



Кинотеатр

Spring MVC и Hibernate

Резюме: Расширим ваши знания о веб-разработке с использованием стека Spring и освоим фреймворк Hibernate

Версия: 1

Содержание

I	Преамбула	2
II	Инструкции	3
III	Правила проекта	5
IV	Упражнение 00: Добро пожаловать в контроллеры и Hibernate	7
V	Упражнение 01 : Живой поиск	9
VI	Упражнение 02 : WebSockets	11

Глава I

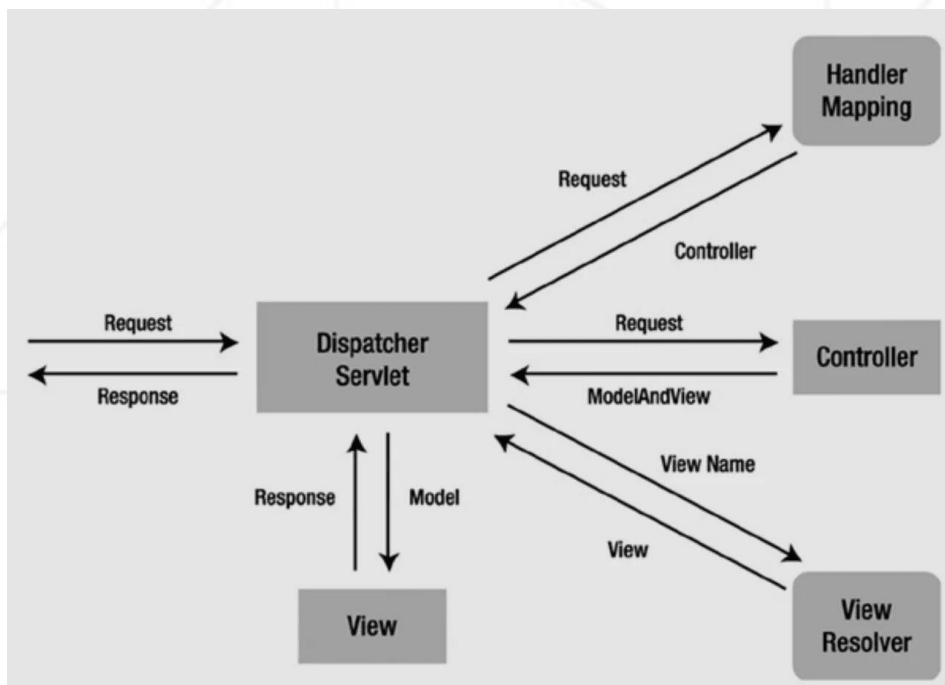
Преамбула

MVC (Model, View, Controller) - это классическая модель проектирования веб-приложений. Эта модель предполагает разделение общей логики на три компонента:

- Вид - это то, что видит пользователь (клиент).
- Модель определяет данные, которые будут включены в представление.
- Контроллер - это обработчик запроса, который генерирует представление с необходимыми данными.

В контексте Spring модель MVC реализуется с помощью классов из целого набора классов в библиотеке spring-webmvc.

Первоначально запрос поступает на DispatcherServlet. Компонент HandlerMapping затем используется для определения контроллера, который должен обработать запрос. Контроллер, в свою очередь, генерирует объект ModelAndView. Последний содержит имя требуемого представления и данные, которые должны быть использованы в этом представлении. Компонент ViewResolver выбирает необходимый файл с представлением (JSP или Freemarker-страницу) и возвращает его в DispatcherServlet. На основе полученных данных сервлет генерирует ответ для пользователя.



