DYNAMIC WEB APPLICATION

WEB TODO LIST

BECKER LUCAS & NGO THOMAS
IOS 1 – PROMO 2022

DESCRIPTION

The objective of the project is to dynamic web application called “WebToDoList”. That app will allow an instructor to put online a todo list for his students and allow the students to access this todo list.

An authentication interface will first prompt in order to identify, through a login process, if the user is a student or an instructor. Depends on the status of the user, different options will be accessible.

- 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. An account is defined by a username, a password and a role (instructor or student). After the authentication process, the web site creates a cookie for the username after the first login by a new user.

THE DATABASE

In the MySQL server, the database “webtodolistdb” was created and initialized by two tables:

- register: receive all the username and password for the login system

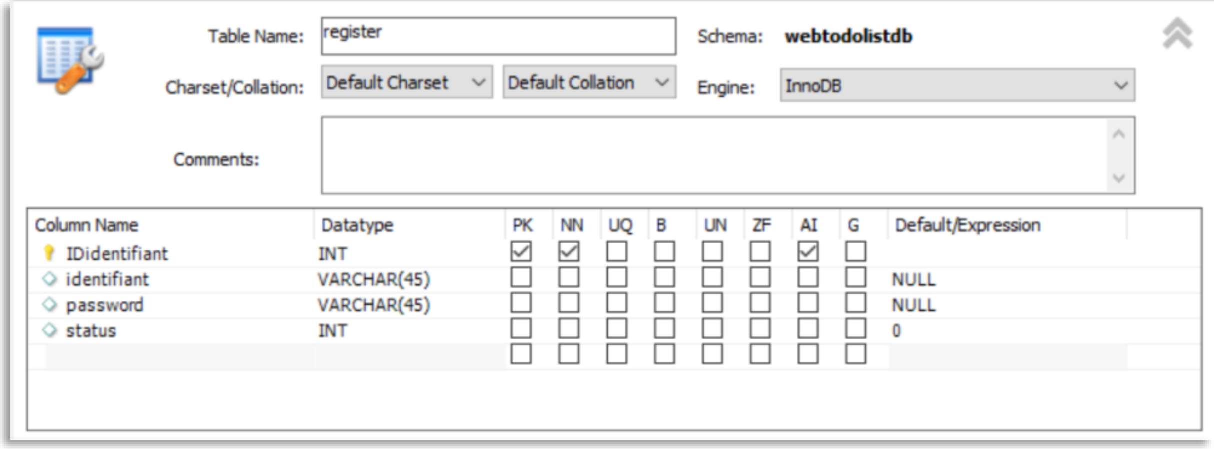


Table Name: register

Schema: webtodolistdb

Charset/Collation: Default Charset · Default Collation

Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
IDidentifiant	INT	☑	☑	□	□	□	□	☑	□	
identifiant	VARCHAR(45)	□	□	□	□	□	□	□	□	NULL
password	VARCHAR(45)	□	□	□	□	□	□	□	□	NULL
status	INT	□	□	□	□	□	□	□	□	0
		□	□	□	□	□	□	□	□	

As said before, an account is defined by a username, a password and a role (instructor or student). The third parameters in the register table define the status of the user. 1 for an instructor and 0 for a student.

