什么是Spring Boot

Spring Boot是Spring框架下的子项目。Spring框架是事实上的Java企业级应用开发标准（开源社区），它最初由Rod Johnson开发，提供了强大的IOC、AOP和Web MVC等功能。目前Spring框架包含Spring Framework、Spring Data、Spring Cloud、Spring Security等子项目，为开发企业应用提供了一揽子解决方案。但早期Spring的配置比较复杂，为了简化Spring应用的搭建过程，Pivotal团队开发了Spring Boot。 Boot有启动之意，使用Spring Boot不会增加开发人员的负担，相反能够极大的简化项目配置。因此从Spring Boot开始学习Spring框架更为便捷。

注意：由于Spring Boot隐藏了许多Spring框架的细节，因此如果希望深入理解Spring框架的原理，还是需要掌握Spring框架的配置，但对于工期紧任务重的软件项目，是否理解Spring的原理，并不重要（懂怎么用就行）。**如果希望成为高水平的软件开发者，掌握Spring框架的原理是必须的。**

下面我们直接从Spring Boot开始，开发Restful风格的CRUD(Create Read Update Delete)]服务。

用Maven创建Spring Boot项目

在此之前，你应该已经掌握了Maven的使用，如果还没有掌握，请先看这个教程。此外，在国内开发，设置Maven国内镜像是必须的（动手百度立刻有答案）。

按如下步骤进行操作：

创建`maven-archetype-quickstart`类型的项目

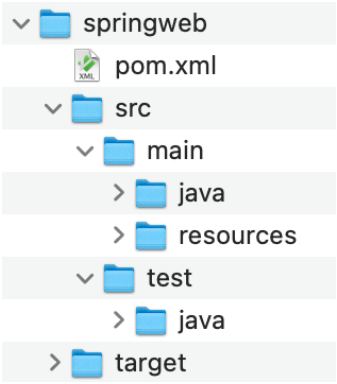
执行

```
mvn archetype:generate "-DgroupId=cn.cqu.edu" "-DartifactId=springweb" "-DarchetypeArtifactId=maven-archetype-quickstart" "-DinteractiveMode=false"
```

参数说明：

- **-DgroupId**: 组织名，公司网址的反写 + 项目名称
- **-DartifactId**: 项目名-模块名
- **-DarchetypeArtifactId**: 指定 ArchetypeId，maven-archetype-quickstart，创建一个简单的 Java 应用
- **-DinteractiveMode**: 是否使用交互模式

成功之后所创建的目录结构如下：



各个目录说明如下：

目录名称	描述
springweb	项目名称，根目录下包含 src 文件夹和 pom.xml
src/main/java	java 代码文件在包结构下（cn/cqu/edu）
src/main/test	测试代码文件在包结构下（cn/cqu/edu）
src/main/resources	包含了 图片 / 属性 文件（**需要手动创建这个目录**）
target	项目编译的class文件、打包的jar和war文件

修改pom.xml文件

修改pom.xml为如下形式：

```
<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/maven-v4_0_0.xsd">
```

```
  <modelVersion>4.0.0</modelVersion>
  <groupId>cn.cqu.edu</groupId>
  <artifactId>springweb</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>springboot</name>
  <url>http://maven.apache.org</url>
  <properties>
    <!-- 设置编译使用JDK1.8 -->
    <java-version>1.8</java-version>
    <!-- 设置编码为UTF-8 -->
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <maven.compiler.encoding>UTF-8</maven.compiler.encoding>
    <failOnMissingWebXml>false</failOnMissingWebXml>
  </properties>
  <!-- 设置项目的parent为spring boot -->
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <!-- spring boot版本可根据需要修改 -->
    <version>2.0.1.RELEASE</version>
  </parent>
  <dependencies>
    <!-- spring-boot-starter依赖（必须） -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter</artifactId>
    </dependency>
    <!-- spring-boot-starter-web依赖（web开发必须） -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <!-- junit依赖（junit测试必须） -->
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <!-- 设置构建插件 -->
    <finalName>springweb</finalName>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
```

```
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
          <!-- 构建时跳过junit测试（可以根据需要设置） -->
          <skipTests>true</skipTests>
        </configuration>
      </plugin>
      <!-- 设置spring-boot-maven构建插件 -->
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

修改App.java

修改App.java为如下：

