

FWA

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

## Инструкции

- Используйте эту страницу как единственную ссылку. Не слушайте никаких слухов и домыслов о том, как приготовить свой раствор.

- Теперь у вас есть только одна версия Java, 1.8. Убедитесь, что на вашем компьютере установлены компилятор и интерпретатор этой версии.

- Вы можете использовать IDE для написания и отладки исходного кода.

- Код читается чаще, чем пишется. Внимательно прочитайте документ, в котором приведены правила форматирования кода.

При выполнении каждой задачи убедитесь, что вы следуете общепринятым стандартам Oracle:

- Комментарии в исходном коде вашего решения запрещены. Они затрудняют чтение кода.

- Обратите внимание на права доступа к вашим файлам и каталогам.

- Для оценки ваше решение должно находиться в вашем репозитории GIT.

- Ваши решения будут оценены вашими товарищами по рыбе.

- Вы не должны оставлять в своем каталоге никакие другие файлы, кроме тех, которые явно указаны в инструкциях к упражнению.

Рекомендуется изменить ваш .gitignore, чтобы избежать несчастных случаев.

- Когда вам нужно получить точный вывод в ваших программах, запрещается отображать предварительно рассчитанный вывод вместо правильного выполнения упражнения.

- Есть вопрос? Спросите у соседа справа. В противном случае попробуйте с соседом слева.

- Ваш справочник: гугля/интернет/гугл. И еще кое-что. На Stackoverflow есть ответ на любой вопрос, который у вас может возникнуть. Научитесь правильно задавать вопросы.

- Внимательно прочитайте примеры. Они могут потребовать вещи, которые не указаны в теме.

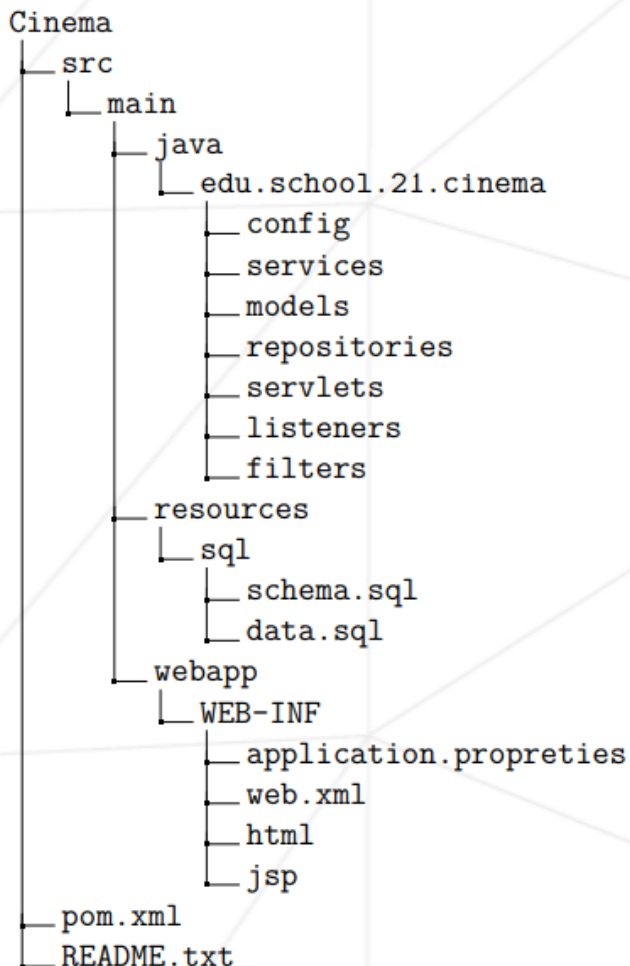
- Используйте "System.out" для вывода.

- И да прибудет с вами Сила!


- Никогда не оставляйте на завтра то, что можно сделать сегодня ;)

## 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 (слой репозитория должен быть реализован с использованием `JdbcTemplate`).
- Для каждой задачи вам потребуется создать файл `README.txt` с инструкциями по развертыванию и использованию вашего приложения.
- К каждой задаче вы должны прикрепить файлы `schema.sql` и `data.sql`, в которых вы описываете схему создаваемой базы данных и тестовые данные соответственно.
- Вы можете добавлять пользовательские классы и файлы в каждый из проектов, не нарушая общей предлагаемой структуры:



## Упражнение 00: Добро пожаловать в сервлеты

	Exercise 00
First Web Application	
Turn-in directory : <i>ex00/</i>	
Files to turn in : <b>Cinema-folder</b>	
Allowed functions : <i>n/a</i>	

Вам необходимо разработать прототип веб-приложения с использованием стека Java Servlet API.

В дальнейшем приложение автоматизирует бизнес-процесс бронирования кинотеатра. Теперь вы разработаете приложение MVP 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 передаваться сервлету `SignUp` в запросе POST с использованием HTML-тега `<form>`.

Информация хранится в базе данных, а пароль шифруется с использованием алгоритма BCrypt.

Когда запрос POST отправляется сервлету `SignIn` с адресом электронной почты и паролем, выполняется проверка, существует ли соответствующий пользователь в базе данных, а также правильность его пароля.

В случае успешной проверки должен быть сгенерирован объект `HttpSession` с пользовательским атрибутом (значение атрибута — объект, содержащий текущие пользовательские данные).

