# SpringData整合JPA

1. 案例
   1. pom.xml 配置

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.1.RELEASE</version>  </parent>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid-spring-boot-starter</artifactId>  <version>1.1.9</version>  </dependency>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <optional>true</optional>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <scope>provided</scope>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <configuration>  <source>1.8</source>  <target>1.8</target>  </configuration>  </plugin>  </plugins>  </build> |

* 1. application.properties 配置

|  |
| --- |
| spring.datasource.url=jdbc:mysql://127.0.0.1:3306/ssm?useSSL=false  spring.datasource.username=root  spring.datasource.password=123456  spring.datasource.driver-class-name=com.mysql.jdbc.Driver  spring.jpa.properties.hibernate.hbm2ddl.auto=update  spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect  spring.jpa.show-sql= true |

* 1. 代码

|  |
| --- |
| @Entity  @Data  @NoArgsConstructor  @AllArgsConstructor  @ToString  @Table(name="user2")  public class User {  @Id  @GeneratedValue(strategy=GenerationType.IDENTITY)//主键生成策略  @Column(name="id")  private Integer id;  @Column(name="name")  private String name;  @Column(name="password")  private String password;  } |
| /\*\*  \*  \* @author fliay  \* 参数T: 当前需要映射的实体  \* 参数ID:当前映射实体中的OID类型  \*/  public interface UserRepository extends JpaRepository<User, Integer> {  } |
| @SpringBootApplication  public class MainClass {  public static void main(String[] args) {  SpringApplication.run(MainClass.class, args);  }  } |
| @RunWith(SpringJUnit4ClassRunner.class)  @SpringBootTest(classes=MainClass.class)  public class UserRepositoryTest {  @Autowired  private UserRepository userRepository;  @Test  public void testSave(){  User u = new User();  u.setName("张三");  u.setPassword("1231234");  this.userRepository.save(u);  }  } |

1. 核心接口

2.1 Repository 接口

1）提供了方法名称命名查询方式

|  |
| --- |
| /\*\*  \*  \* @author fliay  \* Repository 接口的方法名称命名查询  \*  \*  \*  \*/  public interface UserRepositoryByUserName extends Repository<User, Integer> {  //方法名必须要遵循驼峰式命名规则，findByXXX XXX是属性名称 首字母要大写+查询条件（首字母大写）  List<User> findByName(String name);  //通过两个条件进行查询，必须都要满足格式要求  User findByNameAndPassword(String name,String password);  //模糊查询  List<User> findByNameLike(String name);  } |
| @Test  public void testGetName() {  List<User> u = this.userRepositoryByUserName.findByName("张三");  u.forEach((user) -> {  System.out.println(user);  });  }  @Test  public void testFineByNameAndPassword() {  User user = this.userRepositoryByUserName.findByNameAndPassword("张三", "234");  System.out.println(user);  }    @Test  public void testFineByNameLike() {  List<User> users = this.userRepositoryByUserName.findByNameLike("%张%");  users.forEach((user)->{  System.out.println(user);  });  } |

2）提供了基于@Query注解查询与更新

2.2 CrudRepository 接口

2.3 PagingAndSortingRepository接口

2.4 JpaRepository 接口

2.5 JPASpecificationExecutor 接口

1. 2