```
package cn.cqu.edu;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class App
{
    public static void main(String[] args) {
        SpringApplication.run(App.class, args);
    }
}
```

增加MathController.java

在src/main/java/cn/cqu/edu/controller目录下创建MathController.java文件，内容如下：

```
package cn.cqu.edu.controller;

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

@RestController
public class MathController {
  @GetMapping(value="/add")
  public double add(double a,double b)
  {
```

```
    return a+b;
  }
}
```

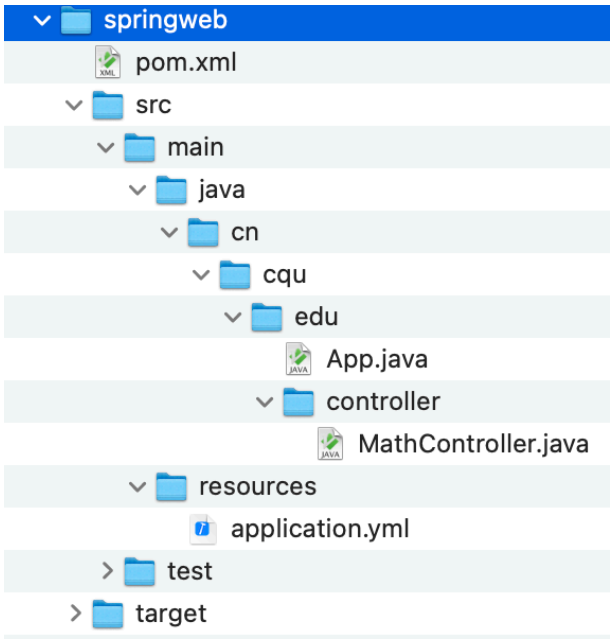
增加application.yml

在src/main/resources目录下增加application.yml文件，内容如下：

```
# 设置端口为8080
server:
  port: 8080
```

运行Spring Boot项目

完成后的项目目录格式如下：



在项目根目录（springweb），执行`mvn spring-boot:run`指令启动spring boot项目

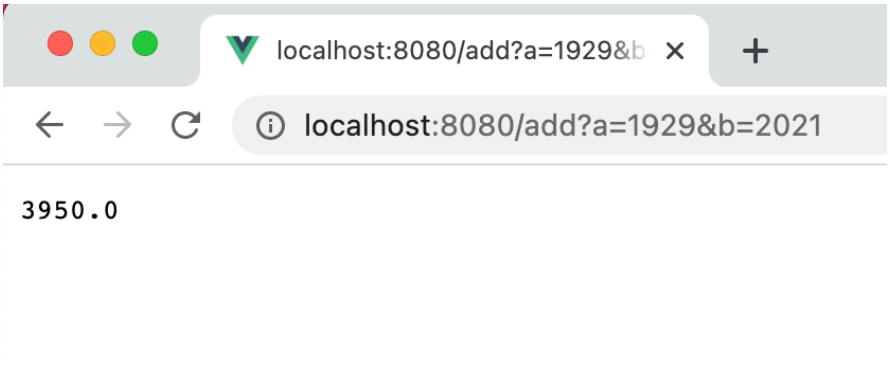
```
mvn spring-boot:run
```

看到如下输出，说明项目启动成功