Пользователь будет перенаправлен на пустую страницу профиля.

В случае неудачной аутентификации пользователь должен быть перенаправлен обратно на страницу входа.

Технические требования: Контекст Spring приложения должен быть отдельным классом конфигурации (см. Spring Java Config), доступным для всех сервлетов через `ServletContextListener`.

В этой конфигурации необходимо указать файлы .bin для подключения к базе данных (DataSource) и шифрования паролей (PasswordEncoder), а также для всех сервисов и репозиториев.

Данные для подключения к базе данных должны быть доступны в 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);
    }

    ...
}
```

## Упражнение 01: Аутентификация

	Exercise 01
Authentication	
Turn-in directory : <i>ex01/</i>	
Files to turn in : <b>Cinema-folder</b>	
Allowed functions : <b>n/a</b>	


Давайте расширим функциональность нашего приложения, предоставив механизм авторизации. Из предыдущей задачи вы знаете, что для аутентифицированных пользователей существует сессия с атрибутом пользователя с указанным значением. Вы должны предоставить доступ к странице профиля (с одним тегом `<h1>Profile</h1>`) только аутентифицированным пользователям.

- Поскольку правила безопасности в нашем приложении будут расширяться, имеет смысл создать Фильтр, способный обрабатывать любые входящие запросы. Этот фильтр будет проверять наличие атрибута в текущем сеансе. Если атрибут найден, должен быть предоставлен доступ к запрошенному ресурсу (в нашем случае `/profile`).

- Страницы для URL-адресов `/signUp` и `/signIn` могут быть получены для несанкционированных запросов. При наличии атрибута пользователь будет перенаправлен на страницу `/profile`.

Также в случае несанкционированного запроса страницы, требующей атрибута, вы должны вернуть статус 403 (ЗАПРЕЩЕНО).

## Упражнение 02: JSP

	Exercise 02
JSP	
Turn-in directory : <i>ex02/</i>	
Files to turn in : <b>Cinema-folder</b>	
Allowed functions : <i>n/a</i>	

Теперь вам нужно реализовать страницу своего профиля в виде файла JSP.

На странице должны отображаться следующие текущие данные пользователя:

- Имя
- Фамилия
- адрес электронной почты

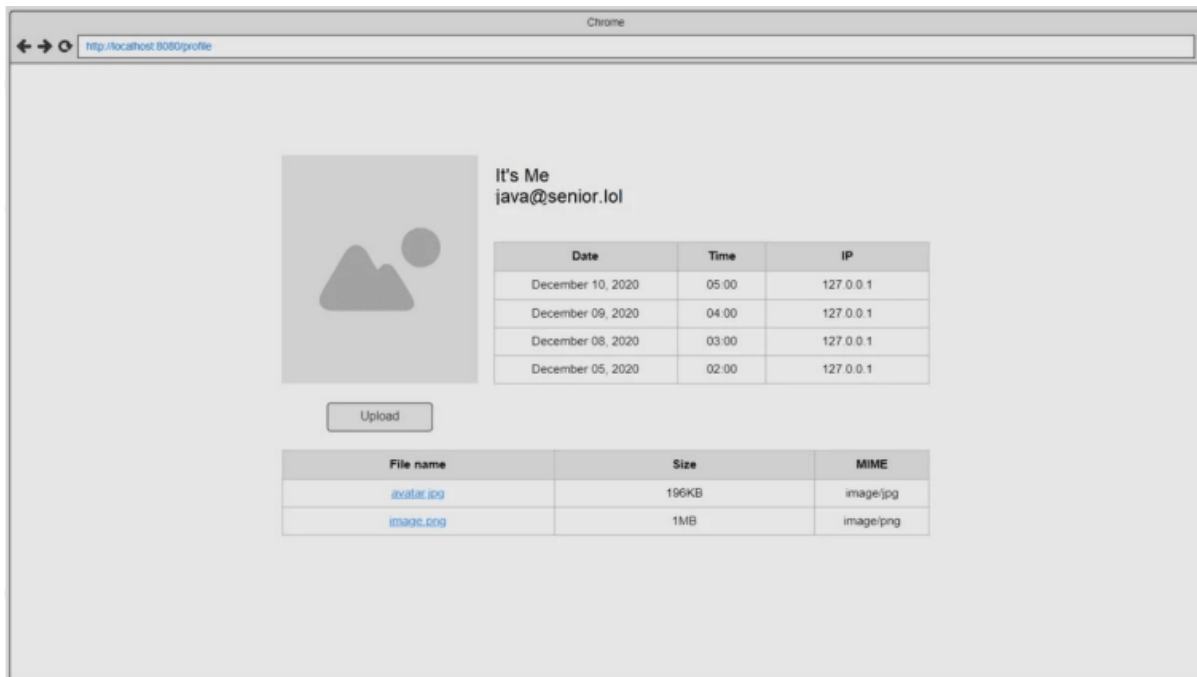
Информация о дате/времени/IP-адресе всех аутентификаций пользователей в виде списка. также должна отображаться на этой странице.

Кроме того, на странице должна быть предусмотрена функция загрузки «аватара» пользователя. Чтобы реализовать это, вы должны обеспечить обработку POST-запроса к URL-адресу `/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:



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