

Содержание

[Введение 3](#_Toc99441863)

[1 Сведения о программном обеспечении 4](#_Toc99441864)

[1.1 IntelliJ IDEA 4](#_Toc99441865)

[1.2 SpringFramework 5](#_Toc99441866)

[2 Реализация сервера 7](#_Toc99441867)

[Заключение 11](#_Toc99441868)

Введение

Целью расчетно-графической работы является знакомство с базовым функционалом экосистемы Spring.

Для достижения поставленной цели необходимо реализовать сервер со следующим функционалом:

* авторизация и регистрация;
* взаимодействие с базой данных (СУБД не регламентируется), хранящей информацию о не менее, чем трех связанных сущностях;
* взаимодействовал с клиентами согласно принципам REST-архитектуры.

Кроме того, необходимо обеспечить доступ к серверу посредством сети Интернет (способ не регламентируется).

1. Сведения о программном обеспечении
   1. IntelliJ IDEA

IntelliJ IDEA – это интеллектуальная IDE, учитывающая контекст. Она предназначена для разработки разнообразных приложений на Java и других языках JVM, например Kotlin, Scala и Groovy. Кроме того, IntelliJ IDEA Ultimate поможет в разработке веб-приложений: она предлагает эффективные встроенные инструменты, поддержку JavaScript и связанных с ним технологий, а также расширенную поддержку таких популярных фреймворков, как Spring, Spring Boot, Jakarta EE, Micronaut, Quarkus и Helidon. А бесплатные плагины, разработанные JetBrains, позволяют дополнительно расширить возможности IntelliJ IDEA и использовать ее для работы с другими языками программирования, в том числе Go, Python, SQL, Ruby и PHP.

IntelliJ IDEA продумана в каждом аспекте и готова к использованию сразу после установки. Среда обеспечивает быстрый доступ ко всем функциям и встроенным инструментам, необходимым разработчику, а также широкие возможности индивидуальной настройки. Вы можете полностью настроить среду в соответствии со своим рабочим процессом: задать сочетания клавиш, установить плагины, настроить интерфейс по своему усмотрению и т. д.

JetBrains стремится сделать свои продукты доступными для всех желающих. IntelliJ IDEA предлагает различные специальные возможности, которые могут вам понадобиться. Например, с IntelliJ IDEA совместимы программы чтения с экрана. Можно также настроить цвета разных элементов интерфейса, добавить контрастности полосам прокрутки, изменить размер окон и шрифта в редакторе и т. п.

IntelliJ IDEA была создана в первую очередь для разработки на Java, но она понимает и многие другие языки программирования, в том числе Groovy, Kotlin, Scala, JavaScript, TypeScript и SQL, и предлагает интеллектуальную помощь в написании кода на каждом из этих языков. Первоначальная индексация исходного кода позволяет IDE создать виртуальную карту проекта. Используя информацию виртуальной карты, она мгновенно обнаруживает ошибки, предлагает варианты автодополнения кода с учетом контекста, выполняет рефакторинг и т. д.

IntelliJ IDEA обладает широкими возможностями проверки качества и валидности кода с помощью инспекций, которые выполняются «на лету». Инспекции помогают быстрее писать код в соответствии с самыми строгими стандартами качества и чувствовать себя уверенно на протяжении всего процесса разработки. Мы предлагаем набор стандартных инспекций и десятки инспекций для отдельных фреймворков. Они позволяют выявить самые разные проблемы: например, ошибки при автоматическом связывании бинов Spring и т. п.

* 1. SpringFramework

SpringFramework (или коротко Spring) — универсальный фреймворк с от-крытым исходным кодом для Java-платформы. Также существует форк для платформы «.NETFramework», названный «Spring.NET».

Первая версия была написана Родом Джонсоном, который впервые опубликовал её вместе с изданием своей книги «ExpertOne-on-OneJavaEEDesignandDevelopment» (WroxPress, октябрь 2002 года).

Фреймворк был впервые выпущен под лицензией Apache 2.0 license в июне 2003 года. Первая стабильная версия 1.0 была выпущена в марте 2004. Spring 2.0 был выпущен в октябре 2006, Spring 2.5 – в ноябре 2007, Spring 3.0 в декабре 2009, и Spring 3.1 в декабре 2011. Текущая версия – 5.3.x.

Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели EnterpriseJavaBeans. Spring предоставляет большую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на JavaEnterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

Spring обеспечивает решения многих задач, с которыми сталкиваются Java-разработчики и организации, которые хотят создать информационную систему, основанную на платформе Java. Из-за широкой функциональности трудно определить наиболее значимые структурные элементы, из которых он состоит. Spring не всецело связан с платформой Java Enterprise, несмотря на свою масштабную интеграцию с ней, что является важной причиной его популярности.

