'information est identifiable par une URI.

Les services Web sont conçus pour être indépendants de leurs clients. Les services Web RESTful sont disponibles publiquement pour de nombreux clients dispersés sur toute la toile. C'est un couplage

lâche dans la mesure où les implémentations ne sont pas nécessaires. Le service pourrait tout à fait être déployé sur un autre serveur (et dans un autre langage).

DukesAgeresource est une ressource JAX-RS qui répond aux requêtes GET et retourne une chaîne de caractères représentant l'âge de Duke au moment de la requête.

Créer le projet

- (M) File > New Project, Categories : Java Web, Web Application
- Projet name : dukes-age
- Changer éventuellement le répertoire du projet
- Server menu : sélectionner Glassfish v3
- Java EE version : choisir Java EE 6 évidemment

Cette option n'est pas dispo sur des versions antérieures de Netbeans, dans ce cas, choisir la 5 et continuer.

- Context Path : /DukesAgeService
- (B) Finish
- Jeter un coup d'œil au fichier index.jsp pour culture. Il n'est pas nécessaire, il faut l'effacer.

Créer le point d'accès du service : la classe DukesAgeResource

- S'assurer que le projet dukes-age est bien sélectionné
- (M) File > New File (ou tenter le menu contextuel New)
- Sélectionner RESTful Web Services from Patterns dans la catégorie Web Services
- Sélectionner Simple Root Resource (vous pouvez trouver aussi Singleton dans les versions antérieures de Netbeans)
- Resource Package : firstcup.webservice
- Path : dukesAge
- Class Name: DukesAgeResource
- MIME Type : text/plain
- (B) Finish
- Laisser Netbeans s'occuper des resources

Configurer l'application Web

- Dans le menu contextuel du projet dukes-age, sélectionner Properties
- Cliquer Librairies. Décocher restlib-gfv3ee6. Cette bibliothèque n'est pas nécessaire car elle est déjà intégrée dans GlassFish.
- Cliquer Run
- Relative URL : /resources/dukesAge
- (B) OK

Adapter le code de la classe

- Supprimer la méthode putText() (ainsi que la javadoc et autres)

- Implémenter la méthode getText(). Dans le menu contextuel ou le menu Source, il est possible de formater automatiquement le code : Format et de réparer automatiquement les erreurs liées aux imports (fix imports)
- Sauvegarder le fichier

```
// Create a new Calendar for Duke's birthday
Calendar dukesBirthday = new GregorianCalendar(1995, Calendar.MAY, 23);
// Create a new Calendar for today
Calendar now = Calendar.getInstance();
// Subtract today's year from Duke's birth year, 1995
int dukesAge = now.get(Calendar.YEAR) - dukesBirthday.get(Calendar.YEAR);
dukesBirthday.add(Calendar.YEAR, dukesAge);
// If today's date is before May 23, subtract a year from Duke's age
if (now.before(dukesBirthday)) {
   dukesAge--;
}
// Return a String representation of Duke's age
return new String("" + dukesAge);
```

Construire et déployer le service Web

- Choisir Run dans le menu contextuel du projet
- Si le fichier war est déployé correctement, un navigateur web s'ouvre et affiche l'âge de Duke

Chapitre 4 : Projet Firstcup

Cette application comprend l'entité, l'EJB et le front-end JavaServer Faces.

Créer le projet

- (M) File > New Project
- Categories : Java Web, Projects : Web Application, (B) Next
- Project name : firstcup
- Choisir l'emplacement que vous voulez.
- Vérifier que GlassFish v3 est bien sélectionné ainsi que Java EE 6
- Context Path : /firstcup
- Frameworks : Java Server Faces
- (O) Configuration
- Servlet URL Pattern : :/firstcup/*
- (B) Finish

Créer l'entité FirstcupUser

- (M) File > New File
- Categories : Persistence, File type : Entity class (B) Next
- Class Name : FirstcupUser, package : firstcup.entity
- (B) Create Persistence Unit
- Data source : jdbc/__default (B) Create
- (B) Finish

