



传智播客.黑马程序员

## 第3章 Spring Boot数据访问



Spring  
Boot

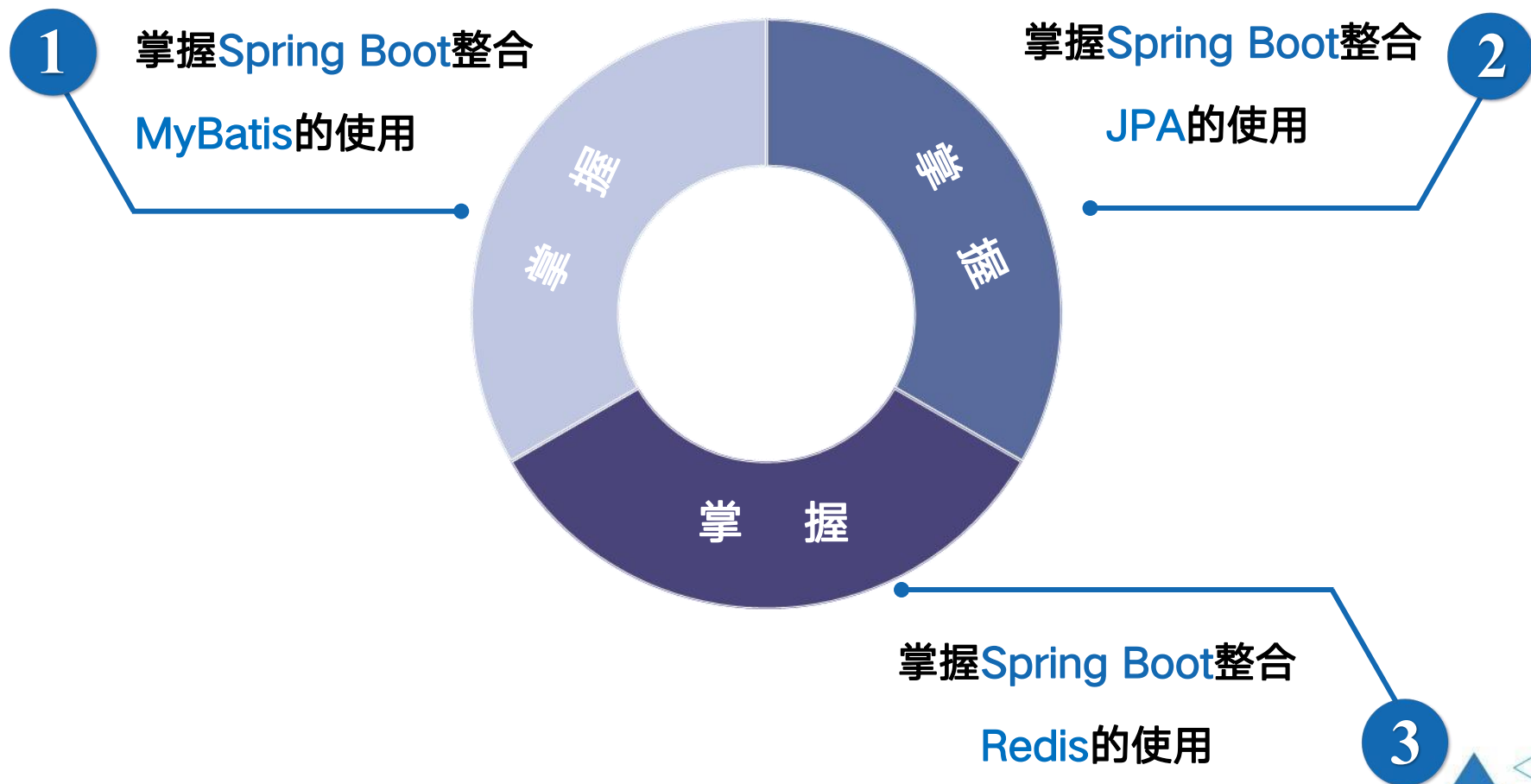
- Spring Boot整合MyBatis的使用
- Spring Boot整合JPA的使用
- Spring Boot整合Redis的使用



# 学习目标



传智播客.黑马程序员  
改变中国IT教育 我们正在行动





# 目录

传智播客·黑马程序员  
改变中国IT教育 我们正在行动

3.1

## Spring Boot 数据访问概述

3.2

## Spring Boot 整合 MyBatis

 [点击查看本案例相关知识点](#)

3.3

## Spring Boot 整合JPA

 [点击查看本案例相关知识点](#)

3.4

## Spring Boot 整合 Redis

 [点击查看本案例相关知识点](#)



返回目录

## 3.2 Spring Boot 整合MyBatis

1

● 基础环境搭建

2

● 使用注解的方式整合 MyBatis

3

● 使用配置文件的方式整合 MyBatis



返回目录

## 3.3 Spring Boot 整合JPA

1

● Spring Data JPA介绍

2

● 使用Spring Boot整合JPA



返回目录

## 3.4 Spring Boot 整合Redis

1

● Redis 介绍

2

● 使用Spring Boot 整合Redis



# 章节概要

传智播客·黑马程序员  
改变中国IT教育 我们正在行动

在开发中，通常会涉及到对数据库的数据进行操作，Spring Boot在简化项目开发以及实现自动化配置的基础上，对[关系型数据库](#)和[非关系型数据库](#)的访问操作都提供了非常好的整合支持。



