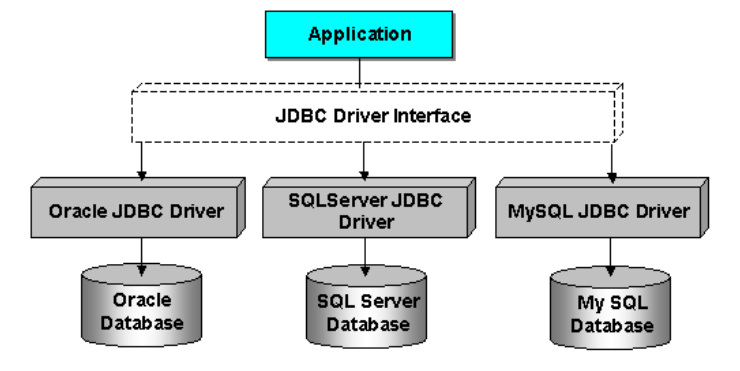
**Spring Boot 自动配置数据源及操作数据库**

**JDBC**

JDBC 全称为 Java Data Base Connectivity（Java 数据库连接），主要由接口组成，是一种用于执行 SQL 语句的 Java API。各个数据库厂家基于它各自实现了自己的驱动程序（Driver），如下图所示：



Java 程序在获取数据库连接时，需要以 URL 方式指定不同类型数据库的 Driver，在获得特定的 Connection 连接后，可按照 JDBC 规范对不同类型的数据库进行数据操作，代码如下：

//第一步，注册驱动程序

//com.MySQL.jdbc.Driver

Class.forName("数据库驱动的完整类名");

//第二步，获取一个数据库的连接

Connection conn = DriverManager.getConnection("数据库地址","用户名","密码");

//第三步，创建一个会话

Statement stmt=conn.createStatement();

//第四步，执行SQL语句

stmt.executeUpdate("SQL语句");

//或者查询记录

ResultSet rs = stmt.executeQuery("查询记录的SQL语句");

//第五步，对查询的结果进行处理

while(rs.next()){

//操作

}

//第六步，关闭连接

rs.close();

stmt.close();

conn.close();

这是初学 JDBC 连接时最熟悉的代码了，虽然现在可能用到了一些数据层 ORM 框架(比如 MyBatis 或者 Hibernate )，但是底层实现依然如同上面代码一样，只不过框架对其做了一些封装而已，通过 JDBC 使得我们可以直接使用 Java 程序来对关系型数据库进行操作，接下来演示 Spring Boot 如何使用 JDBC。

**数据库准备**

**启动 MySQL**

**如果 MySQL 数据库服务没有启动的话需要进行这一步操作。**

命令行中输入以下命令：

sudo service mysql start

因为用户权限的关系，需要增加在命令前增加 sudo 取得 root 权限，不然在启动时会报错，之后等待 MySQL 正常启动即可。

**创建数据库和表结构**

首先，执行如下命令登陆 MySQL 数据库：

sudo mysql -u root

在开发项目之前需要在 MySQL 中先创建数据库和表作为项目演示使用，SQL 语句如下：

CREATE DATABASE /\*!32312 IF NOT EXISTS\*/`lou\_springboot` /\*!40100 DEFAULT CHARACTER SET utf8 \*/;

USE `lou\_springboot`;

DROP TABLE IF EXISTS `tb\_user`;

CREATE TABLE `tb\_user` (

`id` INT(11) NOT NULL AUTO\_INCREMENT COMMENT '主键',

`name` VARCHAR(100) NOT NULL DEFAULT '' COMMENT '登录名',

`password` VARCHAR(100) NOT NULL DEFAULT '' COMMENT '密码',

PRIMARY KEY (`id`)

) ENGINE=INNODB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

首先创建了blog的数据库，之后在数据库中新建了一个名称为 bg\_users 的数据表，表中有 id , name , password 三个字段。

执行完上面的操作后，再执行以下操作，就可以看见创建的数据库了

show databases;

**Spring Boot 连接数据库**

项目结构目录如下：

lou-springboot

├── pom.xml

├── README.md

└── src

├── main

│ ├── java

│ │ └── com

│ │ └── lou

│ │ └── springboot

│ │ ├── Application.java

│ │ └── controller

│ │ ├── HelloController.java

│ │ └── JdbcController.java

│ └── resources

│ ├── application.properties

│ ├── static

│ └── templates

└── test

└── java

└── com

└── lou

└── springboot

└── ApplicationTests.java

#### pom 依赖

在进行数据库连接前，首先将相关 jar 包依赖引入到项目中，Spring Boot 针对 JDBC 的使用提供了对应的 Starter 包：spring-boot-starter-jdbc，方便在 Spring Boot 生态中更好的使用 JDBC，演示项目中使用 MySQL 作为数据库，因此项目中也需要引入 MySQL 驱动包，因此在 pom.xml 文件中新增如下配置：

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

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.1.0.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.lou.springboot</groupId>

