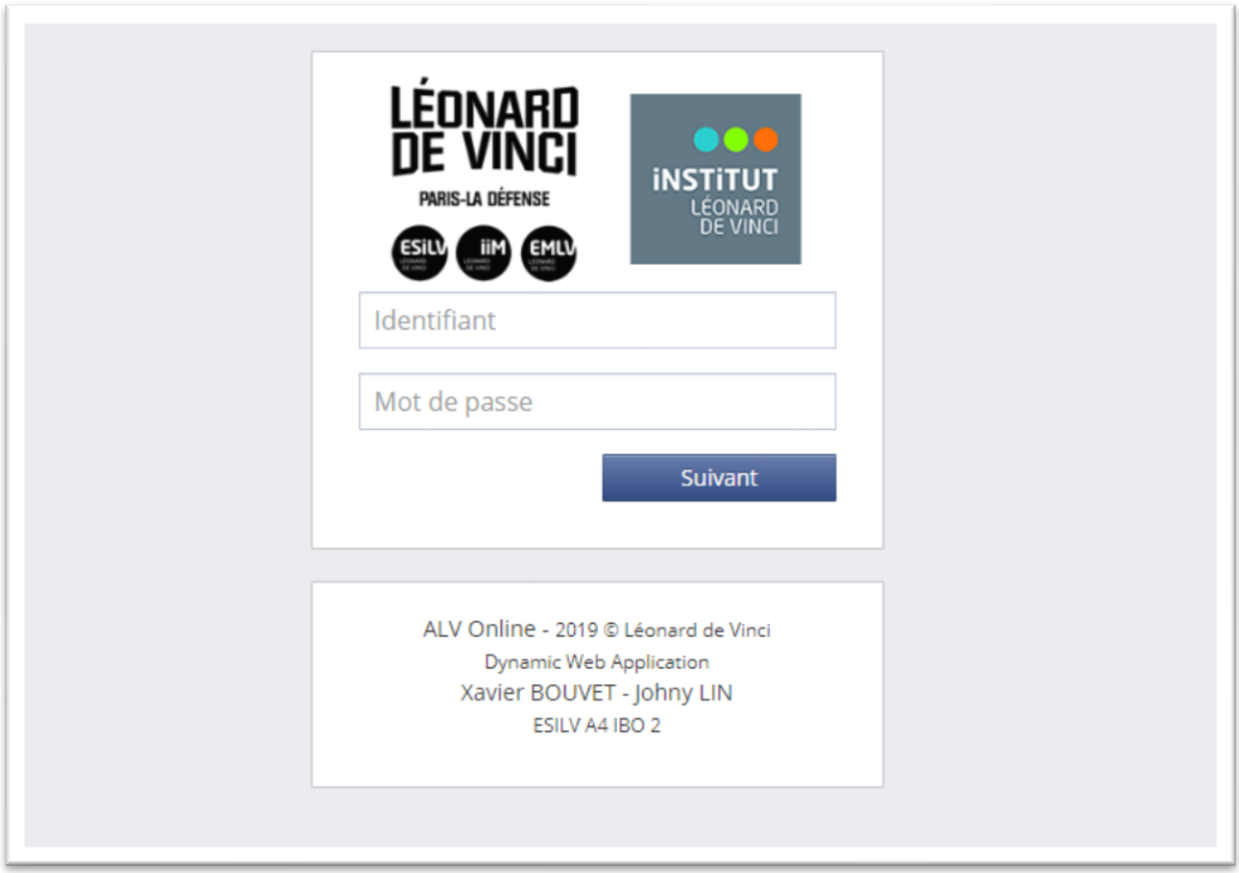
Project 1
Dynamic Web Application
Web ToDo List



Xavier BOUVET – Johny LIN
IBO Core 2 – Promo 2021 A4
Projet Java EE & frameworks

ESILV LÉONARD DE VINCI | ÉCOLE D'INGÉNIEURS PARIS-LA DÉFENSE

Project description:

The objective of the project is to create a dynamic web application with a login which separate users between students and instructor. According to the status, the application gives different options to the access to the todo list:

- A student can only read the todo list.
- An instructor can read, add, edit and delete todo on the list.
- An unknow tries to connect to the application, an error message is displayed.

The instructor and all his students have user accounts in the SQL database. The login system verifies his account if it is valid user or not.
Also, the web site creates a cookie for the username after the first login by a new user.
The login graphic interface is free, we can use a free template to display our login page.

The project was done in pairs by Johny LIN and Xavier Bouvet.
We separate the project in two parts:

- The login program to the database with instructors and student and the graphic interface.
- The todo list program with also his database with the add, edit and delete functionality.