本章，将对Spring Boot的[数据访问](#)进行介绍，并对常用的数据操作框架进行整合讲解。



## 3.1 SpringBoot数据访问

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### Spring Boot 数据访问概述



#### 概述

Spring Boot默认采用整合SpringData的方式统一处理数据访问层，通过添加大量自动配置，引入各种数据访问模板xxxTemplate以及统一的Repository接口，从而达到简化数据访问层的操作。





## 3.1 SpringBoot数据访问



### Spring Boot 数据访问概述



### Spring Boot提供的常见数据库依赖启动器

名称	对应数据库
spring-boot-starter-data-jpa	<ul style="list-style-type: none"><li>• Spring Data JPA</li><li>• Hibernate</li></ul>
spring-boot-starter-data-mongodb	<ul style="list-style-type: none"><li>• MongoDB</li><li>• Spring Data MongoDB</li></ul>
spring-boot-starter-data-neo4j	<ul style="list-style-type: none"><li>• Neo4j图数据库</li><li>• Spring Data Neo4j</li></ul>
spring-boot-starter-data-redis	<ul style="list-style-type: none"><li>• Redis</li></ul>



## 3.2 Spring Boot 整合MyBatis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### 基础环境搭建



#### 搭建步骤：

1. **数据准备：** 创建数据库、数据表并插入一定的数据
2. **创建项目，引入相应的启动器：** 使用Spring Initializr的方式构建项目，选择MySQL和MyBatis依赖,编写实体类。
3. **编写配置文件：** 在配置文件中进行数据库连接配置以及进行第三方数据源的默认参数覆盖。



## 3.2 Spring Boot 整合MyBatis

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### 基础环境搭建

#### ① 创建数据库

```
CREATE DATABASE springbootdata;
```



### 基础环境搭建

#### ② 创建数据表t\_article

```
CREATE TABLE `t_article` (  
  `id` int(20) NOT NULL AUTO_INCREMENT COMMENT '文章id',  
  `title` varchar(200) DEFAULT NULL COMMENT '文章标题',  
  `content` longtext COMMENT '文章内容',  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;
```



### 基础环境搭建

#### ③ 向数据表t\_article插入相关数据

```
INSERT INTO `t_article` VALUES ('1', 'Spring Boot基础入门',  
从入门到精通讲解...');
```

```
INSERT INTO `t_article` VALUES ('2', 'Spring Cloud基础入门',  
'从入门到精通讲解...');
```



### 基础环境搭建

#### ④ 创建数据表t\_comment

```
CREATE TABLE `t_comment` (  
  `id` int(20) NOT NULL AUTO_INCREMENT COMMENT '评论id',  
  `content` longtext COMMENT '评论内容',  
  `author` varchar(200) DEFAULT NULL COMMENT '评论作者',  
  `a_id` int(20) DEFAULT NULL COMMENT '关联的文章id',  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```



### 基础环境搭建

#### ⑤ 向数据表t\_comment插入相关数据