Spring, вероятно, наиболее известен как источник расширений (features), нужных для эффективной разработки сложных бизнес-приложений вне тяжеловесных программных моделей, которые исторически были доминирующими в промышленности. Ещё одно его достоинство в том, что он ввел ранее неиспользуемые функциональные возможности в сегодняшние господствующие методы разработки, даже вне платформы Java.

Этот фреймворк предлагает последовательную модель и делает её применимой к большинству типов приложений, которые уже созданы на основе платформы Java. Считается, что Spring реализует модель разработки, основанную на лучших стандартах индустрии, и делает её доступной во многих областях Java.

1. Реализация сервера

В листинге ниже представлена программная реализация задачи.

Листинг 1 – UserEntity

package com.example.restspring.entity;

import javax.persistence.\*;

import java.util.List;

@Entity

@Table(name = "user")

public class UserEntity {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String username;

private String password;

@OneToMany(cascade = CascadeType.ALL, mappedBy = "user")

private List<BoardEntity> boards;

public UserEntity() {

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public List<BoardEntity> getBoards() {

return boards;

}

public void setBoards(List<BoardEntity> boards) {

this.boards = boards;

}

}

Листинг 2 – UserModel

package com.example.restspring.model;

public class User {

private Long id;

private String username;

private List<Board> boards;

public static User toModel(UserEntity entity) {

User model = new User();

model.setId(entity.getId());

model.setUsername(entity.getUsername());

model.setBoards(entity.getBoards().stream().map(Board::toModel).collect(Collectors.toList()));

return model;

}

public User() {

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public List<Board> getBoards() {

return boards;

}

public void setBoards(List<Board> boards) {

this.boards = boards;

}

}

Листинг 4 – UserRepo

public interface UserRepo extends CrudRepository<UserEntity, Long> {

UserEntity findByUsername(String username);

}

Листинг 5 – UserController

@RestController

@RequestMapping("/users")

public class UserController {

@Autowired

private UserService userService;

@PostMapping("/registration")

public ResponseEntity registration(@RequestBody UserEntity user) {

try {

userService.registration(user);

return ResponseEntity.ok("Пользователь успешно сохранен");

} catch (UserAlreadyExistException e) {

return ResponseEntity.badRequest().body(e.getMessage());

} catch (Exception e) {

return ResponseEntity.badRequest().body("Произошла ошибка");

}

}

@PostMapping("/login")

public ResponseEntity login(@RequestBody UserEntity user) {

try {

User savedUser = userService.login(user.getUsername(), user.getPassword());

return ResponseEntity.ok(savedUser);

} catch (UserNotFoundException e) {

return ResponseEntity.badRequest().body(e.getMessage());

} catch (Exception e) {

return ResponseEntity.badRequest().body("Произошла ошибка");

}

}

}

Листинг 6 – UserService

package com.example.restspring.service;

@Service

public class UserService {

@Autowired

private UserRepo userRepo;

public UserEntity registration(UserEntity user) throws UserAlreadyExistException {

if (userRepo.findByUsername(user.getUsername()) != null) {

throw new UserAlreadyExistException("Пользователь с таким именем существует");

}

return userRepo.save(user);

}

public User login(String login, String password) throws UserNotFoundException {

UserEntity user = userRepo.findByUsername(login);

if (user == null) {

throw new UserNotFoundException("Некорректный логин и/или пароль");

}

if (!user.getPassword().equals(password)) {

throw new UserNotFoundException("Некорректный логин и/или пароль");

}

return User.toModel(user);

}

}

Листинг 6 – UserAlreadyExistException

package com.example.restspring.exception;

public class UserAlreadyExistException extends Exception {

public UserAlreadyExistException(String message) {

super(message);

}

}

На этом работа закончена.

Заключение

Язык программирования Java – это полностью объектно-ориентированный язык, который в отношении синтаксиса многое унаследовал от С++. Конечно, преимущества Java далеко не исчерпываются межплатформенностью. Язык Java в синтаксическом отношении проще и логичнее, чем С++. Java как платформа предоставляет в распоряжение программистов большое количество библиотек (пакетов), в которых содержится большое количество описаний классов и интерфейсов на все случаи жизни. С их помощью можно создавать стопроцентные приложения Java с возможностью обращения к базам данных, поддержкой передачи почтовых сообщений, с клиентской частью, которой необходим web-браузер, или, наоборот, с клиентской частью, обладающей изощренным интерфейсом.

Результатом расчетно-графической работы стало программное, реализованное с помощью фреймфорка Spring.