```
2021-08-26 10:33:10.158  INFO 38439 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2021-08-26 10:33:10.159  INFO 38439 --- [ost-startStop-1] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 2043 ms
2021-08-26 10:33:10.389  INFO 38439 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean  : Servlet dispatcherServlet mapped to [/]
2021-08-26 10:33:10.395  INFO 38439 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'characterEncodingFilter' to: [/*]
2021-08-26 10:33:10.396  INFO 38439 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
2021-08-26 10:33:10.396  INFO 38439 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'httpPutFormContentFilter' to: [/*]
2021-08-26 10:33:10.397  INFO 38439 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to: [/*]
2021-08-26 10:33:10.598  INFO 38439 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2021-08-26 10:33:11.100  INFO 38439 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@7f6f177b: startup date [Thu Aug 26 10:33:08 CST 2021]; root of context hierarchy
2021-08-26 10:33:11.287  INFO 38439 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/add],methods=[GET]}" onto public double cn.cqu.edu.controller.MathController.add(double,double)
2021-08-26 10:33:11.294  INFO 38439 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>> org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2021-08-26 10:33:11.296  INFO 38439 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)
2021-08-26 10:33:11.360  INFO 38439 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2021-08-26 10:33:11.361  INFO 38439 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2021-08-26 10:33:11.780  INFO 38439 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on startup
2021-08-26 10:33:11.882  INFO 38439 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2021-08-26 10:33:11.898  INFO 38439 --- [           main] cn.cqu.edu.App                           : Started App in 4.601 seconds (JVM running for 10.385)
```

在浏览器中测试

打开浏览器，输入`http://localhost:8080/add?a=1929&b=2021`，浏览器中返回`1929+2021`的计算结果



打包发布Spring Boot项目

在项目根目录（springweb），执行`mvn package`指令对spring boot项目进行打包

```
mvn package
```

打包后在target目录下生成文件`target/springweb.jar`，在命令行下执行如下指令，启动Spring boot项目

```
java -jar springweb.jar
```

用Docker发布Spring Boot项目

在项目根目录（springweb），编写如下Dockerfile文件

```
# 以openjdk:8-jdk-alpine镜像为基础
FROM openjdk:8-jdk-alpine
# 拷贝打包后的文件到镜像
COPY target/springweb.jar /root/springweb.jar
# 暴露8080端口
EXPOSE 8080
# 启动镜像时，启动springboot项目
ENTRYPOINT ["java","-jar","/root/springweb.jar"]
```

在项目根目录（springweb），用下列指令创建Docker镜像

```
docker build -t springweb .
```

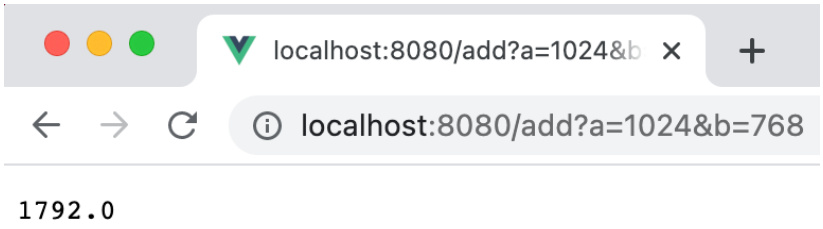
镜像创建成功后，执行如下指令启动容器

```
docker run -it --rm --name springweb -p 8080:8080 springweb
```

执行指令`docker ps`可以查看启动的docker容器

```
CONTAINER ID   IMAGE       COMMAND                  CREATED              STATUS              PORTS                    NAMES
8dd83a5f8135   springweb   "java -jar /root/spr…"   About a minute ago   Up About a minute   0.0.0.0:8080->8080/tcp   springweb
```

打开浏览器，输入`http://localhost:8080/add?a=1024&b=768`，浏览器中返回`1024+768`的计算结果



发布到K8s集群

有兴趣的可以自学K8s（kubernetes），此视频讲解如何用腾讯云的K8s集群部署应用。

基于MongoDB开发CRUD应用

MongoDB是一种NoSQL数据库，为什么使用NoSQL数据库？因为大二的同学还没有学过关系数据库。关系数据库的学习需要至少一个学期的课程，而NoSQL数据库只需要1-2个小时的学习时间，学生就可以掌握基本的CRUD操作。MongoDB是一种文档型数据库，它**不需要**学生掌握：

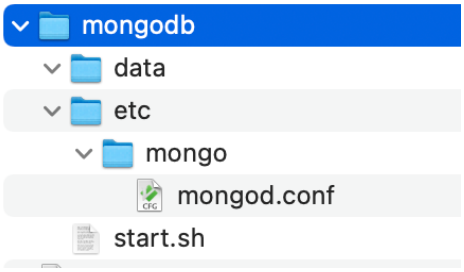
1. E-R模型
2. 事务和日志的原理
3. 数据库范式
4. 关系表的连接（left join）
5. 视图
6. 触发器

学生能够在1-2个小时内掌握MongoDB的CRUD操作，而MongoDB非常适合作为CRUD类应用的数据存储，例如微博评论、个人档案、图书信息等数据的存储。

MongoDB的操作请参考菜鸟MongoDB教程，推荐用Docker轻量化安装MongoDB。

用Docker轻量化运行MongoDB

在命令行下进入mongodb目录，目录下有如下文件：



`start.sh`是启动MongoDB的脚本文件，其内容为：