```
INSERT INTO `t_comment` VALUES ('1','很全、很详细','狂奔的  
蜗牛','1');
```

```
INSERT INTO `t_comment` VALUES ('2','赞一个','tom','1');
```

```
INSERT INTO `t_comment` VALUES ('3','很详细','kitty','1');
```

```
INSERT INTO `t_comment` VALUES ('4','很好，非常详细','张三  
,','1');
```

```
INSERT INTO `t_comment` VALUES ('5','很不错','张杨','2');
```



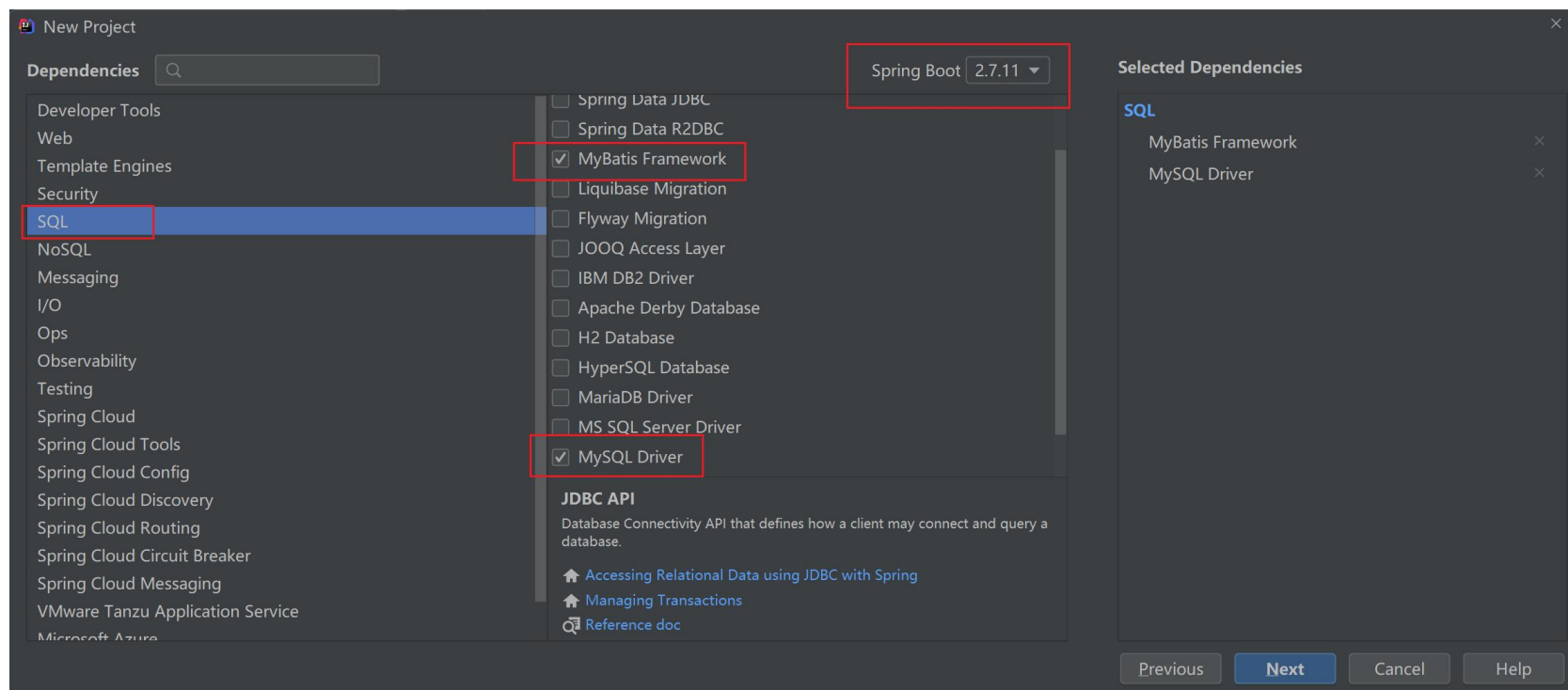
## 3.2 Spring Boot 整合MyBatis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### 基础环境搭建

#### ⑥ 创建项目，引入MySQL和MyBatis的依赖启动器







### 基础环境搭建

⑥ 创建项目，引入MySQL和MyBatis的依赖启动器

⑦ 编写实体类Comment和Article

#### Comment

```
public class Comment {  
    private Integer id;  
    private String content;  
    private String author;  
    private Integer aId;}
```

#### Article

```
public class Article {  
    private Integer id;  
    private String title;  
    private String content;  
    private List<Comment> commentList;}
```



### 基础环境搭建

#### ⑧ 编写配置文件

1、在全局配置文件中进行数据库连接配置

```
spring.datasource.driver-class-name=com.mysql.jdbc.Driver  
spring.datasource.url=jdbc:mysql://localhost:3306/springbootdata?server  
Timezone=UTC  
spring.datasource.username=root  
spring.datasource.password=root
```



### 基础环境搭建

#### ⑧ 编写配置文件

2、设置数据源类型配置（以阿里巴巴的Druid数据源为例）

```
<dependency>  
  <groupId>com.alibaba</groupId>  
  <artifactId>druid-spring-boot-starter</artifactId>  
  <version>1.1.10</version>  
</dependency>
```



### 基础环境搭建

#### ⑨ 在全局配置文件中设置属性

如果在开发过程中，需要对这些第三方Druid的运行参数进行重新设置，必须在application.properties配置文件中默认参数覆盖。

#数据源类型

**spring.datasource.type = com.alibaba.druid.pool.DruidDataSource**

#初始化连接数

**spring.datasource.initialSize=20**

#最小空闲数

**spring.datasource.minIdle=10**

#最大连接数

**spring.datasource.maxActive=100**



## 3.2 Spring Boot 整合MyBatis

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### 使用注解方式整合MyBatis



#### 整合步骤：

1. 创建CommentMapper接口文件：@Mapper  
  
//表明该类是mybaits接口文件，是需要被  
springboot扫描的
2. 编写测试方法进行接口方法测试及整合测试



## 3.2 Spring Boot 整合MyBatis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### 使用注解方式整合MyBatis

#### ① 创建CommentMapper接口文件

```
@Mapper
public interface CommentMapper {
    @Select("SELECT * FROM t_comment WHERE id =#{id}")
    public Comment findById(Integer id);

    @Insert("INSERT INTO t_comment(content,author,a_id) " +
            "values (#{content},#{author},#{aId})")
    public int insertComment(Comment comment);

    @Update("UPDATE t_comment SET content=#{content} WHERE id=#{id}")
    public int updateComment(Comment comment);

    @Delete("DELETE FROM t_comment WHERE id=#{id}")
    public int deleteComment(Integer id);
}
```



### 使用注解方式整合MyBatis

#### ① 创建CommentMapper接口文件

查询数据操作：

```
@Select("SELECT * FROM t_comment WHERE id =#{id}")
```

```
public Comment findById(Integer id);
```

插入数据操作：

```
@Insert( "INSERT INTO t_comment values (#{content},#{author},#{aId})")
```

```
public int insertComment(Comment comment);
```



### 使用注解方式整合MyBatis

#### ① 创建Mapper接口文件

更新数据操作：

```
@Update("UPDATE t_comment SET content=#{content} WHERE id=#{id}")  
public int updateComment(Comment comment);
```

删除数据操作：