Ne pas taper, choisir dans le menu

Ajouter des propriétés à l'entité créée

- Cliquer avec le bouton droit dans la partie éditeur, Insert code, Add property
- Name : birthday
- Type : java.util.Calendar (B) Ok
- Corriger l'erreur, sélectionner Add (ou Create) @Temporal Annotation

Ajouter des constructeurs à l'entité créée

- Menu contextuel Insert Code | Constructor (B) Generate
- Directement sous le constructeur sans argument, ajouter le constructeur suivant

```
public FirstcupUser(Date date) {
   Calendar cal = new GregorianCalendar();
   cal.setTime(date);
   this.setBirthday(cal);
}
```

Ajouter des requêtes nommées à l'entité créée

- Insérer le code suivant, juste avant la définition de la classe

```
@NamedQueries({
   @NamedQuery(name = "findOldestFirstcupUser",
   query = "SELECT MIN(u.birthday) FROM FirstcupUser u"),
   @NamedQuery(name = "findYoungestFirstcupUser",
   query = "SELECT MAX(u.birthday) FROM FirstcupUser u")
})
```

- Faire les bons imports pour les annotations

Créer l'EJB. Avec les beans sans état, les clients font des requêtes indépendantes, qui ne dépendent pas de celles d'avant. Ce bean est aussi local, i.e. qu'il n'est visible uniquement qu'au niveau de l'application déployée et avec vue sans interface : il n'y a pas d'interface métier séparée supplémentaire (locale ou distante) à implémenter.

- (M) File > New file
- Categories : Java EE, File types : Session Bean
- EJB Name : DukesBirthdayBean
- Package : firstcup.ejb
- Choisir stateless
- (B) Finish

Ajouter la logique métier au Bean (avec une instance de Logger et une méthode métier)

- Juste après la déclaration de la classe, ajouter le code suivant

```
private static Logger logger =
        Logger.getLogger("firstcup.ejb.DukesBirthdayBean");
```

- Ajouter également l'attribut annoté

```
@PersistenceContext
private EntityManager em;
```

- Ajouter une méthode métier avec un clic sur le bouton droit : getAgeDifference() qui retourne un int
- Ajouter un paramètre date de type java.util.Date
- Insérer le code suivant

```
int ageDifference;
Calendar theirBirthday = new GregorianCalendar();
Calendar dukesBirthday = new GregorianCalendar(1995, Calendar.MAY, 23);
// Set the Calendar object to the passed in Date
theirBirthday.setTime(date);
// Subtract the user's age from Duke's age
ageDifference = dukesBirthday.get(Calendar.YEAR) -
                theirBirthday.get(Calendar.YEAR);
logger.info("Raw ageDifference is: " + ageDifference);
// Check to see if Duke's birthday occurs before the user's. If so,
// subtract one from the age difference
if (dukesBirthday.before(theirBirthday) && (ageDifference > 0)) {
   ageDifference--;
}
// create and store the user's birthday in the database
FirstcupUser user = new FirstcupUser(date);
em.persist(user);
logger.info("Final ageDifference is: " + ageDifference);
return ageDifference;
```

Tout comme la méthode DukesAgeResource.getText(), la méthode décrite ci-dessus calcule la différence d'âge entre l'utilisateur et Duke. Une entité FirstcupUser est créée qui est ensuite stockée dans la base de données grâce l'EntityManager.

