Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
|  |
| Кафедра «Информатика» |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

|  |
| --- |
| Настройка Hibernate |
| Тема |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  |  |  | А. К. Погребников |
|  | |  |  |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ19-16/1б 031939175 | | |  |  |  | А. Д. Непомнящий |
|  | номер группы, зачетной книжки | | |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2021

1. Цель работы

Цель работы состоит в получении навыков связи данных между базой данных и Java-приложением с использованием ORM-прослойки Hibernate.

1. Задачи

Выполнение работы сводится к следующим задачам.

1. Создать Maven-проект веб-приложения.
2. Подключить необходимые зависимости.
3. Настроить подключение к БД.
4. Реализовать несколько запросов.
5. Ход работы
   1. Зависимости

Был создан Maven-проект и были подключены добавлением в файл pom.xml, помимо прочих, зависимости, представленные в листинге 1.

Листинг 1 – Зависимости

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-jpamodelgen</artifactId>

<version>6.0.0.Alpha7</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>5.3.10</version>

</dependency>

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-jpa</artifactId>

<version>2.5.5</version>

</dependency>

Окончание листинга 1

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>42.2.24</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.3.9</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.9</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

<version>6.0.0.Alpha7</version>

<type>pom</type>

</dependency>

* 1. Подключение к БД

Подключение к БД устанавливается за счет необходимой конфигурации в классе SpringConfiguration, приведенном в листинге 2.

Листинг 2 – Код класса SpringConfiguration

@Configuration

@ComponentScan("com.github.durakin.serverprogramming.lab4")

@PropertySource("classpath:application.properties")

@EnableJpaRepositories("com.github.durakin.serverprogramming.lab4.repository")

@RequiredArgsConstructor

public class SpringConfiguration {

private final Environment env;

@Bean

public DataSource dataSource() {

Продолжение листинга 2

DriverManagerDataSource dataSource = new DriverManagerDataSource();

dataSource.setDriverClassName(Objects.requireNonNull(env.getProperty("spring.datasource.driverClassName")));

dataSource.setUrl(env.getProperty("spring.dataSource.url"));

dataSource.setUsername(env.getProperty("spring.dataSource.username"));

dataSource.setPassword(env.getProperty("spring.dataSource.password"));

return dataSource;

}

@Bean

public Properties jpaProperties() {

return new Properties();

}

@Bean

@Autowired

public EntityManagerFactory entityManagerFactory(DataSource dataSource, Properties jpaProperties) {

HibernateJpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();

LocalContainerEntityManagerFactoryBean factory = new LocalContainerEntityManagerFactoryBean();

factory.setJpaVendorAdapter(vendorAdapter);

factory.setPackagesToScan("com.github.durakin.serverprogramming.lab4");

factory.setDataSource(dataSource);

factory.setJpaProperties(jpaProperties);

factory.afterPropertiesSet();

return factory.getObject();

}

@Bean

@Autowired

public PlatformTransactionManager transactionManager(EntityManagerFactory entityManagerFactory) {

JpaTransactionManager txManager = new JpaTransactionManager();

txManager.setEntityManagerFactory(entityManagerFactory);

return txManager;

}

Окончание листинга 2

@Bean

public SessionFactory localSessionFactory()

{

LocalSessionFactoryBuilder builder = new LocalSessionFactoryBuilder(dataSource());

builder.scanPackages("com.github.durakin.serverprogramming.lab4");

return builder.buildSessionFactory();

}

}

* 1. Реализация запросов

На листингах 3, 4, 5 и 6 приведен код для получения объекта фракции по названию – код сущности, репозитория, службы для вызова метода из репозитория и вызов метода для окончательного получения сущности соответственно.

Листинг 3 – Код класса SpringConfiguration

@Table(name = "factions")

@Entity

public class Faction {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", nullable = false)

private Integer id;

@Lob

@Type(type = "org.hibernate.type.TextType")

@Column(name = "name")

private String name;

@ManyToOne (fetch = FetchType.LAZY)

@JoinColumn(name = "home\_system\_id")

private System homeSystem;

@ManyToOne

@JoinColumn(name = "allegiance\_id")

private Allegiance allegiance;

Окончание листинга 3

@ManyToOne

@JoinColumn(name = "government\_id")

private Government government;

@Column(name = "is\_player\_faction")

private Boolean isPlayerFaction;

//Get и Set методы

@Override

public String toString() {

final StringBuilder sb = new StringBuilder("Faction ");

sb.append(name);

sb.append(" with allegiance ").append(allegiance);

sb.append(" and ").append(government);

sb.append(" government");

return sb.toString();

}

}

Листинг 4 – Код класса FactionRepository

@Repository

public interface FactionRepository extends JpaRepository<Faction, Integer> {

Optional<Faction> findById(Integer id);

Optional<Faction> findByName(String name);

@Transactional

void deleteByName(String name);

}

Листинг 5 – Код класса FactionService

@Service

public class FactionServiceImpl implements FactionService {

@Autowired

private FactionRepository factionRepository;

Окончание листинга 5

@Override

public Faction findById(Integer id) {

return this.factionRepository.findById(id).orElse(null);

}

@Override

public Faction findByName(String name) {

return this.factionRepository.findByName(name).orElse(null);

}

@Override

public void deleteByName(String name) {

this.factionRepository.deleteByName(name);

}

@Override

@Transactional

public Integer add(Faction faction) {

this.factionRepository.save(faction);

return faction.getId();

}

}

Листинг 6 – Часть кода класса Program

public class Program {

public static void main(String[] args) {

var context = new AnnotationConfigApplicationContext(SpringConfiguration.class);

var factionService = context.getBean("factionServiceImpl", FactionService.class);

var newCommander = new Commander();

newCommander.setName("Player created with ORM");

newCommander.setFaction(factionService.findByName("New player faction"));

1. Вывод

В ходе работы была настроена связь между базой данных и  
Java-приложением с использованием ORM-прослойки Hibernate, написана реализация некоторых запросов и получены соответствующие навыки.