The Data Base:

We initialized the database with MySQL Workbench.
We create the database and initialize 2 tables, the first will receive all the username and password for the login system and the second will receive all the todo list.

```
# Data Base Xavier Bouvet Johny LIN A4 ESILV JavaEE
drop database Identification;
CREATE DATABASE IF NOT EXISTS Identification ;
use Identification;

#Etape 1 Table Login
drop table register;

CREATE TABLE register (
  identifiant  VARCHAR(30) NOT NULL ,
  password VARCHAR(30),
  status int(1),
  CONSTRAINT pk_identifiant PRIMARY KEY (identifiant)
);

INSERT INTO register VALUES('Thai','1',1);
INSERT INTO register VALUES('Gossard','2',1);
INSERT INTO register VALUES('Nada','2',1);
INSERT INTO register VALUES('Johny','3',0);
INSERT INTO register VALUES('Xavier','3',0);
```

```
#Etape 2 Table todo
drop table todo;
create table todo (
id int(10) primary key auto_increment,
descrip varchar (80)
);
#auto incrementation de l'id

insert into todo (descrip) value ('Faire le projet de JavaEE');
insert into todo (descrip) value ('Faire une convcall avec son binome');
insert into todo (descrip) value ('Preparer le prochain cours');
insert into todo (descrip) value ('Preparer le cours de Machine Learning');
insert into todo (descrip) value ('Profiter de vos vacances');
```

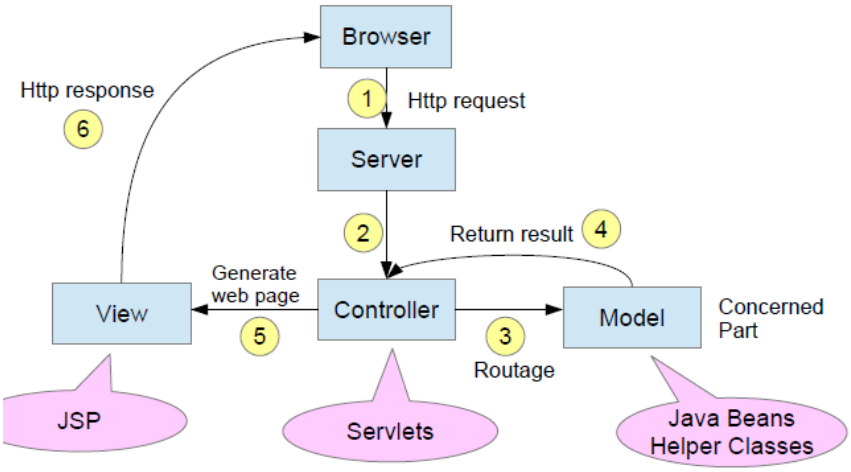
To make the difference between an instructor and a student, we define in the table a field status, which takes the value 0 if it's a student and 1 if it's an instructor. In the two tables, id is auto incremental.

The sql file is given inside the project called bddjava.sql.
We give the instructor and student username and password.

```
INSERT INTO register VALUES('Thai','1',1);
INSERT INTO register VALUES('Gossard','2',1);
INSERT INTO register VALUES('Nada','2',1);
INSERT INTO register VALUES('Johny','3',0);
INSERT INTO register VALUES('Xavier','3',0);
```

The login page:

We use the MVC Design Pattern to set up our dynamic application with Java EE and his components Servlets and JSPs.



Servlets:

- AddToDo.java
- DeleteToDo.java
- EditToDo.java
- RegisterControllerServlet.java
- registerDButil.java
- Todo.java

RegisterControllerServlet is the main servlet with **AddToDo**, **EditToDo** and **DeleteTodo**.
We have **registrerDButil** which works with the database as a Helper Classes. Its goal is to send some request and get the response with the database.
Todo is the java class which keep the todo list inside.

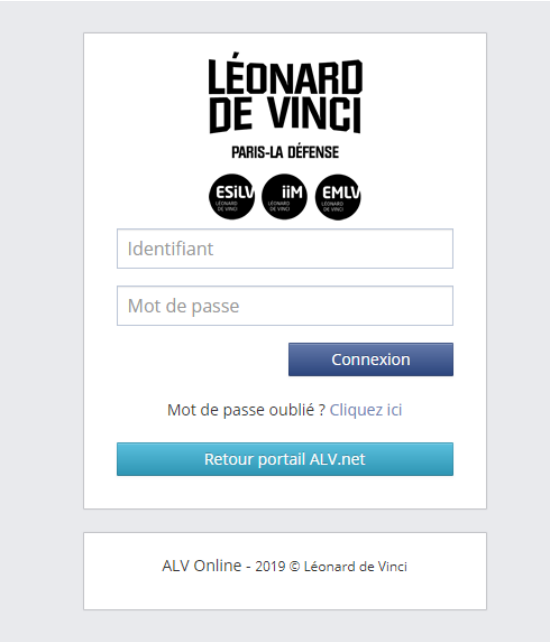
Java Server Pages (JSPs):

We chose the html of the template of the login page of leonard-de-vinci-net as our graphical user interface.