```
@Delete("DELETE FROM t_comment WHERE id=#{id}")  
public int deleteComment(Integer id);}
```





### 使用注解方式整合MyBatis

#### ② 编写测试方法进行接口方法测试以及整合测试

**@RunWith(SpringRunner.class)**

**@SpringBootTest**

**public class Chapter03ApplicationTests {**

**@Autowired**

**private CommentMapper commentMapper;**

**@Test**

**public void selectComment() {**

**Comment comment = commentMapper.findById(1);**

**System.out.println(comment);}}**



## 3.2 Spring Boot 整合MyBatis

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### 使用注解方式整合MyBatis

#### 结果

```
Comment{id=1, content='很全、很详细', author='狂奔的蜗牛', aId=null}
```

```
2023-04-26 21:16:38.940 INFO 8400 --- [ionShutdownHook] com.alibaba.druid.pool.Dr
```

```
Process finished with exit code 0
```

发现**aId**没有正确读取，原因是**Comment**类中**aId**命名采用驼峰命名法，和数据库并不匹配



## 3.2 Spring Boot 整合MyBatis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### 使用注解方式整合MyBatis

#### 全局配置文件中打开驼峰命名规则进行匹配



```
Comment{id=1, content='很全、很详细', author='狂奔的蜗牛', aId=1}
```

```
2023-04-26 21:20:01.118 INFO 11904 --- [ionShutdownHook] com.alibaba.druid.pool.Dru
```

```
Process finished with exit code 0
```



## 3.2 Spring Boot 整合 MyBatis

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### 使用配置文件方式整合 MyBatis



#### 整合步骤：

1. 创建Mapper接口文件：@Mapper
2. 创建XML映射文件：编写对应的SQL语句
3. 在全局文件中配置XML映射文件路径以及实体类别名映射路径
4. 编写测试方法进行接口方法测试及整合测试



## 3.2 Spring Boot 整合 MyBatis

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### 使用配置文件方式整合 MyBatis

#### ① 创建Mapper接口文件：

**@Mapper**

```
public interface ArticleMapper {  
    public Article selectArticle(Integer id);  
    public int updateArticle(Article article);  
}
```



## 3.2 Spring Boot 整合 MyBatis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### 使用配置文件方式整合 MyBatis

#### ② 创建XML映射文件ArticleMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE mapper
```

```
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
```

```
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
```

```
<mapper namespace="com.example.mapper.ArticleMapper">
```

需要映射的  
接口



### 使用配置文件方式整合 MyBatis

#### ② 创建XML映射文件ArticleMapper.xml

查询操作：

起别名以便  
和article id  
区分

```
<select id="selectArticle" resultMap="articleWithComment">  
    SELECT a.*,c.id c_id,c.content c_content,c.author  
    FROM t_article a,t_comment c  
    WHERE a.id=c.a_id AND a.id = #{id}  
</select>
```



### 使用配置文件方式整合 MyBatis

#### ② 创建XML映射文件ArticleMapper.xml

更新操作：

```
<update id="updateArticle" parameterType="Article" >  
    UPDATE t_article  
    <set>  
        <if test="title !=null and title !=''">  
            title=#{title},  
        </if>  
        <if test="content !=null and content !=''">  
            content=#{content}  
        </if>  
    </set>  
    WHERE id=#{id}  
</update>
```





### 使用配置文件方式整合 MyBatis

#### ② 创建XML映射文件ArticleMapper.xml

结果集：

```
<resultMap id="articleWithComment" type="Article">  
  <id property="id" column="id" />  
  <result property="title" column="title" />  
  <result property="content" column="content" />  
  <collection property="commentList" ofType="Comment">  
    <id property="id" column="c_id" />  
    <result property="content" column="c_content" />  
    <result property="author" column="author" />  
  </collection>  
</resultMap>
```

因为起了别名，所以这里要定义为别名



## 3.2 Spring Boot 整合 MyBatis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



使用型

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.itheima.mapper.ArticleMapper">
    <!-- 1、查询文章详细（包括评论信息） -->
    <select id="selectArticle" resultMap="articleWithComment">
        SELECT a.#{c.id} c_id,c.content c_content,c.author
        FROM t_article a,t_comment c
        WHERE a.id=c.a_id AND a.id = #{id}
    </select>
    <resultMap id="articleWithComment" type="Article">
        <id property="id" column="id" />
        <result property="title" column="title" />
        <result property="content" column="content" />
        <collection property="commentList" ofType="Comment">
            <id property="id" column="c_id" />
            <result property="content" column="c_content" />
            <result property="author" column="author" />
        </collection>
    </resultMap>
    <!-- 2、根据文章id更新文章信息 -->
    <update id="updateArticle" parameterType="Article">
        UPDATE t_article
        <set>
            <if test="title != null and title != ''">
                title=#{title},
            </if>
            <if test="content != null and content != ''">
                content=#{content}
            </if>
        </set>
        WHERE id=#{id}
    </update>
</mapper>
```



## 3.2 Spring Boot 整合 MyBatis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### 使用配置文件方式整合 MyBatis

#### ③ 配置XML映射文件路径

#配置MyBatis的xml配置文件路径

**mybatis.mapper-locations=classpath:mapper/\*.xml**

#配置XML映射文件中指定的实体类别名路径

**mybatis.type-aliases-package=com.example.domain**



### 使用配置文件方式整合 MyBatis

#### ④ 编写单元测试对接口进行测试

**@Autowired**

