**Exercises: Servlets & JSPs**

Problems for exercises and homework for the [“Java Web Development Basics” course @ SoftUni](https://softuni.bg/courses/java-web-development-basics). Submit your solutions on the **course page** of the **current instance**.

**Fluffy Duffy Munchkin Cats (FDMC)**

Fluffy Duffy Munchkin Cats is an application in which you register cats, with several properties. It has many versions, and you will most probably see it several times.

You have already implemented several functionalities like **adding cats** for example, and **viewing data** about them. We implemented **JSPs**, and **exported** most of the **page content** there.   
We also implemented **user login** & **register**.

However, there is much more to do. You’ll have to do several other things.

* **User Roles**

Add a role property to the **User** class. There should be 2 roles in total – “**ADMIN**” and “**USER**”.

The **Admins**, should be able to **create cats**, while the **Users** should only be **able** to view them.

Make sure you don’t only hide the functionality from the front-end, but you also **restrict** the **functionality** only   
to **Users** with **ADMIN** role.

* **Implement Cat Views**

Add a **views** variable to the **Cat** class. Each time a **Cat’s profile** is **accessed**, its **views** should be **incremented**. The **first** view **counts** as **1 view**.

The **Views** of the **Cat** should be **rendered** on its **Profile Page**.

On the **All** **Cats** page, the cats should be **ordered** in **descending order**, by **views**.

* **Order Class**

Implement a class named **Order**, which should **hold 3 properties**:

* **client** – a **User** class property.
* **cat** – a **Cat** class property.
* **madeOn** – a **LocalDateTime** class property.
* **Order Functionality**

Implement an **[Order]** button on the **Cat** **Profile** page, which should create an **Order** with:

* **client** – the **currently logged in user**
* **cat** – the currently selected **Cat** **Profile**
* **madeOn** – the current **Date** and **Time**.

The **Order** should be **stored** in a **collection**, where it could easily be accessed.  
The **same way** you store **Users** and **Cats**.

The **Order functionality** should be visible to all **Users**.

* **All Orders**

Implement an **[All Orders]** button on the **Home page** (“**/**”), which should **ONLY be visible** to **Logged-in Admins**. It should list all **Orders**, ordered by **madeOn**, from the **most recent** to the **oldest**.

The Orders should represent:

* The **client’s username**
* The **cat’s name**
* The **madeOn** **time** and **date**
* **Data Layer \*\*\***

Remember the **BaseRepository** and the **Consumer** + **Execute around** patterns we used in the **Casebook** application. Try to implement them here, and try to change the **data layer** from plain List repositories to a dedicated relational database.

Export **all entities** there. (**User**, **Cat** and **Order**).