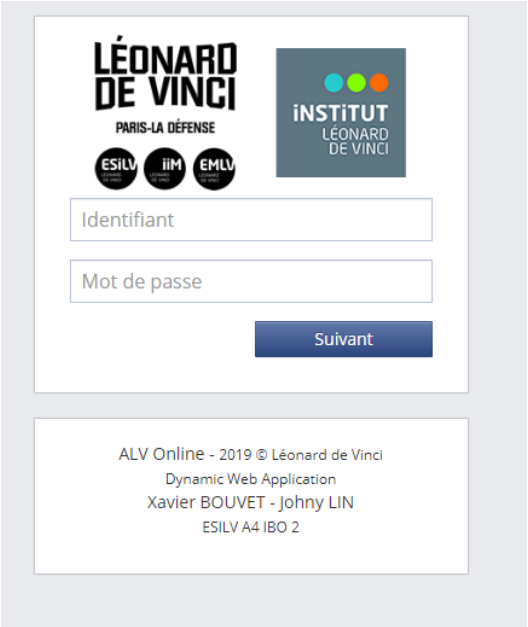
Source : https://www.leonard-de-vinci.net/parent/

We integrate the htmt code in our Java code to improve the graphic rendering.

The original page:

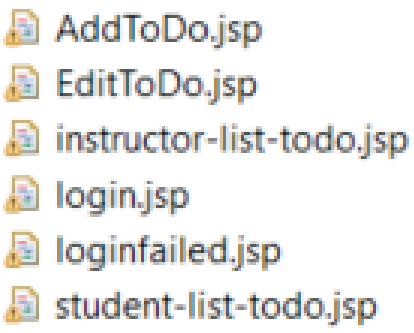


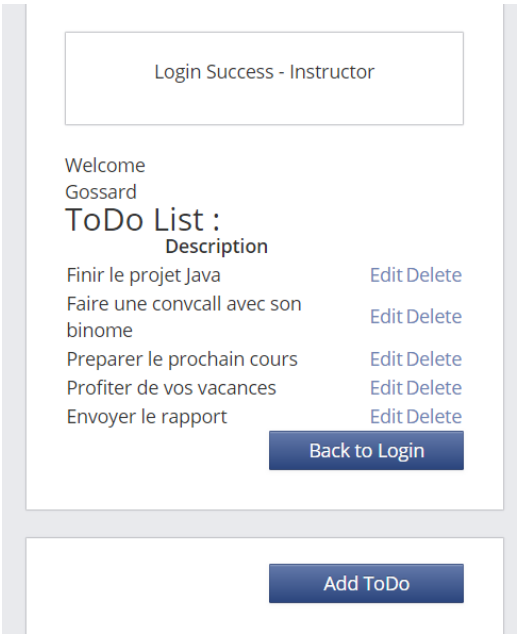
Our page:



Each Servlets directs to a .jsp file, it is our View in the MVC model. The **RegisterControllerServlet** call the **login.jsp** at first the login page.

- **loginfailed.jsp** show an error message and return to the **login.jsp**
- **AddToDo** and **EditToDo** are called by their respective servlets to do some action.
- **instructor-list-todo.jsp** show the todo list in the database. The instructor has 3 buttons to customize the list and a button to return to the login page.
- **student-list-todo.jsp** is the same as the instructor-list-todo.jps, but without 3 buttons to modify the database of todo.





Program:

• How does the login work?

It's the **RegisterControllerServlet** which get the username and the password (the method login) and verifies thank to the helper class **registerDButil**. After that, according the status, an instructor, a student or an unknown the program redirect to another .jsp file.

• Status == 1, it's an instructor, he can add,edit or delete a todo on the list.

• Status == 0, it's a student, he can only read the todo list

• Else, it's an unknown, an error message is displayed

```
if(status == 1) {
    //instructor
    request.getRequestDispatcher("/instructor-list-todo.jsp").forward(request, response);
}
else if(status == 0) {
    //student
    request.getRequestDispatcher("/student-list-todo.jsp").forward(request, response);
}
else {
    //not autorized
    request.getRequestDispatcher("/loginfailed.jsp").forward(request, response);
}
```

• How the todo list work?

The todo list is displayed by **instructor-todo-list.jsp** or **instructor-todo-list.jsp** which recover the list by request to the database which the java class Todo.java**.**

In the case, if the user is an instructor, the add, the editor and the delete function is available.

This function work with any method in the registerDButil.java. The program sends some request to the database to recover, add, edit or delete some todo list according to their id or description.

```
public int login (String id,String mdp) {
public ArrayList<Todo> getRegister() throws Exception {
public Todo fetchTodo(int id) {
public void updateTodo(Todo todo) {
public void deleteTodo(int id) {

public String recover() {

public void add (String descrip ){
```

• How the cookies work?

Each time someone enter his username, the controller saves it in the array of cookies.

```
try {
    Cookie [] cookies = request.getCookies();
    if(cookies!= null) {
        for(Cookie cookie:cookies) {
            if(cookie.getName().equals("prenom")) {
                request.setAttribute("prenom", cookie.getValue());
            }
        }
    }
```