```
private ArticleMapper articleMapper;
```

**@Test**

```
public void selectArticle() {
```

```
    Article article = articleMapper.selectArticle(1);
```

```
    System.out.println(article);
```

```
    Article article = new Article();
```

```
    article.setId(1);
```

```
    article.setContent("入门! ");
```

```
    articleMapper.updateArticle(article);
```

```
    Article articles = articleMapper.selectArticle(1);
```

```
    System.out.println(articles);
```

```
}
```



## 3.3 Spring Boot 整合JPA

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍



#### Spring Data JPA简介

Spring Data JPA是Spring基于ORM框架、JPA规范的基础上封装的一套JPA应用框架，它提供了增删改查等常用功能，使开发者可以用较少的代码实现数据操作，同时还易于扩展。



## 3.3 Spring Boot 整合JPA

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍



### Spring Data JPA基本使用

1. 在pom文件中添加Spring Data JPA依赖启动器
2. 编写ORM实体类：实体类与数据表进行映射，并且配置好映射关系。
3. 编写Repository接口：针对不同的表数据操作编写各自对应的Repository接口，并根据需要编写对应的数据操作方法。
4. 编写单元测试进行接口方法测试及整合测试



## 3.3 Spring Boot 整合JPA

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍



#### POM文件添加依赖

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-data-jpa</artifactId>
```

```
</dependency>
```



## 3.3 Spring Boot 整合JPA

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍

#### ① 编写ORM实体类

```
@Entity(name = "t_comment")  
public class Discuss {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer id;  
    @Column(name = "a_id")  
    private Integer aId;  
    // 省略getXX()和setXX()方法  
}
```





## 3.3 Spring Boot 整合JPA

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍

#### ① 编写ORM实体类

```
@Entity(name = "t_comment")
public class Discuss {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "content")
    private String content;
    @Column(name = "author")
    private String author;
    @Column(name = "a_id")
    private Integer aId;
}
```



## 3.3 Spring Boot 整合JPA

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍

② 编写Repository接口DiscussRepository继承

JpaRepository<T,ID> //T为Entity类,ID为该类主键的类型

```
public interface DiscussRepository extends JpaRepository<Discuss,Integer> {  
}
```



### Spring Data JPA 介绍

#### ② 编写Repository接口DiscussRepository

查询author非空的Discuss评论信息，命名规则是findBy+字段+条件

```
public List<Discuss> findByAuthorNotNull();
```

通过文章id分页查询出Discuss评论信息。

```
@Query("SELECT c FROM t_comment c WHERE c.aId = ?1")  
public List<Discuss> getDiscussPaged(Integer aid,Pageable pageable);
```



### Spring Data JPA 介绍

#### ② 编写Repository接口DiscussRepository

通过文章id分页查询出Discuss评论信息。

```
@Query(value = "SELECT * FROM t_comment WHERE a_Id =  
?1",nativeQuery = true)  
public List<Discuss> getDiscussPaged2(Integer aid,Pageable pageable);
```

注：

- 与getDiscussPaged()方法的参数和作用完全一样。
- 区别是该方法上方的@Query注解将nativeQuery属性设置为了true，用来编写原生SQL语句。



## 3.3 Spring Boot 整合JPA

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍

#### ② 编写Repository接口DiscussRepository

对数据进行更新和删除操作

```
@Transactional
```

```
@Modifying
```

```
@Query("UPDATE t_comment c SET c.author = ?1 WHERE c.id = ?2")
```

```
public int updateDiscuss(String author,Integer id);
```

```
@Transactional
```

```
@Modifying
```

```
@Query("DELETE t_comment c WHERE c.id = ?1")
```

```
public int deleteDiscuss(Integer id);
```



## 3.3 Spring Boot 整合JPA

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍

## 3. 测试

```
import org.springframework.data.domain.Pageable;
import java.util.List;
import java.util.Optional;

@SpringBootTest
class JpaTests {

    @Autowired
    private DiscussRepository discussRepository;

    @Test
    public void selectComment(){
        List<Discuss> list = discussRepository.findByAuthorNotNull();
        for (Discuss discuss : list) {
            System.out.println(discuss);
        }
        System.out.println();

        Optional<Discuss> optional = discussRepository.findById(1);
        System.out.println(optional.get());
    }

    @Test
    public void selectCommentPaged(){
        Pageable pageable = PageRequest.of(0, 3);
        List<Discuss> discussPaged = discussRepository.getDiscussPaged(1, pageable);
        System.out.println(discussPaged);
    }

    @Test
    public void updateComment(){
        int i = discussRepository.updateDiscuss("疯狂的拖拉机", 1);
        System.out.println(i);
    }
}
```



