Проект 2

Техническое задание

Задачи:

1) Переписать Проект 1 с использованием Hibernate и Spring Data JPA. В вашем проекте не должно быть ни одного SQL запроса. Должны быть реализованы сущности (`@Entity`) Книга и Человек, репозитории и сервисы. `PersonDAO` и `BookDAO` должны быть пустыми и не должны использоваться, вся работа с БД через сервисы.

Новый функционал (кратко, подробности далее):

1) Добавить пагинацию для книг.
   Книг может быть много и они могут не помещаться на одной странице в браузере. Чтобы решить эту проблему, метод контроллера должен уметь выдавать не только все книги разом, но и разбивать выдачу на страницы.
2) Добавить сортировку книг по году. Метод контроллера должен уметь выдавать книги в отсортированном порядке.
3) Создать страницу поиска книг. Вводим в поле на странице начальные буквы названия книги, получаем полное название книги и имя автора. Также, если книга сейчас находится у кого-то, получаем имя этого человека.
4) Добавить автоматическую проверку на то, что человек просрочил возврат книги.

Примерное время выполнения проекта: 3-6 часов

Пагинация

Метод `index()` в `BooksController` должен уметь принимать в адресной строке два ключа: `page` и `books_per_page`. Первый ключ сообщает, какую страницу мы запрашиваем, а второй ключ сообщает, сколько книг должно быть на одной странице. Нумерация страниц стартует с 0. Если в адресной строке не передаются эти ключи, то возвращаются как обычно все книги.

Например (обычный запрос на `http://localhost:8080/books`):

Тайные виды на гору Фудзи, Владимир Пелевин, 2018
Бытие и время, Мартин Хайдеггер, 1927
Над пропастью во ржи, Джером Сэлинджер, 1951
Философия Java, Брюс Эккель, 2018
Психопатология обыденной жизни, Фрейд Зигмунд, 1904
День опричника, Владимир Сорокин, 2006
Игра в бисер, Герман Гессе, 1943

Делаем запрос `http://localhost:8080/books?page=1&books_per_page=3`

Получаем:

Философия Java, Брюс Эккель, 2018
Психопатология обыденной жизни, Фрейд Зигмунд, 1904
День опричника, Владимир Сорокин, 2006

(На каждую страницу приходится по 3 книги, мы просим вторую страницу)

Сортировка

Метод `index()` в `BooksController` должен уметь принимать в адресной строке ключ `sort_by_year`. Если он имеет значение `true`, то выдача должна быть отсортирована по году. Если в адресной строке не передается этот ключ, то книги возвращаются в обычном порядке.

Например (обычный запрос на `http://localhost:8080/books`):

Тайные виды на гору Фудзи, Владимир Пелевин, 2018
Бытие и время, Мартин Хайдеггер, 1927
Над пропастью во ржи, Джером Сэлинджер, 1951
Философия Java, Брюс Эккель, 2018
Психопатология обыденной жизни, Фрейд Зигмунд, 1904
День опричника, Владимир Сорокин, 2006
Игра в бисер, Герман Гессе, 1943

Делаем запрос `http://localhost:8080/books?sort_by_year=true`

Получаем:

Психопатология обыденной жизни, Фрейд Зигмунд, 1904
Бытие и время, Мартин Хайдеггер, 1927
Игра в бисер, Герман Гессе, 1943
Над пропастью во ржи, Джером Сэлинджер, 1951
День опричника, Владимир Сорокин, 2006
Философия Java, Брюс Эккель, 2018
Тайные виды на гору Фудзи, Владимир Пелевин, 2018

(Книги отсортированы по году)

Пагинация и сортировка могут работать одновременно (если передаются сразу три параметра в запросе)

Страница поиска

/books/search

Введите поисковой запрос: Психо

Искать!

Нажимаем

Введите поисковой запрос:

Искать!

Психопатология обыденной жизни, Фрейд Зигмунд, 1904

Книга сейчас у: Федоров Мирон Янович

Введите поисковой запрос: Философия

Искать!

Нажимаем

Введите поисковой запрос:

Искать!

Философия Java, Брюс Эккель, 2018

Книга свободна

Введите поисковой запрос: фыва

Искать!

Нажимаем

Введите поисковой запрос:

Искать!

Книг не найдено

Страница поиска

`/books/search`

- На странице поиска должна выдаваться найденная книга и текущий хозяин книги, если он есть. Если такой книги не было найдено, то должно выдаваться сообщение о том, что "Книг не найдено"
- Поиск должен производиться по начальным буквам названия книги (с помощью JPA репозитория)
- Подумайте, как передавать текст поискового запроса с формы в метод контроллера (например, с помощью аннотации `@RequestParam`)
- Подумайте, какие методы контроллера вам необходимо реализовать для этого функционала.

