- TD 2 -
Manage persons with EJBs

Now we want to develop an application that would not only greet people but also store them in a database. In this application we use a relatively naive method (using JDBC) to persist, a more tailored approach will be implemented in later.

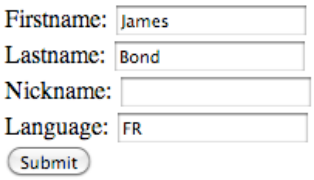
1. Ressource management

The application offers features that may require access to different resources, for example, a database to record information about the persons. To facilitate the implementation, we will use the Derby server integrated with NetBeans but you can test the application with another database installed on your machine. The indications for the creation of a database for creating tables and visualization (and filling) are available at http://netbeans.org/kb/docs/ide/java-db.html. We must create a DB called GLA (gla/gla) and a table `PERSONS` containing the columns `nickname`, `firstname`, `lastname` (a script to create the corresponding table is available on the course page).

We should also create the corresponding resource in the GlassFish Server using the Administration Console: you must first create a JDBC Resource Pool and then JDBC Resource. We will call the pool (type `DataSource` and driver Derby-30) GLAPool - once created the pool must be tested with a Ping (in case of failure you should probably set property SecurityMechanism to 3 or remove it). You must then create a resource jdbc/gla in the pool GLAPool. We can now use this resource to ensure the persistence of our application. The resources can be also created using NetBeans (category Glassfish). We can now use this resource to ensure the persistence for our application.

2. Persons with nicknames

We want to develop the web pages to save persons into a BDD and a list the persons already registered. The business logic of the application is implemented using EJBs.



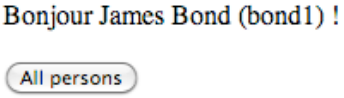
Individuals are characterized by a firstname, a lastname and an identifier (the identifier is unique). You can upgrade your existing application or create a new one called, for example, GestionPersonnes (you could structure the application using a library interfaces for EJBs and possibly for other classes/interfaces). The first page (`index.xhtml`) should allow the registration of new persons and this page may have the form:

For the moment, validating the page should result in displaying (in another page) a welcome message for the person. If the user does not fill the field Nickname then an identifier is generated automatically from the name (of the person) and a random integer. For example, the result for the data below should display « `Bonjour James Bond (bond20) !` » in a page.

To implement this behavior we can develop an EJB NameHandlerBean with a method to generate identifiers from a name. This bean must also provide the method greetingsMessage (as in the previous TD) to greet the person.

3. List persons

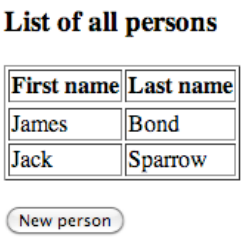
We develop now an EJB PersonManagerBean which allows you to add persons in the database previously created (you can optionally use the Person class provided on the course home page). For now, we assume that the generation of identifiers is sufficiently random and we don't check the uniqueness of the identifier; eventually we will generate unique



identifiers and/or manage duplicate keys. Following a successful submission we must be directed to a page of the form:

When using the button « `All persons` » we can get a list of all the persons already registered:

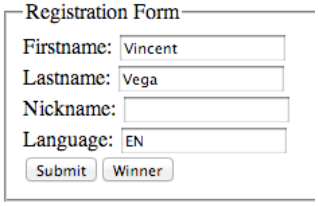
Start by identifying the interface of the EJB bean PersonManagerBean and implement it. The beans need to access different resources - use callbacks to optimize their use.

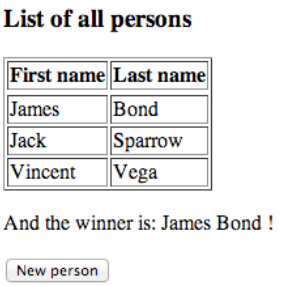


4. Random winner

Every day one of the persons saved in the DB is selected to receive an award - the choice is either random or through the web interface. To select the person explicitly you should add a button to the previous interface:

Complete the application to enable the display of this person effectively. For example, one can imagine that the winner is displayed when you list all persons:





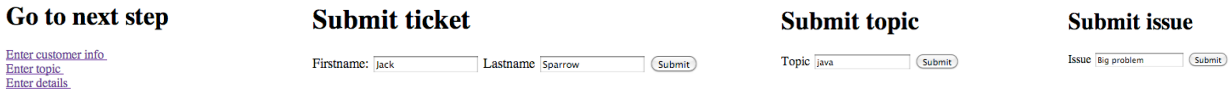
To be sure that a person is selected when the application starts, you can use a bean that allows to automatically record a certain number of persons.

5. Help desk

Users can submit tickets for different problems. The submission of tickets is done in several steps: the user provides personal information (name and surname) and then fills the data (the topic and the description of the problem). The submission performed in several stages allows the implementation of a workflow corresponding to the application business (for example, the possible topics are proposed depending on the identity of the user). Submission of tickets must be done with a web interface and later on with a traditional client.

We must therefore develop a Stateful EJB TicketManagerBean to achieve this behavior. The Bean will be called from an application client and from web pages (JSP and JSF). Be careful, calls from a client without state (Managed Bean or Servlet) must be done carefully.

The web interface, you could have a sequence of screens of the form:



If the workflow is not respected then, an exception `WorkflowViolationException` is thrown and it must be handled in the different clients by indicating the error with error messages/error. Classes `Ticket` and `WorkflowViolationException` are provided. Structure the application using packages.

As far as it concerns the application client , the interface may have the form:

```
First name : James
Last name : Bond
Topic:  JEE
Issue: Big problem with my EJBs
```

and, for the moment, this can result in printing in the console a message of the form (eventually, one can imagine that tickets are stored in a DB):

```
Ticket{person=Person{firstName=James,lastName=Bond},topic=JEE,issue=Big problem with my EJBs}
```

One can imagine that the registration of each ticket is relatively long. Propose eventually a solution where the duration of the ticket recording is transparent to the client.

Using a STSB is justified in this because of the diversity of potential clients. This type of session bean is also useful if the parameters of the invoked methods have significant sizes and therefore you want to minimize their use - you could, for example, imagine a method that takes as a parameter the identity (which contains among other things a high-definition picture) and provides the corresponding topics. You can also use a STSB if you need an `Extended Persistence Context`.