

Тестовое задание

На выбор дается 2 задания (плюс выбор front end framework за вами, из списка предлагаемых).

Результат выкладываете на GitHub или Bitbucket – заодно познакомитесь с ними.

После выполнения задания просьба зайти в свой профиль на JavaRush и заполнить ссылку на решенное задание. Файлы проекта **.idea** и скомпилированные классы в **output** или **target** выкладывать не надо.

Требуемые технологии:

- Maven (для сборки проекта);
- Tomcat 8 (для тестирования своего приложения);
- Spring (версия не ниже 4.3.0.RELEASE);
- Hibernate (версия не ниже 5.2.1.Final);
- MySQL (база данных). Для упрощения тестирования называйте все свою базу **test**, с логином и паролем **root** (нам не нужно будет для тестирования создавать кучу лишних и ненужных баз);

- Frontend: *Spring MVC or Angular or Vaadin or ZK framework*.

Версии можно смело брать самые последние. Конфликтов быть не должно.

Если копируете решение из интернета, то постарайтесь хотя бы сами разобраться что и как работает – очень поможет в дальнейшем в реальном проекте.

Приложение должно быть в виде проекта, который собирается с помощью Maven. Обязательно должен присутствовать скрипт для создания и наполнения тестовыми данными вашей базы данных. Предупреждение: скрипт на SQL – это **скрипт**, а не дамп базы данных из WorkBench, PhpMyAdmin, и прочих программ. Не ленитесь и напишите скрипт сами. В тестовые данные вставьте от 21 до 40 записей (книг или заметок).

Скрипт должен выглядеть как-то так:

```
USE test;

DROP TABLE IF EXISTS book;
CREATE TABLE book(
  id INT(11) NOT NULL AUTO_INCREMENT,
  ...
  PRIMARY KEY (id))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO book (...) VALUES (...),
(...),
...
;
```

Если для старта приложения после успешного деплоя нужны какие-то дополнительные махинации или танцы с бубном – опишите все это в каком-нибудь readme в корне проекта.

Визуальную часть проектов не рисую, т.к. хочу увидеть, как каждый сам реализует и посчитает нормальной свою реализацию задачи.

Огромная просьба: тестируйте свое приложение. Бывало много случаев, когда присылали полностью нерабочее приложение, или с огромным количеством багов.

1. CRUD (create, read, update, delete).

У вас есть всего 1 таблица book. В ней хранится список книг (например, на книжной полке). Книги на полку можно добавлять (create), брать посмотреть (read), заменять на новый выпуск (update), убирать (delete).

Данные, которые должны быть в таблице:

- id – идентификатор книги в БД;
- title – название книги. Можно использовать тип VARCHAR(100);
- description – краткое описание о чем книга. Можно использовать тип VARCHAR(255);
- author – фамилия и имя автора. Можно использовать тип VARCHAR(100);
- isbn – ISBN книги. Можно использовать тип VARCHAR(20);
- printYear – в каком году напечатана книга (INT);
- readAlready – читал ли кто-то эту книгу. Это булево поле.

Бизнес-требование: при редактировании может быть 2 варианта поведения:

- Книгу прочитали, и тогда изменяется только поле readAlready, и только, если оно было false. Значения поля должно стать true.
- Книгу заменили на новое издание. В этом случае должна быть возможность изменить title, description, isbn, printYear. А поле readAlready нужно выставить в false. Поле author должно быть неизменяемым с момента создания книги.

Необходимо реализовать стандартное CRUD приложение, которое отображает список всех книг в базе (с пейджингом по 10 книг на странице). С возможностью их удаления, редактирования, добавления новых, и поиска по уже существующим.

По какому полю искать – каждый решает для себя сам. Можно ограничиться полем title, но согласитесь, удобно просмотреть книги, которые еще не читал, или, которые вышли после 2016 года.

2. NOTES

Реализовать простенькое приложение Notes-list, для отображения списка заметок.

Нужно показывать список уже созданных заметок (с пейджингом по 10 заметок на странице). Каждую из них можно редактировать, добавлять новые, отмечать как «Выполнено», удалять. Список можно фильтровать как «Все заметки», «Только невыполненные», «Выполненные».

Заметки хранить в базе. Схему таблички для хранения нужно придумать самому (можно ограничиться одной таблицей).

Бизнес-требование: кроме фильтрации должна быть возможность сортировки заметок по дате создания (например, поле createDate в БД). Тип поля – DATE или DATETIME, или TIMESTAMP.

В посте [стажировка](#) вы можете задавать вопросы по тестовому заданию, если, например, непонятно что такое пейджинг, или, как хранить булевы значения в БД. На задания вам дается 3 недели (вполне достаточно если учесть, что они делаются и за день). Если у вас уходит много времени и что-то не получается, то не расстраивайтесь – возможно вы сильно забурились в материал, что совсем не обязательно.

Ниже приводится ссылка на архив с полезными книгами, которые помогут вам в решении тестового задания. Вам не нужно их все перечитать. Используйте как справочники.

[литература](#)