FWA

Java Servlet API

Summary: Summary: now you will develop your first web application using standard Java technologies

Version: 1

Contents

Chapter I

Preamble

- 200 - no worries, everything is fine

- 400 - you did not meet server's expectations

- 403 - you entered a wrong area

- 404 - a server did not meet your expectations

- 500 - Congratulations! You have broken the server.

- 504 - Have you seen Hachiko movie?
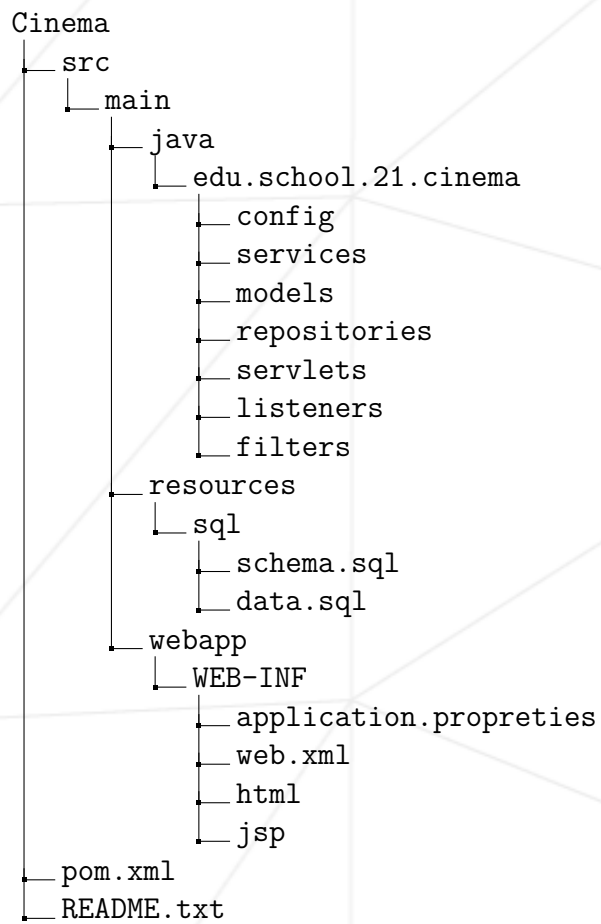
Chapter II

Instructions

- Use this page as the only reference. Do not listen to any rumors and speculations about how to prepare your solution.

- Now there is only one Java version for you, 1.8. Make sure that compiler and interpreter of this version are installed on your machine.

- You can use IDE to write and debug the source code.

- The code is read more often than written. Read carefully the document where code formatting rules are given. When performing each task, make sure you follow the generally accepted Oracle standards:

- Comments are not allowed in the source code of your solution. They make it difficult to read the code.

- Pay attention to the permissions of your files and directories.

- To be assessed, your solution must be in your GIT repository.

- Your solutions will be evaluated by your piscine mates.

- You should not leave in your directory any other file than those explicitly specified by the exercise instructions. It is recommended that you modify your .gitignore to avoid accidents.

- When you need to get precise output in your programs, it is forbidden to display a precalculated output instead of performing the exercise correctly.

- Have a question? Ask your neighbor on the right. Otherwise, try with your neighbor on the left.

- Your reference manual: mates / Internet / Google. And one more thing. There's an answer to any question you may have on Stackoverflow. Learn how to ask questions correctly.

- Read the examples carefully. They may require things that are not otherwise specified in the subject.

- Use "System.out" for output.

- And may the Force be with you!

- Never leave that till tomorrow which you can do today ;)

Chapter III

Rules of the project

- Implemented solutions should enable creating a WAR archive using maven package command. Such archive shall be deployed in Tomcat.

- In this project, the use of Spring MVC and Hibernate components is prohibited (repository layer shall be implemented using JdbcTemplate).

- For each task, you will need to create a README.txt file with instructions on how to deploy and use your application.

- For each task, you shall attach schema.sql and data.sql files where you describe a schema of a database being created and test data, respectively.

- You can add custom classes and files to each of the projects without breaking the overall suggested structure:

```
Cinema
├── src
│   └── main
│       ├── java
│       │   └── edu.school.21.cinema
│       │       ├── config
│       │       ├── services
│       │       ├── models
│       │       ├── repositories
│       │       ├── servlets
│       │       ├── listeners
│       │       └── filters
│       ├── resources
│       │   └── sql
│       │       ├── schema.sql
│       │       └── data.sql
│       └── webapp
│           └── WEB-INF
│               ├── application.propreties
│               ├── web.xml
│               ├── html
│               └── jsp
├── pom.xml
└── README.txt
```

Chapter IV

Exercice 00 : Welcome To Servlets

	Exercise 00	
	First Web Application	
Turn-in directory : *ex00/*		
Files to turn in : `Cinema-folder`		
Allowed functions : `n/a`		

You need to develop a web application prototype using Java Servlet API stack.
The application will automate the booking business process of a movie theater
later on.

Now you will develop an MVP application to partially implement registration
and authentication mechanisms.

Thus, your web application should provide HTML registration and authentication
pages in response to /signIn and /signUp URL requests, respectively.

When registering, a user specifies the following data:

- first name

- last name

- phone number

- password

All data should go to SignUp servlet in a POST request using <form> HTML tag.
The information is stored in a database, while the password shall be encrypted
using BCrypt algorithm.

When a POST request is sent to SignIn servlet with an email and a password,
a check is performed if a corresponding user exists in the database, as well
as their password is correct. If the check is successful, an HttpSession object

with user attribute shall be generated (attribute's value is an object containing
current user data). The user will be redirected to a blank profile page. In
case of a failed authentication, user should be redirected back to the login
page.