<artifactId>springboot-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>springboot-demo</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

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

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- jdbc-starter -->

<dependency>

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

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

<!-- MySQL 驱动包 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

#### 数据库配置信息

为了能够连接上数据库并进行操作，在新增依赖后，我们也需要对数据库的基本信息进行配置，比如数据库地址、账号、密码等信息，这些配置依然是在 application.properties 文件中增加，配置如下：

# datasource config

spring.datasource.url=jdbc:mysql://localhost:3306/my\_blog?useUnicode=true&characterEncoding=utf8&autoReconnect=true&useSSL=false

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.username=root

spring.datasource.password=1002

值得注意的是，在 Spring Boot 2 中，数据库驱动类推荐用 com.mysql.cj.jdbc.Driver，而不是我们平时比较熟悉的 com.mysql.jdbc.Driver 类了。

#### 连接测试

最后，编写一个测试类来测试一下目前能否获得与数据库的连接，在测试类 BlogApplicationTests 中添加 datasourceTest() 单元测试方法，源码及注释如下：

@RunWith(SpringRunner.class)

@SpringBootTest

public class ApplicationTests {

// 注入数据源对象

@Autowired

private DataSource dataSource;

@Test

public void datasourceTest() throws SQLException {

// 获取数据库连接对象

Connection connection = dataSource.getConnection();

// 判断连接对象是否为空

System.out.println(connection != null);

connection.close();

}

}

点击运行单元测试方法，如果 connection 对象不为空，则证明数据库连接成功，如下图所示，在对 connection 对象进行判空操作时，得到的结果是 connection 非空。如果数据库连接对象没有正常获取，则需要检查数据库是否连接或者数据库信息是否配置正确。

**Spring Boot 数据源自动配置**

 Spring Boot 究竟做了哪些自动配置操作使得我们可以如此简单的就直接获取到数据库连接呢？首先，分析一下注入的 DataSource 数据源对象，更改测试类方法如下：

@Test

public void datasourceTest() throws SQLException {

// 获取数据源类型

System.out.println("默认数据源为：" + defaultDataSource.getClass());

}

运行测试方法，得到的结果为：

默认数据源为：class com.zaxxer.hikari.HikariDataSource

基于以上结果可以得出结论，在项目启动前，Spring Boot 已经默认向 IOC 容器中注册了一个类型为 HikariDataSource 的数据源对象，不然在使用 @Autowired 进行数据源的引入时肯定会报错。 HikariCP 是 Spring Boot 2.0 默认使用的数据库连接池，自动配置类为 DataSourceAutoConfiguration.class 和 DataSourceConfiguration.class ，可以阅读一下源码理解一下数据源自动配置的过程。

Spring Boot 不仅仅是自动配置了 DataSource 对象，在数据源对象自动配置完成后，Spring Boot 也自动配置了 JdbcTemplate 对象，JdbcTemplate 是 Spring 对 JDBC 的封装，目的是使 JDBC 更加易于使用。 Spring Boot 默认也并没有集成相关的 ORM 框架，而是提供了 JdbcTemplate 来简化开发者对于数据库的操作，关于 ORM 框架集成到 Spring Boot 项目中后续进行，接下来使用 JdbcTemplate 来进行数据库的常用 SQL 操作。

**Spring Boot 操作 MySQL**

**新建 JdbcController：**

在正确配置数据源之后，开发者可以直接在代码中使用 JdbcTemplate 对象进行数据库操作，接下来就是代码实践，直接新建一个 Controller 类，并注入 JdbcTemplate 对象，接下来我们创建两个方法，一个是查询 bg\_users 中的数据，另一个方法是根据传入的参数向 bg\_users 表中新增数据，代码如下：

package com.lou.springboot.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.util.StringUtils;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

import java.util.Map;

@RestController

public class JdbcController {

//自动配置，因此可以直接通过 @Autowired 注入进来

@Autowired

JdbcTemplate jdbcTemplate;

// 查询所有记录

@GetMapping("/users/queryAll")

public List<Map<String, Object>> queryAll() {

List<Map<String, Object>> list = jdbcTemplate.queryForList("select \* from tb\_user");

return list;

}

// 新增一条记录

@GetMapping("/users/insert")

public Object insert(String name, String password) {

if (StringUtils.isEmpty(name) || StringUtils.isEmpty(password)) {

return false;

}

jdbcTemplate.execute("insert into tb\_user(`name`,`password`) value (\"" + name + "\",\"" + password + "\")");

return true;

}

}

两个方法，分别取查询数据库记录和新增数据库记录。

**功能验证**

**启动 Spring Boot 项目**

由于默认是在 project 目录下，因此想要启动项目，首先需要切换到项目目录下，之后可以通过 Maven 插件的方式启动 Spring Boot 项目，命令为 mvn spring-boot:run ，之后就可以等待项目启动。

**打开 Web 服务**

在项目启动成功后，可以点击页面上方的 Web 服务直接在显示查看网站效果。