



FWA

Java Servlet API

Резюме: Резюме: сейчас вы разработаете свое первое веб-приложение, используя стандартные технологии Java

Версия: 1

Содержание

I	Преамбула	2
II	Инструкции	3
III	Правила проекта	5
IV	Упражнение 00: Добро пожаловать в сервлеты	7
V	Упражнение 01: Аутентификация	9
VI	Упражнение 02 : JSP	10

Глава I

Преамбула

- 200 - не беспокойтесь, все в порядке
- 400 - вы не оправдали ожиданий сервера
- 403 - вы вошли в неправильную область
- 404 - сервер не оправдал ваших ожиданий
- 500 - Поздравляем! Вы сломали сервер.
- 504 - Вы видели фильм "Хатико"?

Глава II

Инструкции

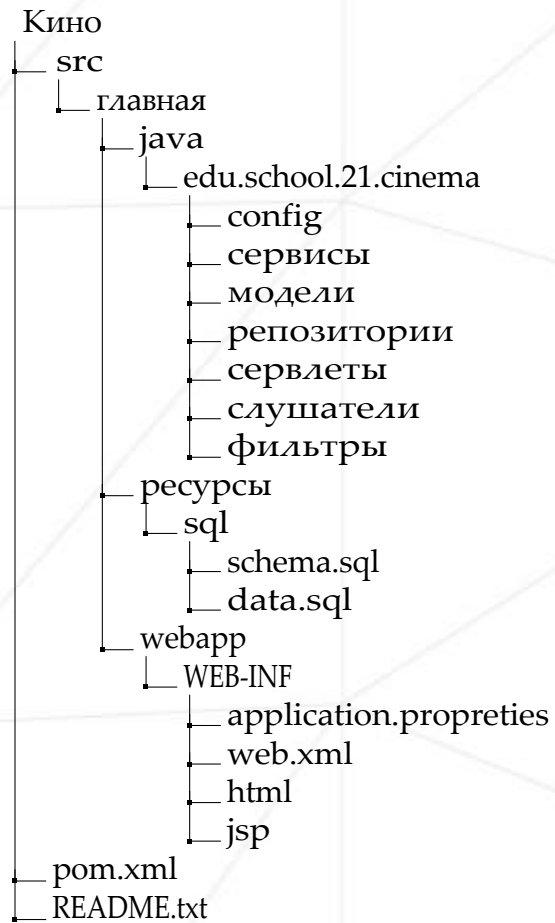
- Используйте эту страницу как единственную ссылку. Не слушайте никаких слухов и домыслов о том, как приготовить раствор.
- Теперь для вас существует только одна версия Java - 1.8. Убедитесь, что компилятор и интерпретатор этой версии установлены на вашей машине.
- Вы можете использовать IDE для написания и отладки исходного кода.
- Код чаще читают, чем пишут. Внимательно прочитайте [документ](#), в котором приведены правила форматирования кода. При выполнении каждой задачи убедитесь, что вы следуете общепринятым [стандартам Oracle](#):
- Комментарии не допускаются в исходном коде вашего решения. Они затрудняют чтение кода.
- Обратите внимание на разрешения ваших файлов и каталогов.
- Для оценки ваше решение должно находиться в вашем GIT-репозитории.
- Ваши решения будут оценивать ваши товарищи по аквариуму.
- Вы не должны оставлять в своем каталоге никаких других файлов, кроме тех, которые явно указаны в инструкциях к упражнению. Рекомендуется изменить свой .gitignore во избежание несчастных случаев.
- Когда вам нужно получить точный вывод в ваших программах, запрещено выводить предварительно рассчитанный вывод вместо правильного выполнения упражнения.
- У вас есть вопрос? Спросите своего соседа справа. В противном случае попробуйте поговорить с соседом слева.
- Ваше справочное пособие: товарищи / Интернет / Google. И еще кое-что. На любой ваш вопрос есть ответ на Stackoverflow. Узнайте, как правильно задавать вопросы.
- Внимательно прочитайте примеры. В них могут потребоваться вещи, которые не указаны в предмете.
- Для вывода используйте "System.out".