Проверка просрочки книги

/people/id

Федоров Мирон Янович, 1985

Книги:

Психопатология обыденной жизни, Фрейд Зигмунд, 1904

День опричника, Владимир Сорокин, 2006

Игра в бисер, Герман Гессе, 1943

Редактировать

Удалить

Если человек взял книгу более 10 дней назад и до сих пор не вернул ее, эта книга на странице этого человека должна подсвечиваться красным цветом.

Проверка просрочки книги

`/people/id`

- При взятии книги человеком, должно сохраняться текущее точное время. Вы должны изменить таблицу и сущность, чтобы сохранять время. Где создать новое поле и колонку остается на ваше усмотрение. Подумайте, как лучше.
- Вся логика по определению просрочки не должна лежать в контроллере, или что еще хуже, в представлении. Вся логика лежит в сервисе, а на представление отправляется только одно булево значение - просрочена книга (`True`) или не просрочена (`False`).
- Подумайте, где хранить и как отправлять булево значение на представление. Возможно, здесь придется кстати аннотация `@Transient`?
- Изучите атрибут `th:style` в Thymeleaf, который позволяет динамически изменять стиль на основании условия (понадобится для того, чтобы красить красным цветом просроченные книги). Можно реализовать и без `th:style`, любой способ подойдет.
- Для того, чтобы проверить, что все работает, вручную положите в таблицу значение времени, которое было раньше, чем 10 дней назад.
- Изучите, как в PostgreSQL класть в таблицу значение в колонку типа `timestamp`
- Логика определения просрочки остается на ваше усмотрение. Можно реализовать как угодно. Главное, чтобы работало.
- **Не стесняйтесь гуглить :)**

Возможные ошибки и способы их исправления

Если возникает такая ошибка при запуске:

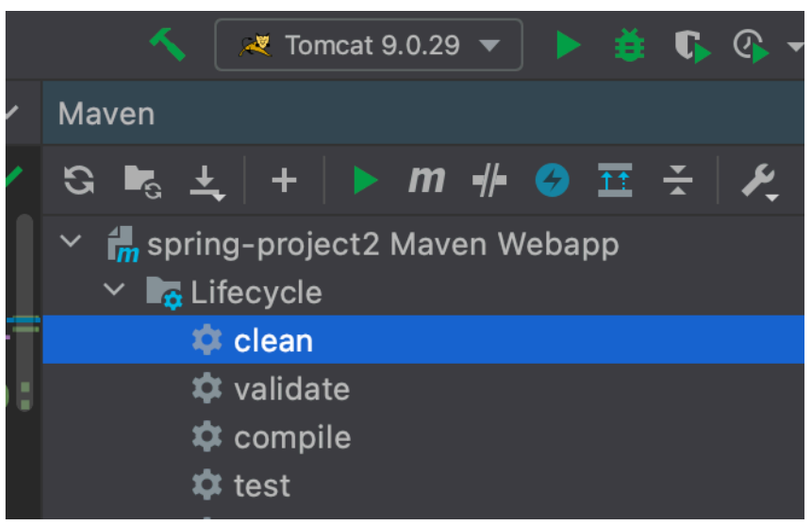
```
: Invocation of init method failed; nested exception is java.lang.NoClassDefFoundError: org/springframework/core/NativeDetector
```

Надо обновить в pom.xml зависимости Spring до последней версии (как я показывал в первом уроке по Spring Data JPA):

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>1.7</maven.compiler.source>
  <maven.compiler.target>1.7</maven.compiler.target>

  <spring.version>5.3.5</spring.version>
  <hibernate.version>5.4.28.Final</hibernate.version>
</properties>
```

После этого надо сделать maven clean



Если возникает ошибка `LazyInitializationException` при получении связанных сущностей, не забывайте вызывать `Hibernate.initialize()` на связанных сущностях в пределах транзакции, чтобы заставить Hibernate подгружать ленивые связанные сущности

```java
public List<Book> getBooksByPersonId(int id) {
    Optional<Person> person = peopleRepository.findById(id);

    if (person.isPresent()) {
        Hibernate.initialize(person.get().getBooks());
        return person.get().getBooks();
    }
    else {
        return Collections.emptyList();
    }
}
```

После этого при получении книг вне транзакции ошибки `LazyInitializationException` больше быть не должно

Помните, что просто вызвать геттер недостаточно для того, чтобы Hibernate подгрузил сущности, так как Java компилятор оптимизирует код и если данные, полученные от геттера никак не используются, то этот геттер вызван не будет. Поэтому используется `Hibernate.initialize()`

Это обсуждалось в уроке про Ленивую и Не ленивую загрузку (Eager vs Lazy loading)

Если остались вопросы - задавайте в закрытом Telegram чате