Доработка веб-приложения — интернет-магазина по продаже электронных книг

Задача 3

Цель задачи

Внедрить в разрабатываемый проект элементы управления контентом.

Что нужно сделать

CMS — Content Management System, или админка — это информационная система, используемая для процесса создания, редактирования и управления содержимым сайта — контентом. Типичная CMS обычно состоит из двух основных компонентов: приложения для управления контентом в качестве внешнего пользовательского интерфейса, позволяющего своему пользователю добавлять, изменять и удалять контент с веб-сайта без вмешательства веб-мастера, и приложение доставки контента, которое компилирует контент и обновляет веб-сайт.

Руководство магазина хочет обладать рычагами управления системой, которую вы для них разрабатываете, чтобы, когда ваша работа будет завершена, они могли нанять контент-менеджера, который будет управлять магазином инструментами CMS. Перед вами стоит задача подключить CMS-функционал к интернет-магазину книг.

Сперва необходимо определить, пойдёте ли вы по пути интеграции с готовой CMS, которых много на разный вкус и функционал, или предпочтёте реализовать собственную. Если с первым вариантом вся сложность в том, как подружить чужой код с вашим собственным, то второй подход имеет несколько вариаций. Вы можете разработать отделённый от основного интерфейс, который будет доступен только контент-менеджеру, и далее реализовать некоторый функционал по управлению магазином:

- добавление и удаление книг;
- смену обложек книг;
- редактирование информации о книгах и авторах;
- модерацию отзывов и пользователей;

- редактирование книжных полок пользователей (например, чтобы потом была возможность добавлять подарки в рамках промоакций).

Такой подход потребует написать много логики на бэкенде и сверстать ещё больше кода фронтенд-составляющей этого вопроса. Чтобы облегчить задачу, мы можем не разрабатывать интерфейс админки отдельно, а реализовать его прямо в интерфейсе магазина. Тут вам пригодится использование тех authorities, о которых мы говорили, когда изучали вопросы обеспечения безопасности Spring-приложений.

Если мы привяжем пользователей к определённым ролям, то сможем показывать разный контент в зависимости от того, какой ролью обладает пользователь: ANONYMOUS, не имеющий учётной записи, простой USER, зарегистрировавший учётку, или ADMIN. Вы уже сделали первый шаг в этом направлении, когда добавляли функционал смены обложки, чтобы протестировать механизм загрузки изображений на сервер. Теперь вы можете разместить по всему магазину дополнительные элементы для реализации CMS-функций и при помощи Thymeleaf определять, когда их показывать, в зависимости от authorities, которыми обладает клиент, выполняющий в данный момент запросы к магазину. Выберите и реализуйте наиболее подходящий и интересный для вас подход.

Задача 4

Цель задачи

Повышение качества исходных кодов ПО.

Что нужно сделать

Писать качественный и оптимизированный код сразу и без ошибок возможно, но это требует многолетнего опыта в пределах наиболее часто решаемой в профессиональной практике задачи. Например, если вы будете годами разрабатывать исключительно интернет-магазины, то в какой-то момент наберёте достаточно опыта и реализованных кейсов, чтобы из ваших рук и сразу в продакшен струился чистый, как капля горной росы, и без единой ошибки код.

Но разрабатывать можно не только интернет-магазины, а кодовая база среднестатистического веб-проекта на Spring может быть очень внушительной и довольно трудной даже для мысленного охвата. Это приводит к багам и сложностям в разработке и поддержке, а человеческий фактор обусловливает вливание в проект разного по качеству исполнения кода, который при

наслаивании друг на друга провоцирует новые баги, часто с эффектом «бомбы замедленного действия».

Чтобы держать ситуацию под контролем, необходимо как минимум следить за этим вливанием кода в проект и вовремя реагировать. Фактически это означает необходимость проводить code review всех исходников программы, учитывая огромное количество критериев, определяющих тот самый «качественный код», который мы в итоге и хотим получить. Таких критериев много. (Вы можете убедиться в этом сами, прочитав книгу Роберта Мартина «Чистый код».) Для выполнения такой задачи было написано множество программ — статических анализаторов, автоматизирующих весь процесс проверки исходного кода. Одна из них — программный продукт SonarQube — платформа с открытым исходным кодом для анализа и измерения качества программного кода. Задачу можно разбить на несколько этапов:

Этап 1

Ознакомьтесь с документацией 😁

Этап 2

Скачайте и распакуйте сервер SonarQube.

Этап 3

Запустите сервер SonarQube одним из способов в зависимости от вашей операционной системы:

- Windows:
  - Из командной строки перейдите в каталог с распакованным сервером SonarQube, далее перейдите до \sonarqube\bin\windows-x86-64\ и запустите выполнение пакетного файла StartSonar.bat.
- Unix-подобные:
  - Через терминал перейдите в каталог с распакованным сервером SonarQube, далее перейдите в /sonarqube/bin/[OS] и запустите выполнение bash-скрипта sonar.sh.

Этап 4

Теперь можно проверить работу сервера SonarQube по адресу http://localhost:9000, где вас встретит login-форма. По умолчанию создаётся пользователь Administrator, для которого устанавливается логин admin и пароль admin 😅 После входа на панель sonar-сервера вам необходимо сгенерировать security-токен, чтобы maven, при помощи которого мы загрузим код проекта в анализатор, мог взаимодействовать с SonarQube.

👉 Подробная инструкция

Этап 5

В pom.xml проекта добавьте плагин sonar-maven-plugin:

```
<plugins>

....

            <plugin>

                <groupId>org.sonarsource.scanner.maven</groupId>

                <artifactId>sonar-maven-plugin</artifactId>

                <version>3.4.0.905</version>

            </plugin>

</plugins>
```

Этап 6

Теперь остатся только собрать проект (команда — mvn clean install) и запустить анализатор кода при помощи sonar-maven-plugin по команде:

```
mvn sonar:sonar -Dsonar.host.url=http://localhost:9000 -Dsonar.login=sonar-токен
```

Решение на случай проблем с mvn для пользователей Windows.

Решение на случай проблем с mvn для пользователей Linux.

Этап 7

Вернитесь на http://localhost:9000 (панель управления SonarQube) и прочитайте отчёт об анализе кода проекта и выполните работы по устранению выявленных замечаний 🥳 Fin!

Формат сдачи материалов и оценивание

Готовый дипломный проект и его промежуточные итерации принимаются в формате git-репозитория с историей коммитов. Промежуточные итерации подлежат обсуждению с дипломным руководителем по мере появления вопросов и необходимости консультации. Готовый дипломный проект после утверждения руководителем подлежит защите перед комиссией преподавателей курса.

Рекомендации по выполнению

В работе над дипломным проектом руководствуйтесь техническим заданием. Старайтесь формировать коммиты таким образом, чтобы их вклад в репозиторий был не слишком большим, когда историю составляют два коммита: initial и code_all_stuff. Но и слишком незначительный вклад коммита, вроде clean_old_comments, нежелателен и будет только захламлять общую картину. Также стоит придерживаться правила, что коммита в репозиторий достоин только проверенный на работоспособность код. Иными словами, если ваш новый commit ломает текущую стабильную версию, то лучше не делать такой коммит, сберечь внешний вид репозитория и сначала довести код до рабочего состояния на localhost. Старайтесь использовать в названии коммита ёмкие и не слишком длинные (в районе 30 символов) фразы. Детали можно указать в message-области коммита.