- И да пребудет с вами Сила!
- Никогда не оставляйте на завтра то, что вы можете сделать сегодня ;)

Глава III


Правила проекта

- Реализованные решения должны позволять создавать WAR-архив с помощью команды `maven package`. Такой архив должен быть развернут в Tomcat.
- В данном проекте запрещено использование компонентов Spring MVC и Hibernate (уровень репозитория должен быть реализован с помощью JdbcTemplate).
- Для каждой задачи вам нужно будет создать файл README.txt с инструкциями по развертыванию и использованию вашего приложения.
- К каждому заданию необходимо приложить файлы `schema.sql` и `data.sql`, в которых описать схему создаваемой базы данных и тестовые данные соответственно.
- Вы можете добавлять пользовательские классы и файлы в каждый из проектов, не нарушая общей предложенной структуры:



Глава IV

Упражнение 00 : Добро пожаловать в сервлеты

	Упражнение 00
Первое веб-приложение	
Входящий каталог : <i>ex00/</i>	
Файлы для сдачи :	
Кинопапка	
Разрешенные функции : н/а	

Вам необходимо разработать прототип веб-приложения с использованием стека Java Servlet API. В дальнейшем приложение будет автоматизировать бизнес-процесс бронирования билетов в кинотеатре.

Теперь вы разработаете MVP-приложение для частичной реализации механизмов регистрации и аутентификации.

Таким образом, ваше веб-приложение должно предоставлять HTML-страницы регистрации и аутентификации в ответ на URL-запросы `/signIn` и `/signUp` соответственно.

При регистрации пользователь указывает следующие данные:

- имя
- фамилия
- номер телефона
- пароль

Все данные должны поступать на сервлет `SignUp` в POST-запросе с использованием HTML-тега `<form>`. Информация хранится в базе данных, а пароль должен быть зашифрован с помощью алгоритма `BCrypt`.

Когда POST-запрос отправляется на сервлет `SignIn` с электронной почтой и паролем, выполняется проверка, существует ли соответствующий

пользователь в базе данных, а также
что их пароль верен. Если проверка прошла успешно, объект HttpSession

с атрибутом user будет сгенерирован (значение атрибута - объект, содержащий текущие данные пользователя). Пользователь будет перенаправлен на пустую страницу профиля. В случае неудачной аутентификации пользователь должен быть перенаправлен обратно на страницу входа.

Технические требования:

Spring контекст приложения должен быть отдельным конфигурационным классом (см. Spring Java Config), доступным для всех сервлетов через ServletContextListener.

В этой конфигурации необходимо указать .bin файлы для подключения к базе данных (DataSource) и шифрования паролей (PasswordEncoder), а также для всех сервисов и репозиторий. Данные для подключения к базе данных должны быть доступны в файле application.properties.

Вот пример использования этой конфигурации в сервлете:


```
@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);
    }

    ...
}
```


Глава V

Упражнение 01 : Аутентификация


	Упражнение 01
Аутентификация	
Входящий каталог : <i>ex01/</i>	
Файлы для сдачи :	
Кинопапка	
Разрешенные функции : н/а	

- Давайте расширим функциональность нашего приложения, предоставив механизм авторизации. Из предыдущего задания известно, что для аутентифицированных пользователей существует сессия, которая имеет атрибут `user` с заданным значением. Вы должны предоставить доступ к странице профиля (той, что имеет один тег `<h1>Profile</h1>`) только аутентифицированным пользователям.
- Поскольку правила безопасности в нашем приложении будут расширяться, имеет смысл создать фильтр, который сможет обрабатывать любые входящие запросы. Этот фильтр будет проверять наличие атрибута в текущей сессии. Если атрибут найден, будет предоставлен доступ к запрашиваемому ресурсу (`/profile` в нашем случае).
- Страницы для URL `/signUp` и `/signIn` могут быть получены при несанкционированных запросах. Если атрибут присутствует, пользователь должен быть перенаправлен на страницу `/profile`. Также, в случае несанкционированного запроса страницы, требующей атрибут, должен быть возвращен статус 403 (FORBIDDEN).