Глава II

Инструкции

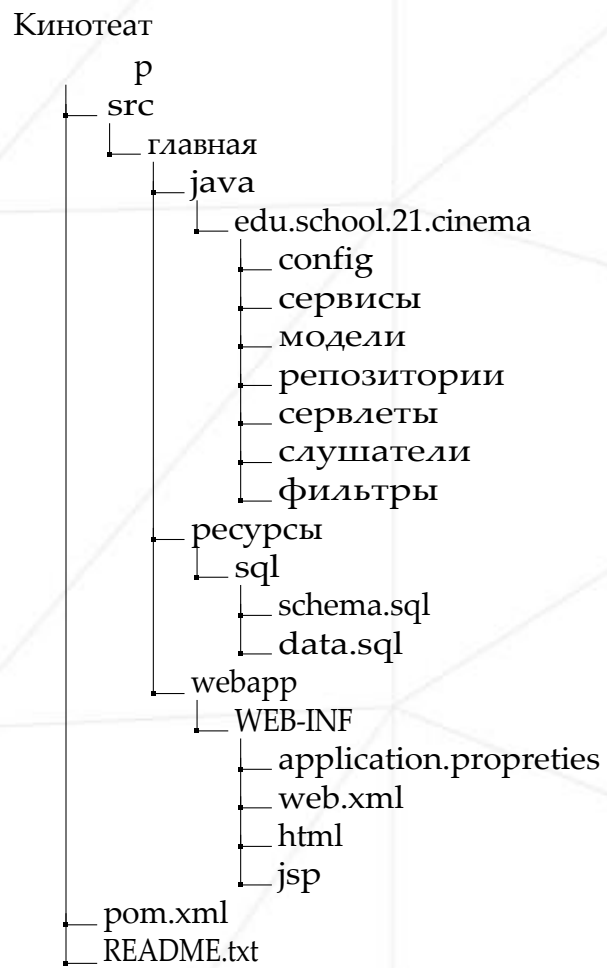
- Используйте эту страницу как единственную ссылку. Не слушайте никаких слухов и домыслов о том, как приготовить раствор.
- Теперь для вас существует только одна версия Java - 1.8. Убедитесь, что компилятор и интерпретатор этой версии установлены на вашей машине.
- Вы можете использовать IDE для написания 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

- Проект, который вы реализуете сейчас, может использовать базу данных из предыдущего задания.
- Данный проект не предполагает использования авторизации, регистрации и т.д. и реализуется независимо от предыдущего задания.
- Spring Boot запрещен в этом проекте.
- Для каждого задания 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 контроллеры и Hibernate

	Упражнение 00
Добро пожаловать в контроллеры и Hibernate	
Входящий каталог :	
ex00/	
Файлы для сдачи : Кинопапка	
Разрешенные функции : н/а	

Теперь вам нужно реализовать функциональность администратора кинотеатра, используя механизмы Spring MVC.

Ниже приведены URL-адреса каждой из страниц, а также описание необходимых функций.

- /admin/panel/halls

Страница для работы с кинозалами содержит список всех кинозалов, созданных администратором. Администратор может создать кинозал с определенной конфигурацией. Каждому кинозалу присваивается серийный номер и количество мест.

- /admin/panel/films

Страница фильма содержит список всех фильмов, созданных администратором. Администратор может добавить фильм. Для каждого фильма указывается название, год выпуска, указываются возрастные ограничения и описание. Также можно загрузить постер к фильму.

- /admin/panel/sessions

Страница для работы с киносеансами. Администратор может создать сеанс на определенный фильм в определенном кинозале в нужное время.

Администратор должен иметь возможность указать стоимость билета.

Необходимо реализовать загрузку всех данных о фильме и кинозале в виде атрибутов на страницу для последующего выбора администратором.

На этот раз слой репозитория будет реализован с использованием фреймворка Hibernate в сочетании с JPA. Таким образом, каждая из моделей должна быть аннотирована @Entity. Пример репозитория на основе Hibernate/JPA в контексте Spring приведен ниже:

```
@Repository
public class MessagesRepositoryEntityManagerImpl implements MessagesRepository {

    @PersistenceContext
    private EntityManager entityManager;

    @Override
    public List<Message> findAll() {
        return entityManager
            .createQuery("from Message", Message.class).getResultList();
    }


    @Override
    @Transactional
    public void save(Message entity) {
        entityManager.persist(entity);
    }
}
```

Технические требования:

- Каждая страница должна представлять собой шаблон Freemarker.
- Вместо сервлетов следует использовать контроллеры Spring.
- Предусмотреть реализацию WebApplicationInitializer и исключить использование web.xml.