- todo: receive all the todo list

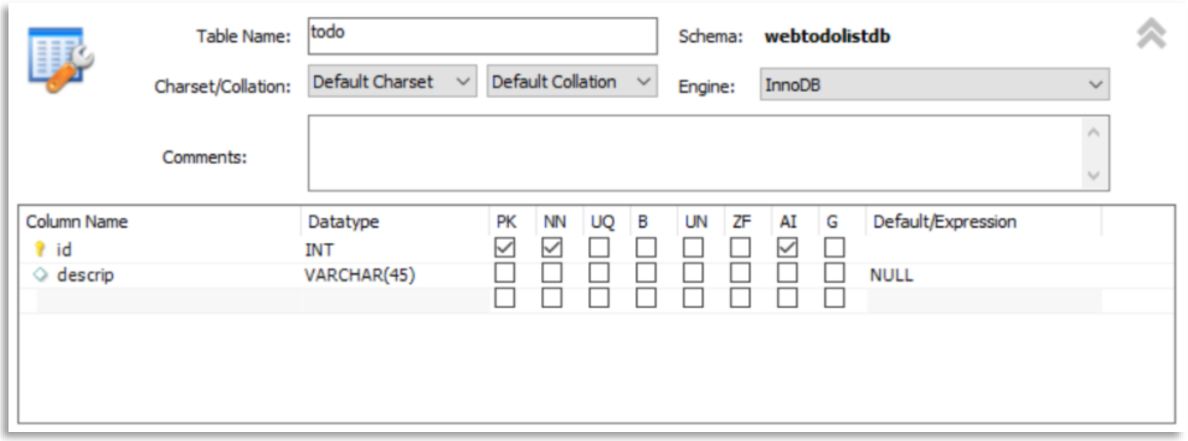


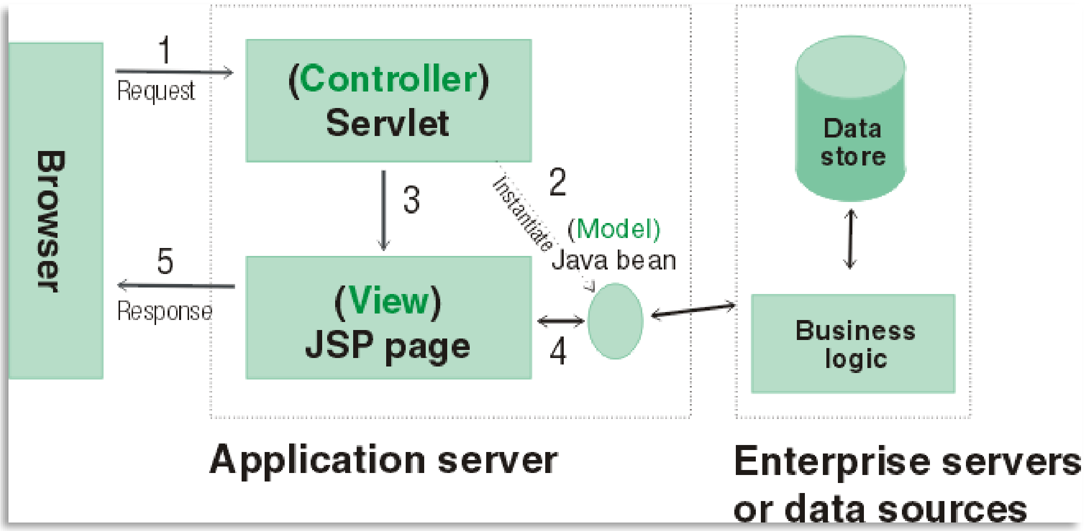
Table Name: todo
Schema: webtodolistdb
Charset/Collation: Default Charset | Default Collation
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	☑	☑	□	□	□	□	☑	□	
descrip	VARCHAR(45)	□	□	□	□	□	□	□	□	NULL
		□	□	□	□	□	□	□	□	

Todo table, meanwhile, is just composed of an auto-incremented id and a description.

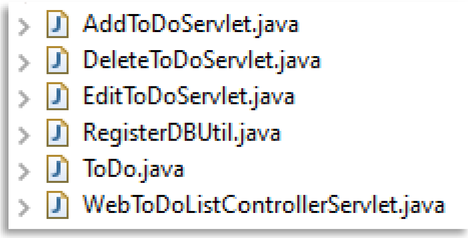
HOW IT WORKS?

Our program is based on MVC Design Pattern, it based on Servlets and JSPs components.



SERVLETS

We created several servlets as shown in the picture below.



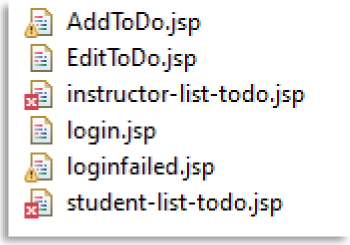
- AddToDoServlet.java
- DeleteToDoServlet.java
- EditToDoServlet.java
- RegisterDBUtil.java
- ToDo.java
- WebToDoListControllerServlet.java

In total we have 4 servlets. The main servlet is **WebToDoListControllerServlet.java** for the we have **AddToDoServlet.java**, **DeleteToDoServlet.java** and **EditToDoServlet.java**.

RegisterDBUtil.java and **ToDo.java** are two classes created respectively for working as a helper class with the database (send requests and get the response of the database) and to keep the todo list.