```
docker run --rm -d -p 27017:27017 -v [mongodb目录绝对路径]/data:/data/db -v [mongodb目录绝对路径]/etc/mongo:/etc/mongo  --name mongodb mongo --config /etc/mongo/mongod.conf
```

直接在`mongodb`目录下执行上述指令，即可启动MongoDB。MongoDB启动后，执行如下指令连接MongoDB Shell：

```
docker exec -it mongodb mongo -u root -p 123456
```

教程提供的数据库管理员用户为`root`，密码为`123456`。只要出现下面界面，则表明数据库运行成功。

```
liuji@LiuJi-MacBook-Pro-2018 ~ % docker exec -it mongodb mongo -u root -p 123456
MongoDB shell version v4.4.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("dfc616a0-1fa8-4859-8b6e-69bfeacb3dee") }
MongoDB server version: 4.4.6
> use demo      切换到demo
switched to db demo
> db.demo.find()  查询demo中的所有数据
{ "_id" : ObjectId("6127421d3a970d1aad8769f1"), "name" : "测试" }
>
```

使用Spring Boot实现MongoDB的CRUD

修改pom.xml增加spring-boot-starter-data-mongodb依赖

pom.xml文件修改如下：

```
<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
```

```xml
  <groupId>cn.cqu.edu</groupId>
  <artifactId>springweb</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>springboot</name>
  <url>http://maven.apache.org</url>
  <properties>
    <!-- 设置编译使用JDK1.8 -->
    <java-version>1.8</java-version>
    <!-- 设置编码为UTF-8 -->
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <maven.compiler.encoding>UTF-8</maven.compiler.encoding>
    <failOnMissingWebXml>false</failOnMissingWebXml>
  </properties>
  <!-- 设置项目的parent为spring boot -->
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <!-- spring boot版本可根据需要修改 -->
    <version>2.0.1.RELEASE</version>
  </parent>
  <dependencies>
    <!-- spring-boot-starter依赖（必须） -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter</artifactId>
    </dependency>
    <!-- spring-boot-starter-web依赖（web开发必须） -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <!-- mongodb的依赖 -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-mongodb</artifactId>
    </dependency>
    <!-- junit依赖（junit测试必须） -->
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <!-- 设置构建插件 -->
    <finalName>springweb</finalName>
```

```
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
          <!-- 构建时跳过junit测试（可以根据需要设置） -->
          <skipTests>true</skipTests>
        </configuration>
      </plugin>
      <!-- 设置spring-boot-maven构建插件 -->
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

修改application.yml文件增加MongoDB数据库连接

application.yml修改如下：

```
# 设置端口为8080
server:
  port: 8080
spring:
   data:
      mongodb:
         authentication-database: admin
         database: demo
         username: root
         password: "123456" #注意这里要用引号把密码括起来
         host: 127.0.0.1
         port: 27017
```

增加User.java描述要存储的数据

User.java代码如下：

```
package cn.cqu.edu.domain;

import java.util.Date;

import org.springframework.data.annotation.Id;

public class User {
```

```
    @Id
    private String userId;
    private String name;
    private Integer age;
    private Date createTime = new Date();
  public String getUserId() {
    return userId;
  }
  public void setUserId(String userId) {
    this.userId = userId;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public Integer getAge() {
    return age;
  }
  public void setAge(Integer age) {
    this.age = age;
  }
  public Date getCreateTime() {
    return createTime;
  }
  public void setCreateTime(Date createTime) {
    this.createTime = createTime;
  }
}
```

