

## 8

### 综合实例 充电站管理系统

## ◆ 8.1 背景知识 -- Spring简介

---

- 2004年， Spring Framework 1.0正式发布
- Spring是一个轻量级控制反转(IoC)和面向切面(AOP)的容器框架
- Spring是Java企业版(Java Enterprise Edition, JEE, 也称J2EE)的轻量级替代品， Spring为企业级Java开发提供了一种相对简单的方法，用简单的Java对象(Plain Old Java Object, POJO)实现了EJB的功能

## ◆ 8.1 背景知识 -- Spring产生的初衷

---

- (1) JAVA EE开发应该更加简单;
- (2) 使用接口而不是使用类, 是更好的编程习惯;
- (3) 为JavaBean提供了一个更好的应用配置框架;
- (4) 更多地强调面向对象的设计;
- (5) 尽量减少不必要的异常捕捉;
- (6) 使应用程序更加容易测试。

## ◆ 8.1 背景知识 -- Spring能力

---

- (1) 最完善的轻量级核心框架；
- (2) 通用的事务管理抽象层；
- (3) JDBC抽象层；
- (4) 灵活的MVC Web应用框架。

## ◆ 8.1 背景知识 -- SpringBoot

---

- 从本质上说，Spring Boot就是Spring
- Spring Boot把你从复杂的配置工作中解放出来，从而可以聚焦于项目开发本身。
- 有了Spring Boot，可以很容易创建一个Spring框架的项目，而几乎不用进行Spring配置

## ◆ 8.1 背景知识 -- IntelliJ IDEA介绍

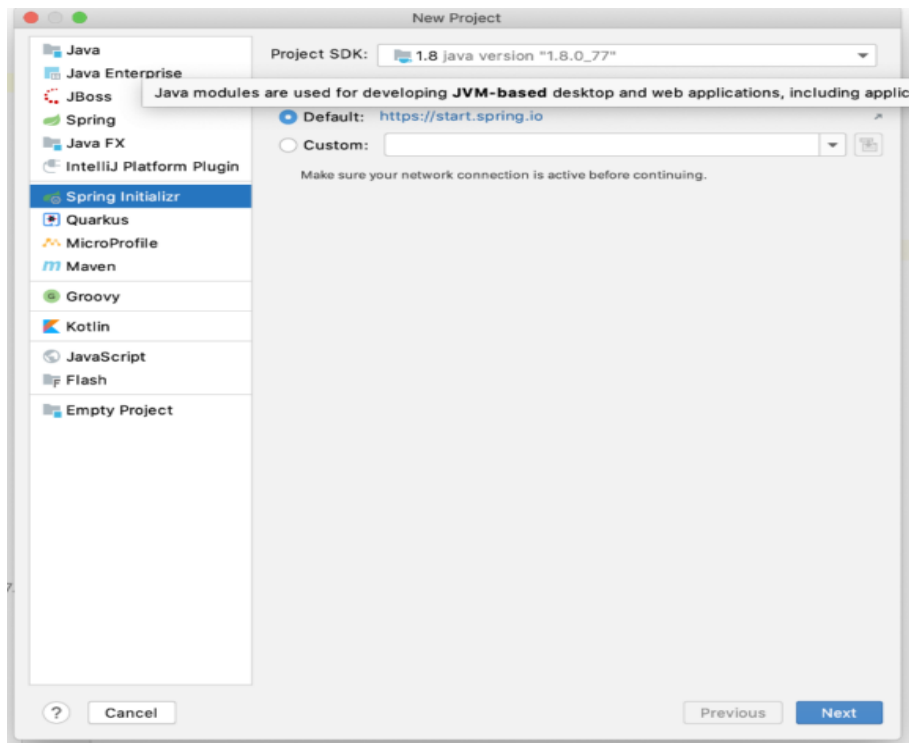
---

- IntelliJ IDEA，是Java编程语言开发的集成环境
- IntelliJ IDEA在业界被公认为最好的Java开发工具
- 智能代码助手、代码自动提示、重构、J2EE支持 ...