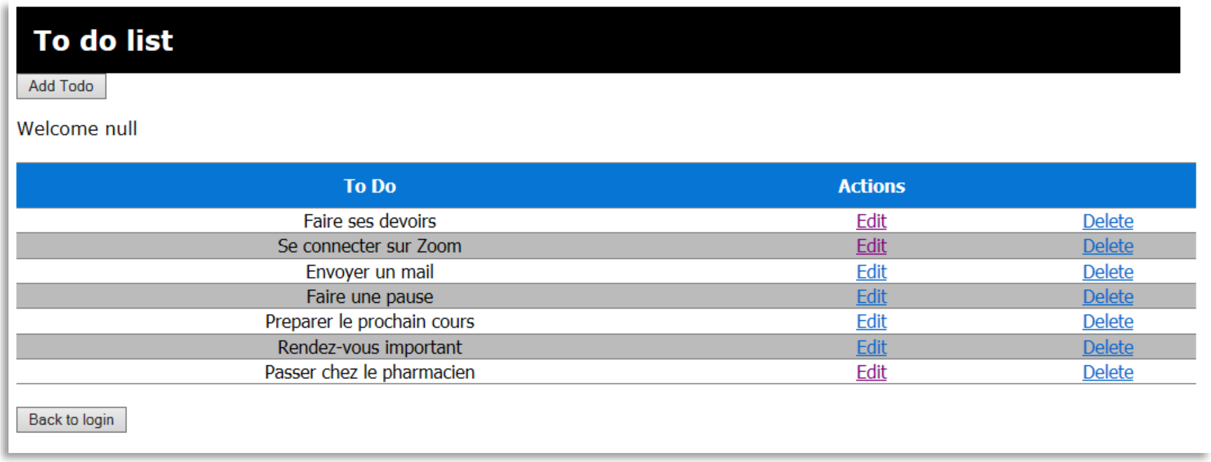
JSP

Servlets presented above directs to several .jsp files list below.



AddToDo.jsp
EditToDo.jsp
instructor-list-todo.jsp
login.jsp
loginfailed.jsp
student-list-todo.jsp

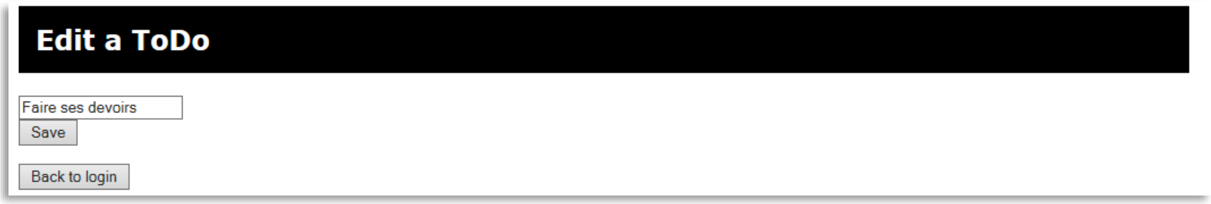
- **login.jsp** is called by the main servlet **WebToDoListControllerServlet.java** in order to let the user authenticate
- **loginfailed.jsp** shows an error of login
- **AddToDo.jsp** is called by **AddToDoServlet.java** to add a todo in the list
- **EditList.jsp** is called by **EditToDoServlet.java** to edit or delete a todo in the list
- **instructor-list-todo.jsp** shows the todo of the list in the database, buttons to customize the list and a button to return to the login page
- **student-list-todo.jsp** shows practically the same interface as **instructor-list-todo.jsp** but without buttons to customize the list



To do list

Welcome null

To Do	Actions	
Faire ses devoirs	Edit	Delete
Se connecter sur Zoom	Edit	Delete
Envoyer un mail	Edit	Delete
Faire une pause	Edit	Delete
Preparer le prochain cours	Edit	Delete
Rendez-vous important	Edit	Delete
Passer chez le pharmacien	Edit	Delete