Глава VI

Упражнение

02 : JSP

	Упражнение 02
Входящий каталог : <i>ex02/</i> JSP	
Файлы для сдачи :	
Кинопапка	
Разрешенные функции : н/а	

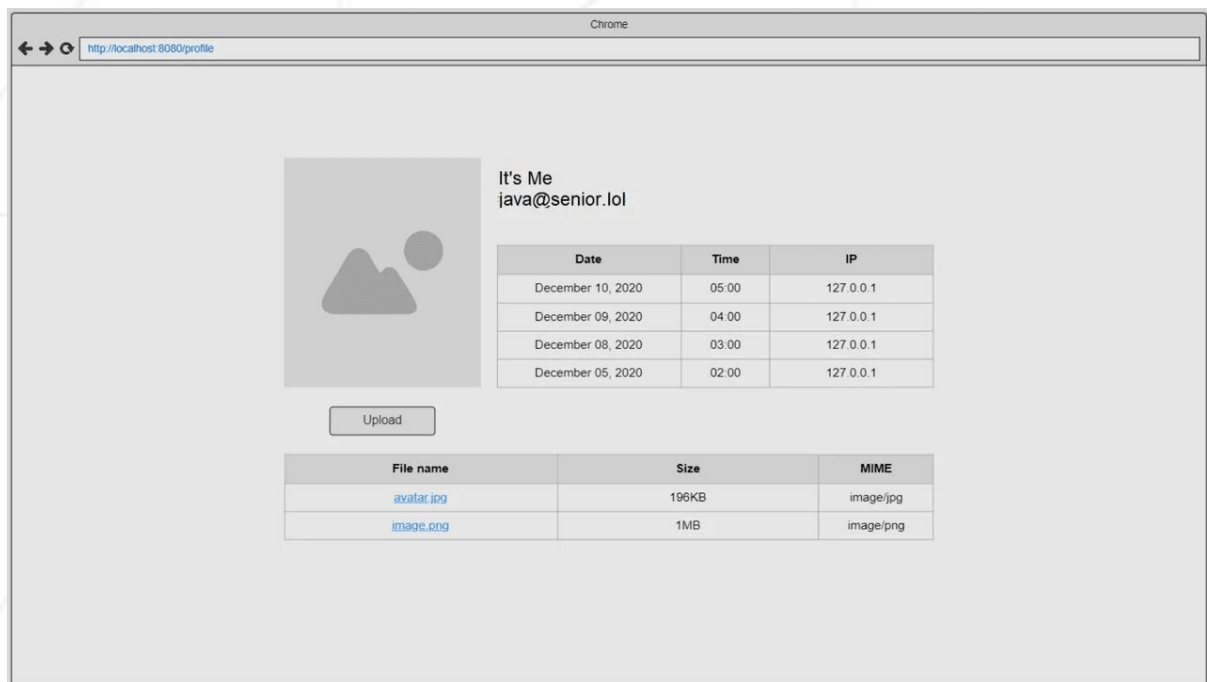
Теперь необходимо реализовать страницу профиля в виде JSP-файла. Страница должна отображать следующие текущие данные пользователя:

- Имя
- Фамилия
- электронная почта

На этой странице также отображается информация о дате / времени / IP-адресе всех аутентификаций пользователей в виде списка.

Кроме того, страница должна иметь функцию загрузки "аватара" пользователя. Для ее реализации необходимо предусмотреть обработку POST-запроса к URL /images. Загруженное изображение должно быть сохранено на диск. Поскольку пользователи могут загружать изображения в одинаковых файлах, необходимо обеспечить уникальность имен файлов на диске.

Все загруженные изображения с их оригинальными названиями должны быть доступны в виде списка ссылок. Когда пользователь нажимает на ссылку, изображение должно открываться в новой вкладке. Пример интерфейса страницы профиля показан ниже:



Дополнительные требования:

- Для отображения списка аутентификаций и загруженных файлов необходимо использовать соответствующие теги JSTL.
- Загруженное изображение должно быть доступно по его URL - `http://host:port/app-name/images/image-unique-name`.
- В `application.properties` должен быть параметр `storage.path`, указывающий путь к папке, в которой хранятся загруженные файлы.