JDBC Template

使用Spring组件JDBC Template简化持久化操作

课程结构

- 课程介绍
- JDBC Template概念
- 环境配置
- 基本操作
- 优缺点分析
- 课程总结

课程介绍

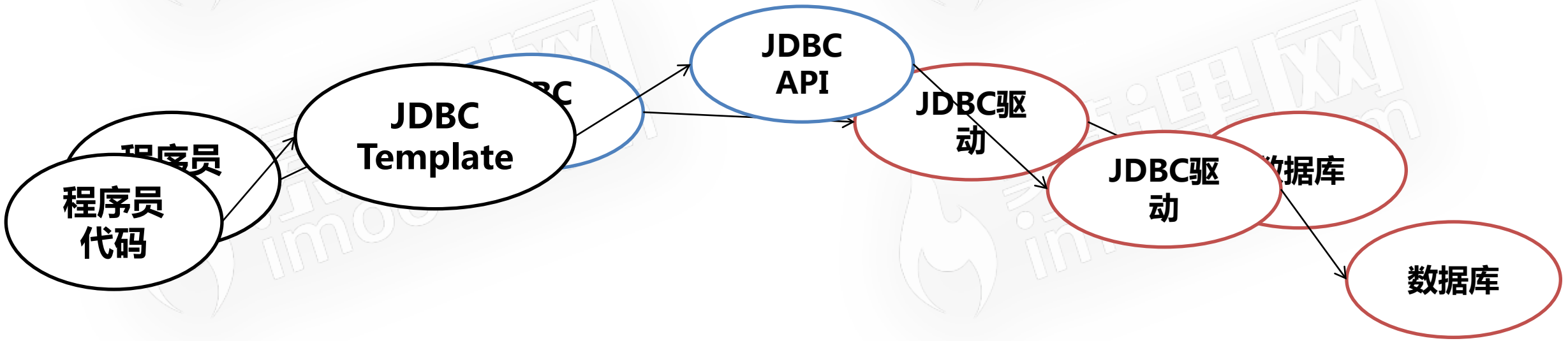
- 课程目标
  - 了解Spring组件JDBC Template
  - 能使用JDBC Template进行持久化操作
  - 帮助自己学习Hibernate、MyBatis等ORM框架
- 前置条件
  - JDBC
  - Spring IOC、Apring AOP
  - Mysql