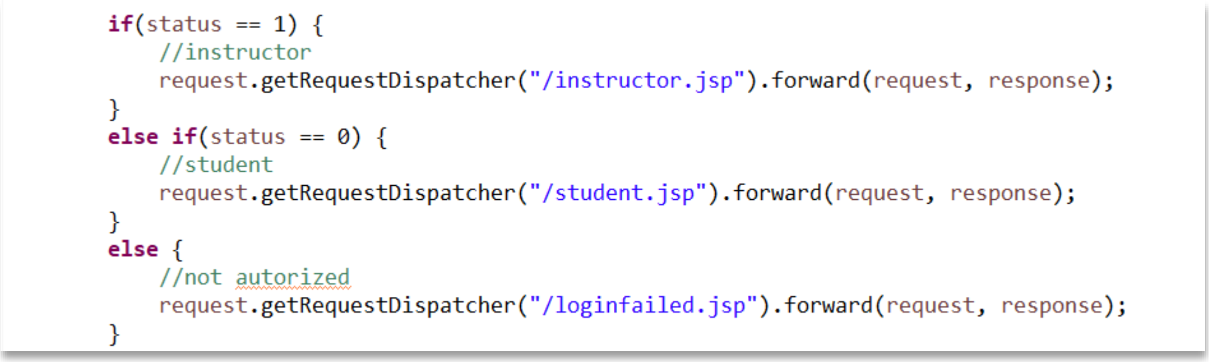


Edit a ToDo

Faire ses devoirs

LOGIN

Let’s go a bit deeper in the code to understand the login process.

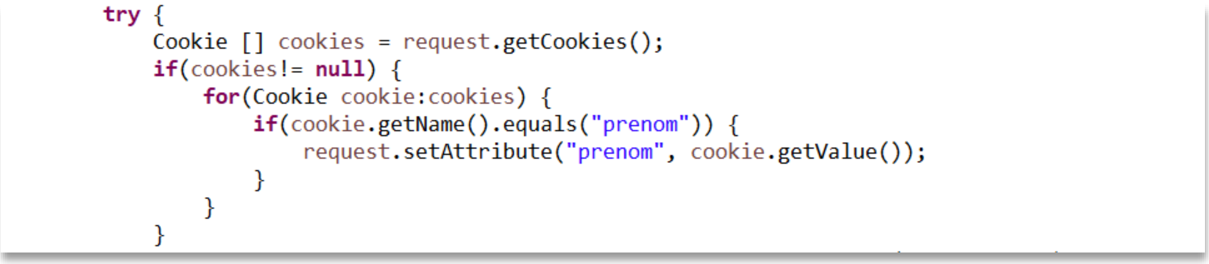


```
if(status == 1) {
    //instructor
    request.getRequestDispatcher("/instructor.jsp").forward(request, response);
}
else if(status == 0) {
    //student
    request.getRequestDispatcher("/student.jsp").forward(request, response);
}
else {
    //not autorized
    request.getRequestDispatcher("/loginfailed.jsp").forward(request,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

WebToDoListControllerServlet.java gets the username and the password and verifies in the database with the helper class **RegisterDBUtil.java**. Recall that the status of accounts of defined by 1 for instructors and 0 for students. After analyzing status, the program will redirect to another .jsp file.

COOKIES

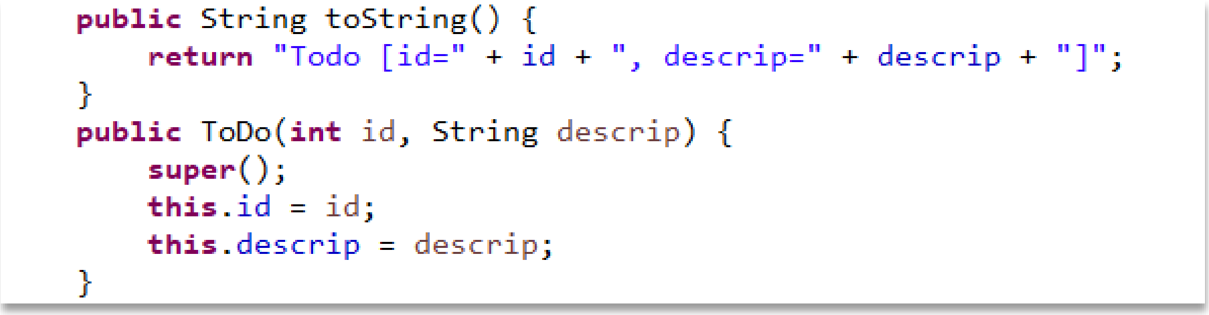
Each time the user inserts his name, **WebToDoListControllerServlet.java** will save it in the array of cookies.



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

TODO

instructor-list-todo.jsp and **student-list-todo.jsp** will display the todo list that is stored in the ToDo.java java class.



```
public String toString() {
    return "Todo [id=" + id + ", descrip=" + descrip + "]";
}
public ToDo(int id, String descrip) {
    super();
    this.id = id;
    this.descrip = descrip;
}
```