   Technical requirements:

Application's Spring context should be a separate configuration class (see
Spring Java Config) accessible to all servlets via ServletContextListener.
In this configuration, you shall specify .bin files to connect to the database
(DataSource) and encrypt passwords (PasswordEncoder), as well as for all services
and repositories. Data for connecting to the database shall be available in
application.properties.

Here is an example of the use of this configuration in a servlet:

```
@WebServlet("/users")
public class UsersServlet extends HttpServlet {

    private UsersService usersService;

    @Override
    public void init(ServletConfig config) throws ServletException {
        ServletContext context = config.getServletContext();
        ApplicationContext springContext = (ApplicationContext) context.getAttribute("springContext");
        this.usersService = springContext.getBean(UsersService.class);
    }

    ...
}
```

Chapter V

Exercice 01 : Authentication

	Exercise 01
	Authentication

Turn-in directory : *ex01/*
Files to turn in : `Cinema-folder`
Allowed functions : `n/a`

- Let us expand the functionality of our application by providing an authorization mechanism. You know from the previous task that for authenticated users there is a session that has user attribute with the specified value. You shall provide profile page access (the one with a single <h1>Profile</h1> tag) only to authenticated users.

- Since security rules within our application will expand, it makes sense to create a Filter that can handle any incoming requests. This filter will check for presence of the attribute in the current session. If the attribute is found, access to the requested resource (/profile in our case) shall be provided.

- Pages for /signUp and /signIn URLs may be retrieved for unauthorized requests. If the attribute is present, a user shall be redirected to /profile page. Also, in case of an unauthorized request of a page that requires an attribute, you shall return 403 (FORBIDDEN) status.

Chapter VI

Exercice 02 : JSP

|	Exercise 02
	JSP
Turn-in directory : *ex02/*	
Files to turn in : `Cinema-folder`	
Allowed functions : `n/a`	

Now you need to implement your profile page as a JSP file. The page shall display
the following current user data:

- First name

- Last name

- email

Information about the date / time / IP address of all user authentications
as a list shall also be displayed on this page.

In addition, the page shall have a user's "avatar" loading functionality. To
implement that, you shall provide for processing a POST request to /images
URL. The uploaded image shall be saved to disk. Since users can upload images
in identical files, you shall ensure the uniqueness of file names on the disk.

All uploaded images with their original names shall be available as a list
of links. When a user clicks on the link, the image shall be opened in a new
tab. An example of a profile page interface is shown below:
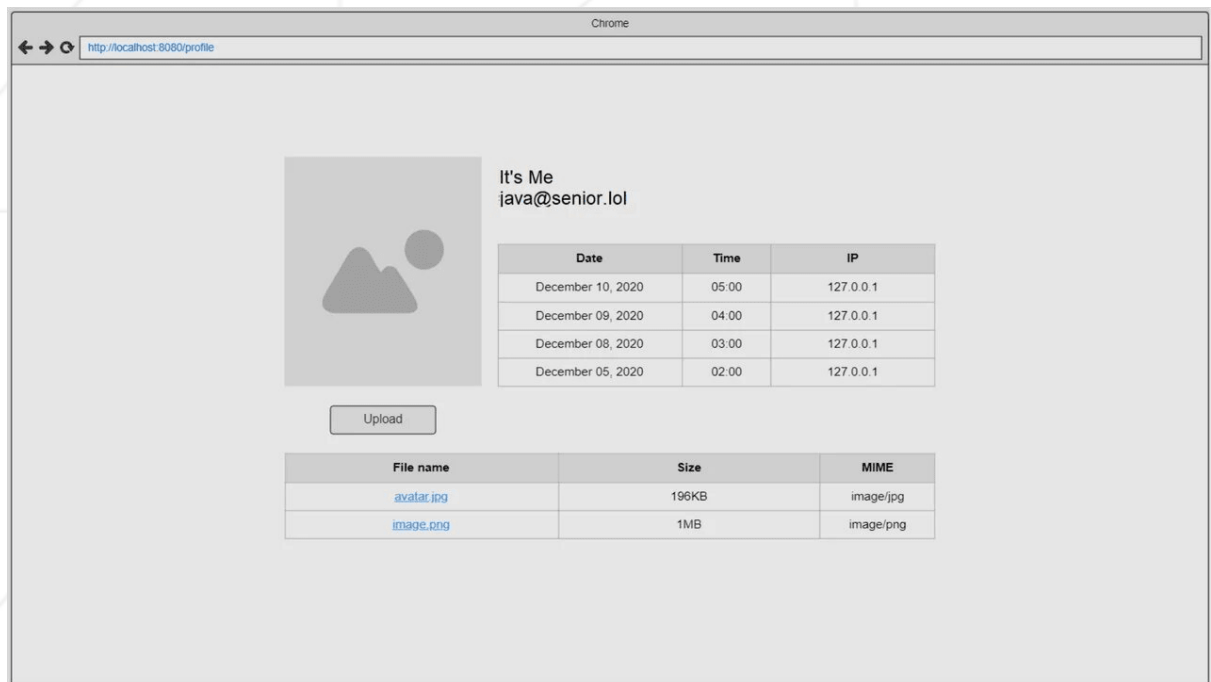
Additional requirements:

- To display a list of authentications and uploaded files, you shall use corresponding JSTL tags.

- An uploaded image shall be available via its URL - http://host:port/app-name/images/image-unique-name

- In application.properties, there shall be storage.path parameter to indicate the path to the folder where uploaded files are stored.