- Formater le code et résoudre les imports.
- Sauvegarder

Ajouter des méthodes métier pour connaître l'utilisateur le plus jeune et le plus vieux, soit directement, soit par add Business Method (clic droit)

- Pour l'utilisateur le plus vieux
- Pour l'utilisateur le plus jeune
- Sauvegarder

```
public Calendar getOldestUserBirthday() {
   Date date = (Date) em.createNamedQuery("findOldestFirstcupUser")
               .getSingleResult();
   Calendar cal = new GregorianCalendar();
   cal.setTime(date);
   return cal;
}
```

```
public Calendar getYoungestUserBirthday() {
   Date date = (Date) em.createNamedQuery("finYoungestFirstcupUser")
               .getSingleResult();
   Calendar cal = new GregorianCalendar();
   cal.setTime(date);
   return cal;
}
```

Ces méthodes appellent des requêtes nommées en utilisant le gestionnaire d'entités. La requête retourne un objet de type Calendar retravaillé pour renvoyer un objet de classe Date.

Créer le client Web en réalisant les différentes tâches

- Créer des ressources pour la localisation
- Configurer ces ressources
- Créer un bean géré DukesBDay
- Créer les pages Facelets

Créer un lot de ressources

- Sélectionner le projet
- Choisir (B) New > Other puis Properties File
- File Name : WebMessages, Folder : src/java/firstcup/web
- (B) Finish
- Editer le fichier avec ce contenu, puis sauvegarder

```
Welcome=Hi. My name is Duke. Let us find out who is older -- You or me.
DukeIs=Duke is
YearsOldToday=years old today.
Instructions=Enter your birthday and click submit.
YourBD=Your birthday
Pattern=MM/dd/yyyy
DateError=Please enter the date in the form MM/dd/yyyy.
YouAre=You are
Year=year
Years=years
Older=older than Duke!
Younger=younger than Duke!
SameAge= the same age as Duke!
Submit=Submit
Back=Back
YoungestUser=The youngest user was born
OldestUser=The oldest user was born
```

- Sauvegarder sous un autre nom WebMessages_fr.properties. Editer le nouveau fichier (voir page suivante), l'interface comparative est sympa ;-) Stocker les messages dans de tels fichiers permet d'éviter d'avoir des pages web différenciées sur les traductions.

Configurer l'application

- Sélectionner le projet (B) File > New > Other
- Categories : JavaServer Faces, File Types : JavaServe Faces Configuration
- (B) Finish
- Ajouter une balise <application>. Le choix de la langue va se faire sur la locale du navigateur web de l'utilisateur. <base-name> définit le nom complet du lot (bundle) de ressources. <var> spécifie le petit nom utilisé dans les pages xhtml.
- Formater et sauvegarder

```
Welcome=Bonjour, mon nom est Duke. Lequel de nous deux est le plus vieux ?
DukeIs=Duke est âgé de
YearsOldToday=années aujourd'hui
Instructions=Entre ta date de naissance et clique sur Soumettre
YourBD=Ta date de naissance
Pattern=MM/dd/yyyy
DateError=Entre , s'il te plait, la date sous la forme MM/dd/yyyy.
YouAre=Tu as
Year=an
Years=années
Older=de plus que Duke
Younger=de moins que Duke
SameAge= le même age que Duke!
Submit=Soumettre
Back=Retour
YoungestUser=L'utilsateur le plus jeune est né
OldestUser=L'utilisateur le plus vieux est né
```

```
<application>
  <resource-bundle>
    <base-name>firstcup.web.WebMessages</base-name>
    <var>bundle</var>
  </resource-bundle>
  <locale-config>
    <default-locale>en</default-locale>
    <supported-locale>fr</supported-locale>
  </locale-config>
</application>
```

Créer le bean managé (géré) DukesBD. Ce bean est ce que l'on appelle un baking bean, c'est un composant qui peut stocker temporairement des informations instancié par le serveur JSF pour une portée (scope) à préciser

Crée le bean géré

- Sélectionner avec le bouton droit le package firstcup.web
- New > Other Categories : JSF, File Types : Managed bean
- Class Name : DukesBDay
- Package : firstcup.web
- Scope : session
- (B) Finish

Ajouter une référence sur un EJB

- Clic droit sur l'éditeur, Insert Code, Call Enterprise Bean
- Sélectionner dans l'arbre DukesBirthdayBean
- (B) OK. L'annotation qui apparait dans le code permet de faire de l'injection sur cet EJB