Глава V

Упражнение 01 : Живой поиск

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

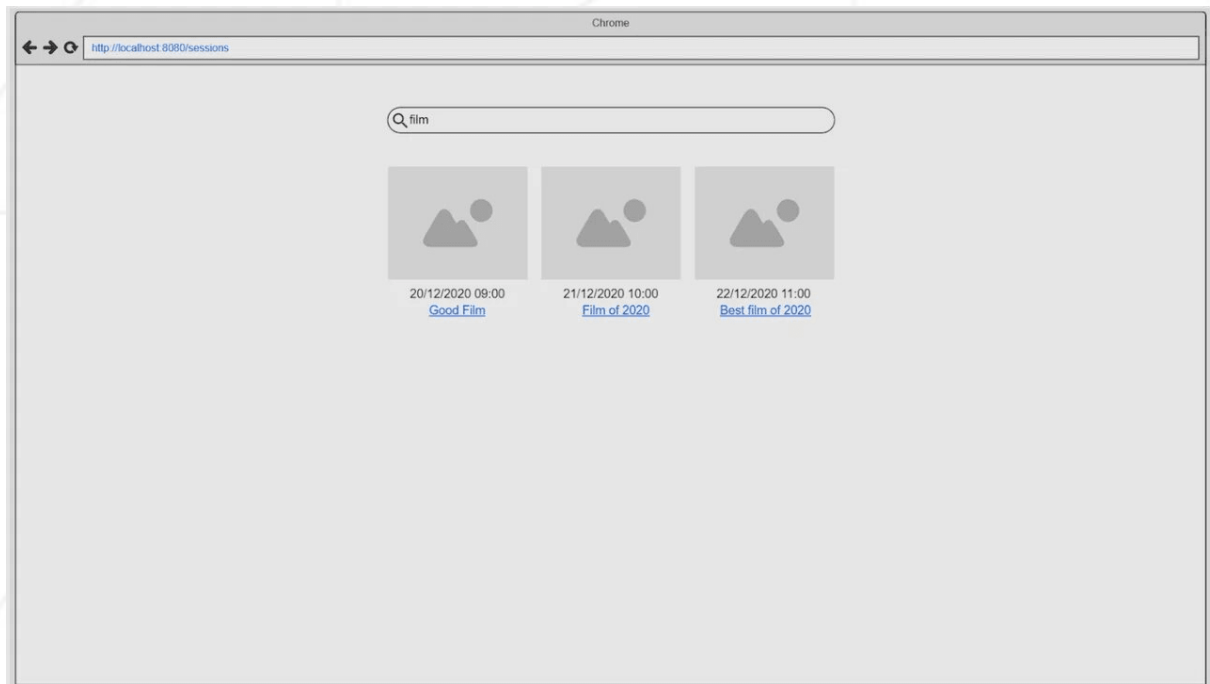
Сейчас вы изучите новую технологию фронтенда JQuery. Вы будете использовать ее для реализации AJAX-запроса, чтобы получить список всех шоу в соответствии с поисковым запросом.

Таким образом, в ответ на GET-запрос `/sessions/search?filmName=<название фильма>` сервер должен вернуть список передач с названиями фильмов, соответствующих поисковому запросу. Этот список представляет собой объект JSON в следующем виде:

```
{
  "сессии":[
    {
      "id":2,
      "dateTime": "20/12/2020 09:00",
      "film":{
        "название": "Хороший фильм",
        "posterUrl":"images/874e9d50-5cc8-41b3-90e6-9666ccc80ef1"
      }
    }
  ]
}
```

На основе этих данных на странице пользователя составляется список киносекансов


в реальном времени без полной перезагрузки страницы во время ввода данных пользователем. Пример работы страницы поиска:



- При нажатии на название фильма пользователь переходит по ссылке `/sessions/session-id`. На этой странице пользователь увидит полное описание фильма, а также информацию о кинозале, назначенном на этот сеанс.