## ◆ 8.2 创建一个Spring Boot项目（1）

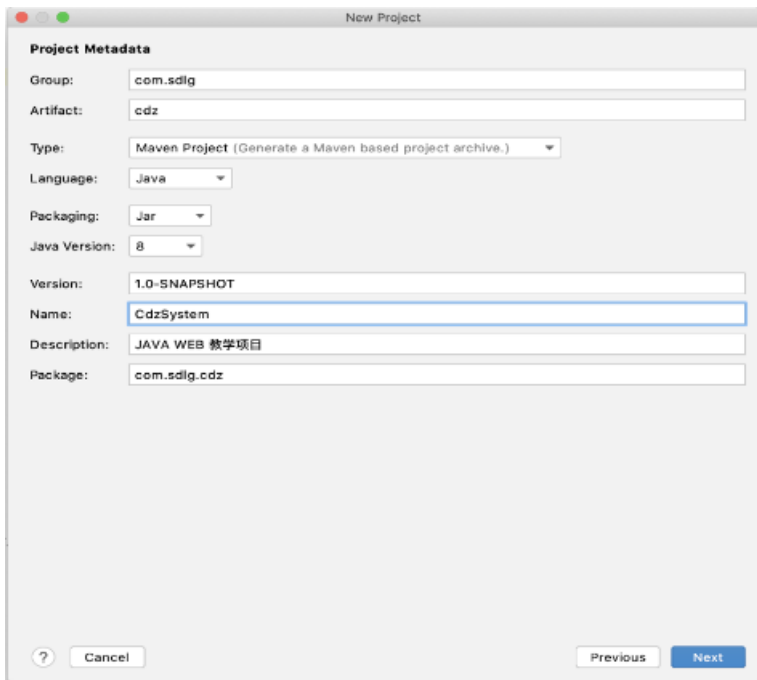


## ◆ 8.2 创建一个Spring Boot项目（2）





## ◆ 8.2 创建一个Spring Boot项目（3）

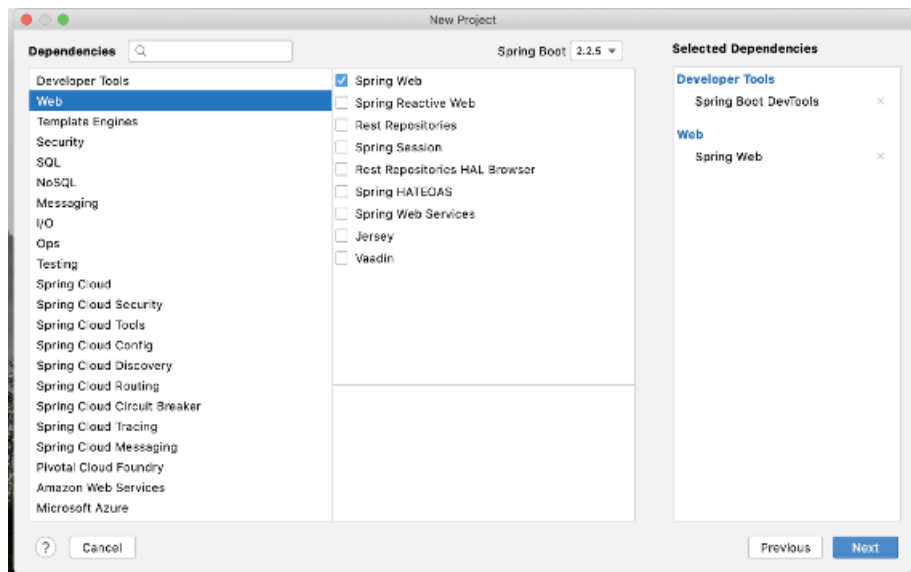


The screenshot shows the 'New Project' dialog box with the following configuration:

Field	Value
Group	com.sd1g
Artifact	cdz
Type	Maven Project (Generate a Maven based project archive.)
Language	Java
Packaging	Jar
Java Version	8
Version	1.0-SNAPSHOT
Name	CdzSystem
Description	JAVA WEB 教学项目
Package	com.sd1g.cdz

- **Group:** 组织名，对应Java包的结构
- **Artifact:** 项目唯一标识，对应项目名称，要求全小写字母。由于我们实践的项目是充电站管理系统，所以我们使用名称cdz
- **Type:** 项目类型，此处不需要修改，保持Maven Project即可
- **Packaging:** 打包类型，默认打包成Jar文件即可
- **Java Version:** JDK版本
- **Language:** 开发语言，默认选中JAVA即可
- **Version:** 初始项目版本
- **Description:** 项目描述，里面用一句话简短的介绍下项目
- **Name:** 项目名称，我们使用名称CdzSystem
- **Package:** 包名，默认由Group+ Artifact组合而成

## ◆ 8.2 创建一个Spring Boot项目（4）





## ◆ 8.3 项目结构 -- 启动类

---

- 在项目中，src为项目代码目录，Spring Initializr为我们在cdz包里默认创建了一个CdzSystemApplication文件，这是Spring的启动执行类：

```
@SpringBootApplication
```

```
public class CdzSystemApplication {
```

```
    public static void main(String[] args) {
```

```
        SpringApplication.run(CdzSystemApplication.class, args);
```

```
    }
```

```
}
```