Ajouter des propriétés au bean : par Insert Code ou à la main (ne pas oublier les getters et les setters)

- De type int : age, ageDiff, absAgeDiff
- De type java.util.Date : yourBD
- de type java.util.Calendar : olderUserBirthday, youngestUserBirthday

- de type java.util.TimeZone : timeZone
- Initialiser toutes valeurs de ces propriétés dans le constructeur sans argument (-1 pour les entiers, null pour les references)
- Créer une instance de java.util.logging.Logger.Ceci permet de garder des informations dans les logs du serveur.

```
private static Logger logger = Logger.getLogger("firstcup.web.DukesBDay");
```

Obtenir l'âge actuel de Duke. Cela va se faire par une connexion HTTP au service web.

- Ajouter ce code à la méthode getAge()

```
public int getAge() {
  // Use the java.net.* APIs to access the Duke's Age RESTful web service
  HttpURLConnection connection = null;
  BufferedReader rd = null;
  StringBuilder sb = null;
  String line = null;
  URL serverAddress = null;
  try {
    serverAddress = new URL(
      "http://localhost:8080/DukesAgeService/resources/dukesAge");
    connection = (HttpURLConnection) serverAddress.openConnection();
    connection.setRequestMethod("GET");
    connection.setDoOutput(true);
    connection.setReadTimeout(10000);
    // Make the connection to Duke's Age
    connection.connect();
    // Read in the response
    rd = new BufferedReader(
         new InputStreamReader(connection.getInputStream()));
    sb = new StringBuilder();
    while ((line = rd.readLine()) != null) {
      sb.append(line);
    }
    // Convert the response to an int
    age = Integer.parseInt(sb.toString());
  } catch (MalformedURLException e) {
    logger.warning("A MalformedURLException occurred.");
    e.printStackTrace();
  } catch (ProtocolException e) {
    logger.warning("A ProtocolException occurred.");
    e.printStackTrace();
  } catch (IOException e) {
    logger.warning("An IOException occurred");
    e.printStackTrace();
  }
  return age;
}
```

- Formater le code, résoudre les imports et sauvegarder

Obtenir la date de naissance de l'utilisateur le plus jeune et le plus vieux.

```
public Calendar getOldestUserBirthday() {
  this.oldestUserBirthday = dukesBirthdayBean.getOldestUserBirthday();
  logger.info("Oldest user was born: " + this.oldestUserBirthday);
  return oldestUserBirthday;
}
```

```
public Calendar getYoungestUserBirthday() {
  this.youngestUserBirthday = dukesBirthdayBean.getYoungestUserBirthday();
  logger.info("Youngest user was born: " + this.youngestUserBirthday);
  return youngestUserBirthday;
}
```

Obtenir la différence d'âge grâce à l'EJB DukesBirthdayBean. Le code « success » retourné par cette méthode est utilisé par le serveur JSF pour afficher la page

```
public String processBirthday() {
  this.setAgeDiff(dukesBirthdayBean.getAgeDifference(yourBD));
  logger.info("age diff from dukesbday " + ageDiff);
  this.setAbsAgeDiff(Math.abs(this.getAgeDiff()));
  logger.info("absAgeDiff " + absAgeDiff);
  return new String("success");
}
```

Nous allons maintenant créer le client Facelets constitué d'une bibliothèque de ressources, d'un composant composite et de deux fichiers xhtml.

La bibliothèque de ressources est une collection de composants créés par l'utilisateur que l'on peut trouver à un emplacement standard dans une application web. Les ressources sont identifiées grâce à un identifiant de ressource, une chaine de caractères particulière dans une application web de la forme : [localePrefix/][libraryName/][libraryVersion/]resourcename[/resourceversion]

Si les ressources sont packagées à la racine de l'application web, elles doivent se trouver dans le répertoire *resources* de l'application. Si elles sont packagées dans le classpath, elles doivent se trouver dans le répertoire META-INF/resources.