课程环境

- 操作系统
  - Win7
- JDK
  - jdk1.8.0
- 数据库
  - Mysql-5.7
- 开发工具
  - IntelliJ IDEA

Spring JDBC Template

- 为了简化持久化操作，Spring在JDBC API之上提供了JDBC Template组件

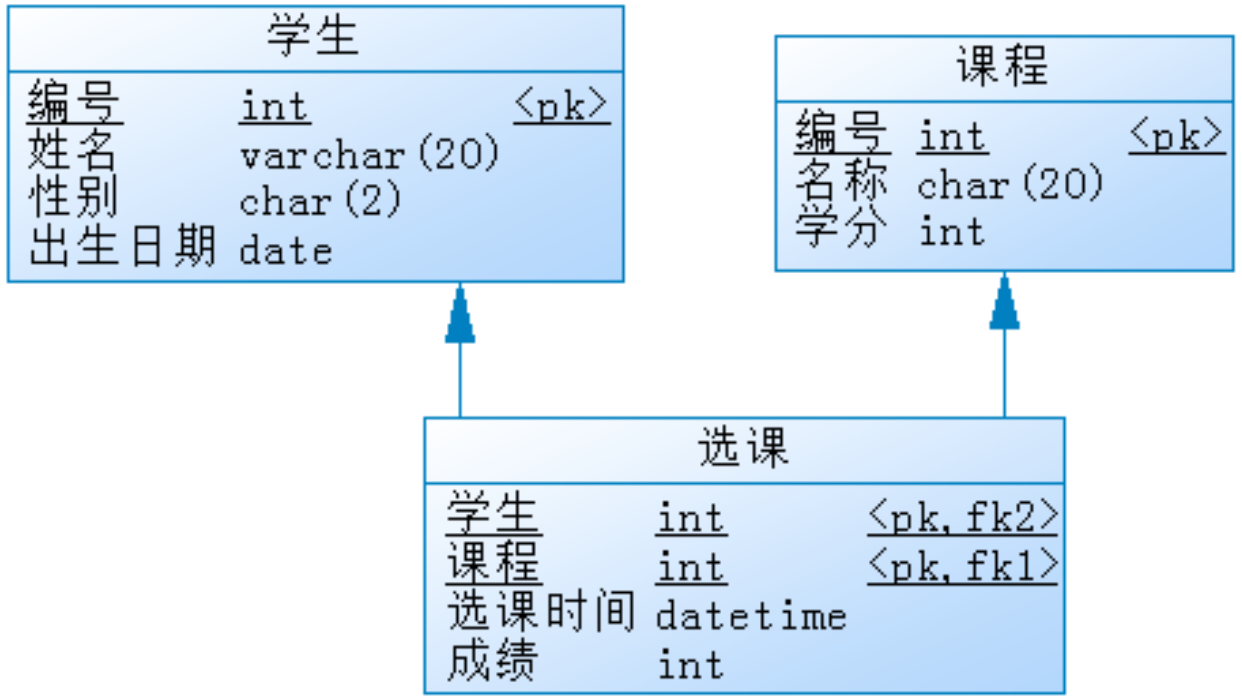


Spring JDBC Template

- **JDBC Template提供统一的模板方法，在保留代码灵活性的基础上，尽量减少持久化代码**

```
//JDBC API
Statement statement = conn.createStatement();
ResultSet resultSet = statement.executeQuery( sql: "select count(*) COUNT from student");
if(resultSet.next()){
    Integer count = resultSet.getInt( columnLabel: "COUNT");
}
//JDBC Template
Integer count = jt.queryForObject("select count(*) from student", Integer.class);
```

数据库表结构


学生
编号 int <pk>
姓名 varchar(20)
性别 char(2)
出生日期 date
课程
编号 int <pk>
名称 char(20)
学分 int
选课
学生 int <pk, fk2>
课程 int <pk, fk1>
选课时间 datetime
成绩 int

创建项目

- Maven
  - Mysql驱动
  - Spring组件（core、beans、context、aop）
  - JDBC Template（jdbc、tx）
- Spring配置
  - 数据源
  - JDBC Template

JDBC Template基本使用

- execute方法
- update与batchUpdate方法
- query与queryXXX方法
- call方法

JDBC Template基本使用

- update方法
  - 对数据进行增删改操作

```
int update(String sql, Object[] args)
int update(String sql, Object… args)
```

- batchUpdate方法
  - 批量增删改操作

```
int[] batchUpdate(String[] sql)
int[] batchUpdate(String sql, List<Object[]> args)
```

JDBC Template基本使用

- 查询简单数据项
  - 获取一个

```
T queryForObject(String sql, Class<T> type)
T queryForObject(String sql, Object[] args, Class<T> type)
T queryForObject(String sql, Class<T> type,Object… arg)
```

  - 获取多个

```
List<T> queryForList(String sql , Class<T> type)
List<T> queryForList(String sql , Object[] args , Class<T> type)
List<T> queryForList(String sql , Class<T> type,Object… arg)
```

JDBC Template基本使用

- 查询复杂对象（封装为Map）
  - 获取一个

```
Map queryForMap(String sql)
Map queryForMap(String sql , Object[] args)
Map queryForMap(String sql , Object… arg)
```

  - 获取多个

```
List<Map<String,Object>> queryForList(String sql)
List<Map<String,Object>> queryForList(String sql , Object[] args)
List<Map<String,Object>> queryForList(String sql , Object… arg)
```

JDBC Template基本使用

- 查询复杂对象（封装为实体对象）
  - RowMapper接口
  - 获取一个

```
T queryForObject(String sql,RowMapper<T> mapper)
T queryForObject(String sql,Object[] args,RowMapper<T> mapper)
T queryForObject(String sql,RowMapper<T> mapper,Object… arg)
```

  - 获取多个

```
List<T> query(String sql,RowMapper<T> mapper)
List<T> query(String sql,Object[] args,RowMapper<T> mapper)
List<T> query(String sql,RowMapper<T> mapper,Object… arg)
```

JDBC Template持久层示例

- 实体类
- DAO
  - 注入JdbcTemplate
  - 声明RowMapper

优缺点分析

- 优点：
  - 简单
  - 灵活
- 确定：
  - SQL与Java代码参杂
  - 功能不丰富

课程总结

- **持久化操作特点**
  - **必须**
  - **机械性**
- **ORM**
  - **对象-关系**

- JDBC Template是Spring框架对JDBC操作的封装，简单、灵活但不够强大。
- 实际应用中还需要和其它ORM框架混合使用