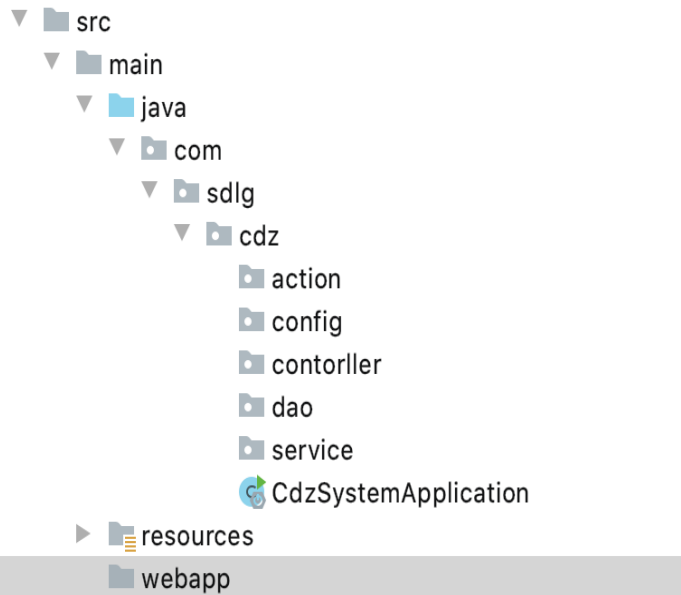
## ◆ 8.3 项目结构 -- Spring注解

---

注解是JDK5.0引入的一种代码注释说明机制，注解一般有以下三类作用：

- 格式检查：告诉编译器信息，比如被@Override标记的方法如果不是父类的某个方法，IDE会报错；
- 减少配置：运行时动态处理，得到注解信息，实现代替配置文件的功能；比如@SpringBootApplication，代表被注解的类是一个Spring启动类。
- 减少重复工作：比如@Autowired，它可以帮我们对被注解的元素自动完成装配，我们会在后面使用时详细说明。

## ◆ 8.4 项目设计 -- 代码层级设计



- resources目录暂时只存放配置文件 application.properties
- webapp存放用jsp和js编写的前端页面代码
- java目录存放我们用java语言写的代码, 我们大多数后台逻辑代码都存放在此处。

## ◆ 8.5 登录功能开发(1)

---

开发一个登录功能，我们需要开发以下三部分：

- 数据库新增一个用户表user，用来存储用户的用户名，密码
- webapp目录中开发登录对应的jsp页面
- java目录下开发用户表数据库设计对应的登录逻辑后台代码，用来响应用户的登录行为。

## ◆ 8.5 登录功能开发(2) -- 用户表数据库设计

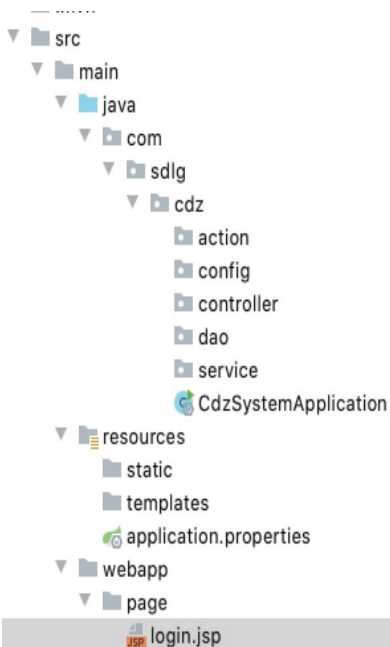
---

user表中新增三个字段即可满足我们需求：

- id作为自增主键
- user\_name作为用户名，因为用户名一般为字符串，所以我们字段的类型设置为VARCHAR，长度设置为32即可
- password作为用户密码，用户密码一般也是字符串，所以类型同样设置为VARCHAR。



## ◆ 8.5 登录功能开发(3) --新增login.jsp



- 创建目录page，然后在page目录下新建一个login.jsp文件。
- 打开application.properties配置文件，在其中添加以下几行：

```
spring.mvc.view.prefix= /page/
```

```
spring.mvc.view.suffix= .jsp
```

```
spring.http.encoding.force=true
```

```
spring.http.encoding.charset=UTF-8
```

```
spring.http.encoding.enabled=true
```

```
server.tomcat.uri-encoding=UTF-8
```

## ◆ 8.5 登录功能开发(4) -- Spring Controller

---

- 在Spring框架中，DispatcherServlet负责分发请求到控制器Controller 处理。
- Controller负责接收用户请求，Controller收到用户请求后，经过一定处理后，将处理结果返回给对应View进行显示。
- Spring定义一个Controller非常简单，在这个类上添加一个注解@Controller即可。

## ◆ 8.5 登录功能开发(5) --响应登录请求的Controller

---

```
import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;

@Controller
public class MainController {
    @RequestMapping(path = "/login")
    public String loginPage() { return "login"; }
```

- 我们在MainController上面添加了@Controller说明这个类是一个Spring的Controller。现在MainController就可以响应用户的请求。
- 方法loginPage添加了一个注解@ RequestMapping。这个被注解的方法将响应用户的login请求。在方法中，我们直接返回了一个字符串login，这里的意思是说，将login.jsp页面返回给用户。

## ◆ 8.5 登录功能开发(6) -- 页面发起登录行为

```
$(document).ready(function () {  
    $("#confirmBtn").click(function () {  
        var _name = $("#loginName").val();  
        var _password = $("#loginPassword").val();  
        $.ajax({  
            url: "/api/login",  
            type: "post",  
            data: {  
                name: _name,  
                password: _password,  
            },  
            success: function (data) {  
                if (data.code === 0) {  
                    window.location.href = "/main";  
                } else {  
                    alert(data.data);  
                }  
            },  
            error: function (xhr, textStatus, errorThrown) {  
                alert(errorThrown);  
            }  
        });  
    });  
});
```

- 我们可以将loginName和loginPassword两个input中的数据传递给后台的/api/login 接口。
- 一旦后台处理成功，我们可以在success函数中，根据后台返回的结果，进行相应的处理。

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/016210055040010213>