Un composant composite est un ensemble de composants JSF et Facelets définis par l'utilisateur placé dans une ressource. Pour ces composants, il y a une définition et une implémentation. Le composant peut contenir des convertisseurs, par exemple pour transformer une chaîne de caractère en une date. Le composant à créer va lire dans un formulaire la date de naissance de l'utilisateur.

Créer le composant composite inputDate. Le wizard : JSF Composite Component ne marche pas alors on va le faire à la main

- WIZARD : Sélectionner le projet et (M) File > New File
- Categories : JavaServer Faces File Types : JSF Composite Component (B) Next
- FileName : inputDate, Folder : resources/components, le prefix est rempli automatiquement
- (B) Finish
- Sélectionner Web Pages du projet
- (BD) créer un répertoire *resources* puis un sous-répertoire *components*

- Créer une page JSF nommée inputDate.xhtml dans le répertoire créé. Prendre comme option : Facelets
- (B) Finish
- Ajouter dans la balise html les espaces de nommage suivants :

```
xmlns:composite=http://java.sun.com/jsf/composite
xmlns:f="http://java.sun.com/jsf/core"
```

- Ajouter entre les balises <body> et </body> le code suivant

```
<composite:interface>
  <composite:attribute name="date" required="true" />
</composite:interface>
```

- Place ce code après la définition

```
<composite:implementation>
  <h:inputText value="#{cc.attrs.date}">
  <f:convertDateTime pattern="MM/dd/yyyy" />
  </h:inputText>
</composite:implementation>
```

- Les fichiers xhtml peuvent aussi être formattés et sauvegardés ;-)

Créer les fichiers greetings.xhtml et response.xhtml en créant des pages JSF comme ce que vous venez de faire. Le code est donné sur les pages suivantes.

Le fichier greetings.xhtml utilise les bibliothèques de balise JSF composite/components (la notre créé avec le composant composite) et HTML. Les éléments du backing bean DubkeBD sont lus directement grâce à EL : #{DukesBD.yourBD}.

Les chaînes localisées(dont la traduction dépend de la locale) utilisent EL et préfixe *bundle* : #{bundle.Welcome}.

La balise <h :commandButton> permet d'afficher un bouton « soumettre ». S'il y a un problème la balise <h :message> associée affiche par exemple : « j_idt4:userBirthday:j_idt13: 'qsd' could not be understood as a date ».