Глава VI

Упражнение02 : WebSockets

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

Для каждого фильма мы создадим чат с обсуждением. Любой пользователь, который перейдет по ссылке `/films/{film-id}/chat`, увидит страницу с обсуждением фильма в режиме реального времени. Эта страница доступна со страницы показа фильма.

Каждому пользователю чата присваивается уникальный идентификатор, который хранится в cookies браузера. Таким образом, один пользователь будет соответствовать одному браузеру.

Требования:

- При открытии этой страницы загружаются последние 20 сообщений чата.
- Сообщения должны быть доступны после перезапуска приложения, обновления страниц в браузерах и т.д.
- Обмен сообщениями должен быть реализован с использованием протокола STOMP на основе Websockets (см. Spring Websockets STOMP). Таким образом, каждая страница подписывается на `/films/film-id/chat/messages`, а также отправляет сообщения на указанный URL. Каждое сообщение должно быть отправлено в формате JSON.

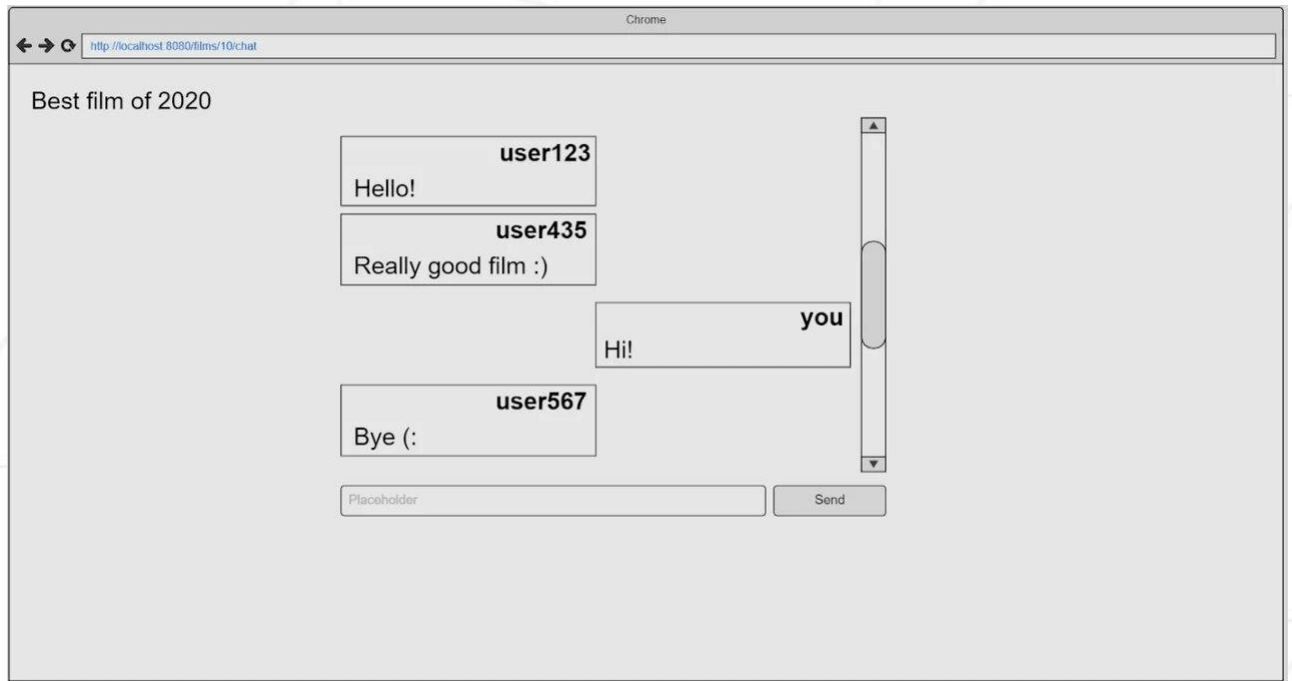
На этой странице также отображается информация о дате / времени / IP-адресе всех аутентификаций пользователей в виде списка.

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:

Пример интерфейса :



Примечание:

- Для тестирования данного приложения рекомендуется использовать актуальные версии нескольких браузеров.