## 3.3 Spring Boot 整合JPA

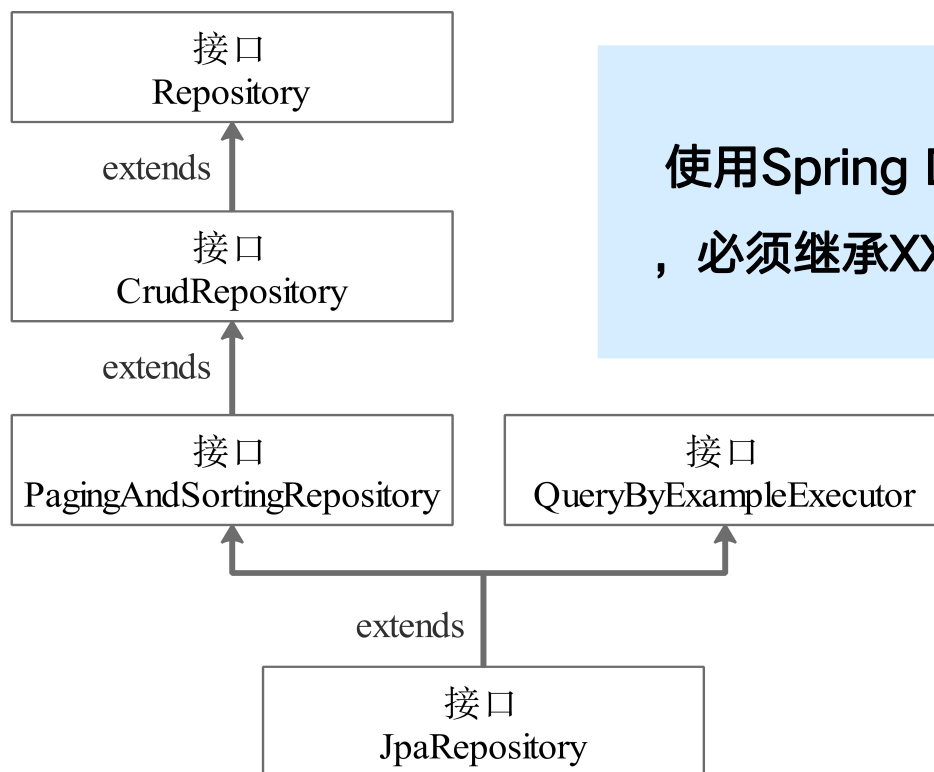
传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍



#### Repository继承关系



使用Spring Data JPA自定义Repository接口，必须继承XXRepository<T, ID>接口。



## 3.3 Spring Boot 整合JPA

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍



#### 使用Spring Data JPA进行数据操作的多种实现方式



如果自定义接口继承了JpaRepository接口，则默认包含了一些常用的CRUD方法。



自定义Repository接口中，可以使用@Query注解配合SQL语句进行数据的查、改、删操作。



自定义Repository接口中，可以直接使用方法名关键字进行查询操作。





## 3.3 Spring Boot 整合JPA

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍



#### 自定义Repository接口中的@Transactional注解

- 在自定义的Repository接口中，针对数据的变更操作（修改、删除），无论是否使用了@Query注解，都必须在方法上方添加@Transactional注解进行事务管理，否则程序执行就会出现InvalidDataAccessApiUsageException异常。
- 如果在调用Repository接口方法的业务层Service类上已经添加了@Transactional注解进行事务管理，那么Repository接口文件中就可以省略@Transactional注解。



## 3.3 Spring Boot 整合JPA

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍



#### 变更操作，要配合使用@Query与Modify注解

- 在自定义的Repository接口中，使用@Query注解方式执行数据变更操作（修改、删除），除了要使用@Query注解，还必须添加@Modifying注解表示数据变更。



## 3.3 Spring Boot 整合JPA

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### Spring Data JPA 介绍



#### 变更操作，要配合使用@Query与Modify注解

- 在自定义的Repository接口中，使用@Query注解方式执行数据变更操作（修改、删除），除了要使用@Query注解，还必须添加@Modifying注解表示数据变更。



#### JPA还支持使用Example实例进行复杂条件查询



## 3.4 Spring Boot 整合Redis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### Redis 介绍



#### Redis简介

Redis 是一个开源（BSD许可）的、内存中的数据结构存储系统，它可以用作数据库、缓存和消息中间件，并提供多种语言的API。



### Redis 介绍



### Redis优点

1. **存取速度快**：Redis速度非常快，每秒可执行大约110000次的设值操作，或者执行81000次的读取操作。
2. **支持丰富的数据类型**：Redis支持开发人员常用的大多数数据类型，例如列表、集合、排序集和散列等。
3. **操作具有原子性**：所有Redis操作都是原子操作，这确保如果两个客户端并发访问，Redis服务器能接收更新后的值。
4. **提供多种功能**：Redis提供了多种功能特性，可用作非关系型数据库、缓存中间件、消息中间件等。



## 3.4 Spring Boot 整合Redis








传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### Redis 介绍



### 安装Redis相关软件

 redis-benchmark.exe	2016/7/1 15:55	应用程序	397 KB
 redis-benchmark.pdb	2016/7/1 15:55	Program Debug Da...	4,268 KB
 redis-check-aof.exe	2016/7/1 15:55	应用程序	251 KB
 redis-check-aof.pdb	2016/7/1 15:55	Program Debug Da...	3,436 KB
 redis-check-dump.exe	2016/7/1 15:55	应用程序	262 KB
 redis-check-dump.pdb	2016/7/1 15:55	Program Debug Da...	3,404 KB
 redis-cli.exe 客户端	2016/7/1 15:55	应用程序	471 KB
 redis-cli.pdb	2016/7/1 15:55	Program Debug Da...	4,412 KB
 redis-server.exe 服务器端	2016/7/1 15:55	应用程序	1,517 KB
 redis-server.pdb	创建日期: 2023/5/2 20:46 大小: 1.48 MB	Program Debug Da...	6,748 KB
 Windows Service Documentation.docx	2016/7/1 9:17	Microsoft Word 文档	14 KB