Le fichier response.xhtml affiche des renseignements dans un formulaire. Les éléments du formulaire <ouputText> ne sont affichés que si une condition est vérifiée (Attention, EL utilise une syntaxe différente du java < est transformé en lt par exemple). La condition est dans l'attribut rendered.

```
<?xml version="1.0" encoding="UTF-8"?>
<!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" xml:lang="en" lang="en"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:fc="http://java.sun.com/jsf/composite/components">
  <head>
  <title>Firstcup Greeting Page</title>
  </head>
  <body>
    <h:form>
      <h2>
        <h:outputText value="#{bundle.Welcome}"/>
      </h2>
      <h:outputText value="#{bundle.DukeIs} "/>
      <h:outputText value="#{DukesBDay.age} #{bundle.YearsOldToday}"/>
      <p/>
      <h:outputText value="#{bundle.Instructions}"/>
      <p/>
      <h:outputText value="#{bundle.YourBD} "/>
      <fc:inputDate id="userBirthday" date="#{DukesBDay.yourBD}" />
      <h:outputText value=" #{bundle.Pattern}"/>
      <p/>
      <h:commandButton value="#{bundle.Submit}"
         action="#{DukesBDay.processBirthday}"/>
      <p/>
      <h:message for="userBirthday" style="color:red"/>
    </h:form>
  </body>
</html>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<!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" xml:lang="en" lang="en"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html">
  <head>
    <title>Response Page</title>
  </head>
  <body>
    <h:form>
      <h:outputText value="#{bundle.YouAre} "/>
      <h:outputText value="#{bundle.SameAge}"
         rendered="#{DukesBDay.ageDiff == 0}"/>
      <h:outputText value="#{DukesBDay.absAgeDiff}"
        rendered="#{DukesBDay.ageDiff lt 0}"/>
      <h:outputText value=" #{bundle.Year} "
        rendered="#{DukesBDay.ageDiff == -1}"/>
      <h:outputText value=" #{bundle.Years} "
        rendered="#{DukesBDay.ageDiff lt -1}"/>
      <h:outputText value="#{bundle.Younger}"
        rendered="#{DukesBDay.ageDiff lt 0}"/>
      <h:outputText value="#{DukesBDay.absAgeDiff}"
        rendered="#{DukesBDay.ageDiff gt 0}"/>
      <h:outputText value=" #{bundle.Year} "
        rendered="#{DukesBDay.ageDiff == 1}"/>
      <h:outputText value=" #{bundle.Years} "
        rendered="#{DukesBDay.ageDiff gt 1}"/>
      <h:outputText value="#{bundle.Older}"
        rendered="#{DukesBDay.ageDiff gt 0}"/>
      <p/>
      <h:outputText value="#{bundle.YoungestUser}: " />
      <h:outputText value="#{DukesBDay.youngestUserBirthday.getTime()}">
      <f:convertDateTime pattern="MM/dd/yyyy" />
      </h:outputText>
      <p/>
      <h:outputText value="#{bundle.OldestUser}: " />
      <h:outputText value="#{DukesBDay.oldestUserBirthday.getTime()}">
        <f:convertDateTime pattern="MM/dd/yyyy" />
      </h:outputText>
      <p/>
      <h:commandButton id="back" value="#{bundle.Back}" action="greeting"/>
    </h:form>
  </body>
</html>
```

Modifier le fichier web.xml pour que greetings soit la page d'accueil (welcome) de l'application web.

Préciser la navigation entre les pages : en particulier, lorsque processBirthday() est un succès, afficher la page response.xhtml

- Double cliquer sur le fichier faces-config.xml dans les fichiers de configuration (ou alors dans WEB-INF des pages web)
- Cliquer sur PageFlow en haut à gauche pour avoir l'éditeur visuel
- Sélectionner greeting.xml (vers la droite de la boite) et tracer une flèche vers response.xhtml
- Cliquer sur le nom de la relationet changer case1 par success

- Sauvegarder

Il ne reste plus qu'à déployer et exécuter pour voir ce qui se passe.

Dernières remarques :

Si le programme provoque une erreur au déploiement, je vous propose de vérifier que le fichier persistence.xml est correct et qu'il contient la bonne source jdbc : : jdbc/__default.

Si la page greetings.xhtml ne s'affiche pas correctement, il faut vérifier que le serveur JSF doit traiter les pages (le répertoire faces n'a plus besoin d'être présent dans le pattern de reconnaissance des noms de fichiers mais le laisser ne mange pas de pain)

Il a une exception sur la page response.xhtml …regarder le message d'erreur et corriger l'erreur. Celle-ci vient de l'EJB, il manque une lettre quelque part.

Le bouton retour ne marche pas car il manque un lien de retour. Le message d'erreur suivant est affiché : « Unable to find matching navigation case with from-view-id '/response.xhtml' for action 'greeting' with outcome 'greeting'". Il est très facile à corriger

On peut essayer de tester les locales avec les navigateurs de langues différentes .j'ai essayé sans succès cette extension pour firefox : https://addons.mozilla.org/en-US/firefox/addon/1333/. La langue apparait dans la barre des taches.

Il est relativement facile de passer par une base mySQL et non plus Derby pour stocker les dates de naissances. IL faut créer une ressource JNDI et un pool de connexions. Les détails sont dans un document annexe que vous trouverez également sur mon site Web.