## 3.4 Spring Boot 整合Redis

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### Redis 介绍



### 安装Redis相关软件

双击服务器端打开redis 服务器

```
F:\迅雷下载\Redis-x64-3.0 x + v
[24556] 02 May 21:10:11.759 # Warning: no config file specified, using the default config. In order to specify a config
file use F:\迅雷下载\Redis-x64-3.0.504\redis-server.exe /path/to/redis.conf

      _ _ _
     / _ _\
    / _ _ \
   / _ _ \
  / _ _ \
 / _ _ \
/_ _ _ \

Redis 3.0.504 (00000000/0) 64 bit

Running in standalone mode
Port: 6379
PID: 24556

http://redis.io

[24556] 02 May 21:10:11.762 # Server started, Redis version 3.0.504
[24556] 02 May 21:10:11.762 * The server is now ready to accept connections on port 6379
```



## 3.4 Spring Boot 整合Redis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动

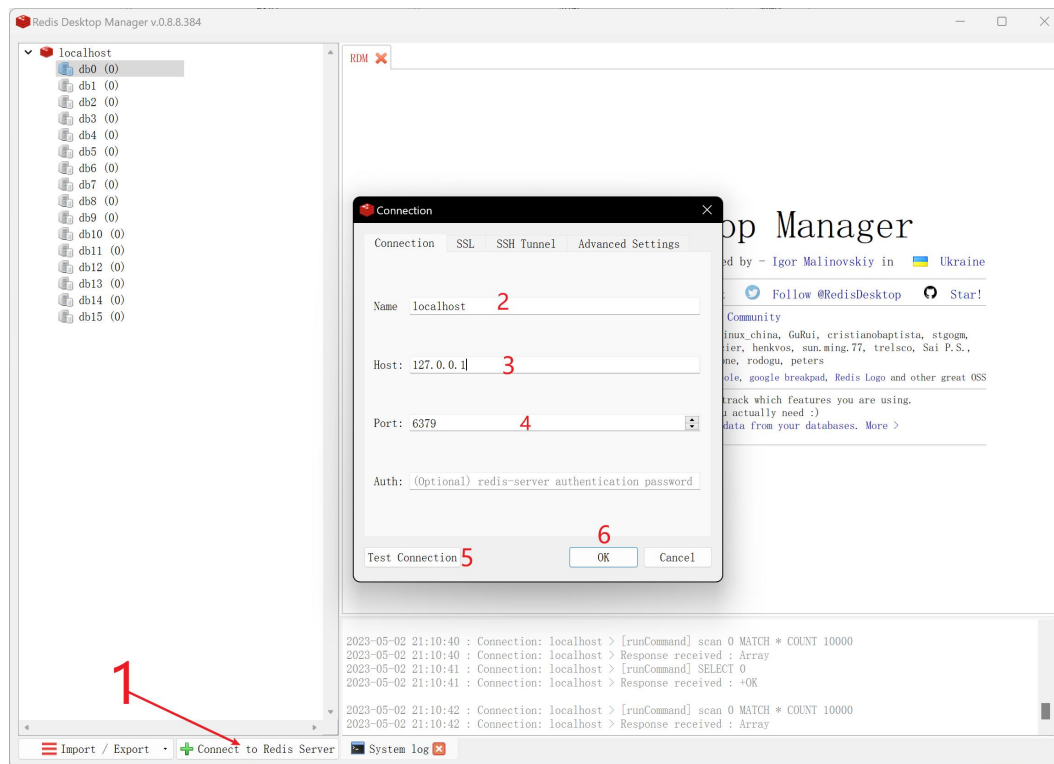


### Redis 介绍



### 安装Redis相关软件

安装redis desktop  
按照上图步骤创建数据库







## 3.4 Spring Boot 整合Redis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### 使用Spring Boot 整合 Redis



#### 整合步骤：

1. 在pom文件中添加Spring Data Redis依赖启动器
2. 编写实体类
3. 编写Repository接口
4. 在全局配置文件application.properties中添加Redis数据库连接配置
5. 编写单元测试进行接口方法测试以及整合测试



## 3.4 Spring Boot 整合Redis

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### 使用Spring Boot 整合 Redis

#### ① 在pom文件中添加Spring Data Redis依赖启动器

```
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-data-redis</artifactId>  
</dependency>
```



## 3.4 Spring Boot 整合Redis

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### 使用Spring Boot 整合 Redis

#### ② 编写实体类Person

**@RedisHash("persons")**

**public class Person {**

**@Id** //用于标识主键 import org.springframework.data.annotation.Id;

**private String id;**

**@Indexed** //用于标识改属性会在redis中生成二级索引（可能根据该属性进行查询）

**private String firstname;**

**@Indexed**

**private String lastname;**

**private Address address;**

**private List<Family> familyList;}**



## 3.4 Spring Boot 整合Redis

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### 使用Spring Boot 整合 Redis

#### ② 编写实体类Address

```
public class Address {  
    @Indexed  
    private String city;  
    @Indexed  
    private String country;  
}
```



## 3.4 Spring Boot 整合Redis

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### 使用Spring Boot 整合 Redis

#### ② 编写实体类Family

```
public class Family {  
    @Indexed  
    private String type;  
    @Indexed  
    private String username;}  
}
```



## 3.4 Spring Boot 整合Redis

传智播客.黑马程序员  
改变中国IT教育 我们正在行动



### 使用Spring Boot 整合 Redis

#### ③ 编写Repository接口PersonRepository

```
public interface PersonRepository extends CrudRepository<Person, String> {  
    List<Person> findByLastname(String lastname);  
    Page<Person> findPersonByLastname(String lastname, Pageable page);  
    List<Person> findByFirstnameAndLastname(String firstname, String lastname);  
    List<Person> findByAddress_City(String city);  
    List<Person> findByFamilyList_Username(String username);  
}
```



## 3.4 Spring Boot 整合Redis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### 使用Spring Boot 整合 Redis

#### ④ Redis数据库连接配置

**spring.redis.host=127.0.0.1**

**spring.redis.port=6379**

**spring.redis.password=**



## 3.4 Spring Boot 整合Redis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### 使用Spring Boot 整合 Redis

#### ⑤ 编写测试

#### 方法整合测试

```
@Autowired
PersonRepository personRepository;

private Person createPerson(String id, String firstname, String lastname, String city, String country, String type1,
    String username1, String type2, String username2){
    Person person = new Person();
    person.setFirstname(firstname);
    person.setLastname(lastname);
    person.setId(id);
    Address address = new Address();
    address.setCity(city);
    address.setCountry(country);
    person.setAddress(address);
    Family family1 = new Family();
    family1.setType(type1);
    family1.setUsername(username1);
    Family family2 = new Family();
    family2.setType(type2);
    family2.setUsername(username2);
    ArrayList<Family> families = new ArrayList<>();
    families.add(family1);
    families.add(family2);
    person.setFamilyList(families);
    return person;
}

@Test
public void test(){
    Person person1 = createPerson("1","张","三","温州","中国","父亲",
        "张四","母亲","小花");
    Person person2 = createPerson("2","王","五","杭州","中国","父亲",
        "老王","母亲","小菲");

    personRepository.save(person1);
    personRepository.save(person2);
}
```





## 3.4 Spring Boot 整合Redis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### 使用Spring Boot 整合 Redis

#### ⑤ 编写测试方法整合测试

The screenshot shows the Redis Desktop Manager interface. On the left, the database structure is displayed as a tree view under 'localhost' > 'db0' > 'persons'. The 'persons' key is selected, showing its sub-keys: 'persons:1', 'persons:2', 'address.city', 'address.country', 'familyList.type', 'familyList.username', 'firstname', and 'lastname'. The 'persons:1' key is expanded, showing its values: 'persons:address.city:杭州', 'persons:address.city:温州', 'persons:address.country:中国', 'persons:familyList.type:母亲', 'persons:familyList.username:小花', 'persons:familyList.username:小菲', 'persons:firstname:张', 'persons:firstname:王', 'persons:lastname:三', and 'persons:lastname:五'.

The main panel shows the details for the 'persons:1' key. The key type is 'HASH'. The key size is 8 bytes, and the TTL is -1. The key value is displayed as a table:

row	key	value
1	_class	com.example.chapter03.domain.Person
2	address.city	温州
3	address.country	中国
4	familyList.[0].type	母亲
5	familyList.[0].username	小花
6	firstname	张
7	id	1
8	lastname	三

The 'Value' field is empty, and the 'View as' dropdown is set to 'Plain Text'. The 'Save' button is visible at the bottom right.

The bottom status bar shows the following log messages:

```
2023-05-02 22:07:10 : Connection: localhost > Response received : +hash
2023-05-02 22:07:10 : Connection: localhost > [runCommand] ttl persons:1
2023-05-02 22:07:10 : Connection: localhost > Response received :
2023-05-02 22:07:10 : Connection: localhost > [runCommand] HLEN persons:1
2023-05-02 22:07:10 : Connection: localhost > Response received :
2023-05-02 22:07:10 : Connection: localhost > [runCommand] HSCAN persons:1 0 COUNT 10000
2023-05-02 22:07:10 : Connection: localhost > Response received : Array
```



## 3.4 Spring Boot 整合Redis

传智播客·黑马程序员  
改变中国IT教育 我们正在行动



### 使用Spring Boot 整合 Redis

#### ⑤ 编写测试方法整合测试

```
@Test
public void testFind(){
    List<Person> personList = personRepository.findByLastname("三");
    for (Person person : personList) {
        System.out.println(person);
    }
}
```

```
2023-05-02 22:08:35.797 INFO 26120 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform
2023-05-02 22:08:35.804 INFO 26120 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2023-05-02 22:08:36.428 INFO 26120 --- [main] com.example.chapter03.RedisTests : Started RedisTests in 2.215 seconds (JVM running for 2.638)
Person{id='1', firstname='张', lastname='三', address=Address{city='温州', country='中国'}, familyList=[Family{type='母亲', username='小花'}]}
2023-05-02 22:08:36.855 INFO 26120 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
2023-05-02 22:08:36.858 INFO 26120 --- [ionShutdownHook] com.alibaba.druid.pool.DruidDataSource : {dataSource-1} closed
```



# Thank You!

yx.boxuegu.com