增加UserRepository.java实现CRUD操作

UserRepository.java代码如下：

```
package cn.cqu.edu.repository;

import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import cn.cqu.edu.domain.User;

@Repository
public interface UserRepository extends MongoRepository<User, String> {
}
```

增加UserController.java实现CRUD的Restful服务

UserController.java代码如下：

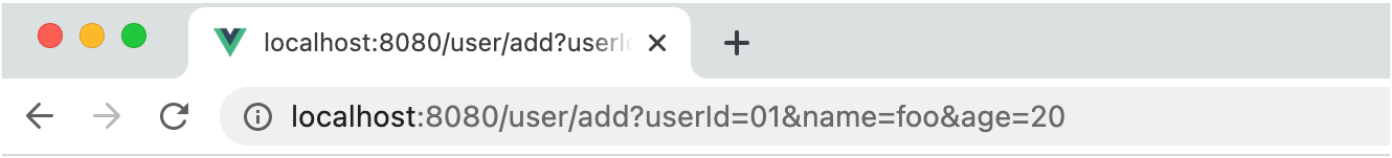
```java
package cn.cqu.edu.controller;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import cn.cqu.edu.domain.User;
import cn.cqu.edu.repository.UserRepository;

@RestController
public class UserController {
  @Autowired
  private UserRepository userRepository;
  @RequestMapping(value="/user/add")
  public User add(User user)
  {
    return userRepository.insert(user);
  }
  @RequestMapping(value="/user/findAll")
  public List<User> findAll()
  {
    return userRepository.findAll();
  }
  @RequestMapping(value="/user/findById")
  public Optional<User> findById(String userId)
  {
    return userRepository.findById(userId);
  }
  @RequestMapping(value="/user/update")
  public User update(User user)
  {
    if(userRepository.existsById(user.getUserId()))
      return userRepository.save(user);
    else
      return null;
  }
  @RequestMapping(value="/user/delete")
  public boolean delete(String userId)
  {
    userRepository.deleteById(userId);
    if(userRepository.existsById(userId))
      return false;
```

```
46      else
47        return true;
48    }
49  }
```

在浏览器中测试CRUD应用

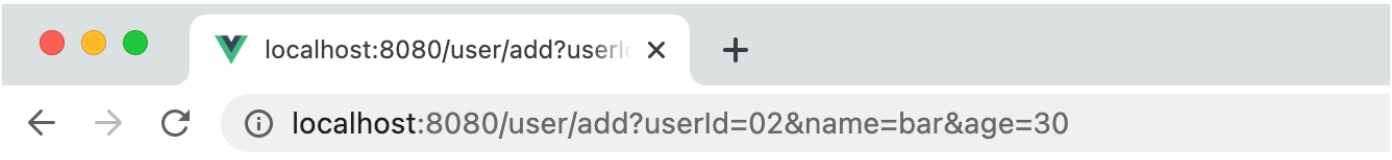
测试Create

在浏览器地址栏中输入：`http://localhost:8080/user/add?userId=01&name=foo&age=20`，插入数据。




```
{"userId":"01","name":"foo","age":20,"createTime":"2021-08-28T02:14:08.079+0000"}
```

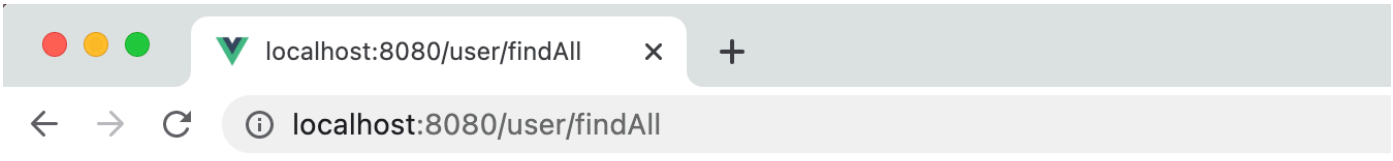

在浏览器地址栏中输入：`http://localhost:8080/user/add?userId=02&name=bar&age=30`，插入数据。




```
{"userId":"02","name":"bar","age":30,"createTime":"2021-08-28T02:15:21.942+0000"}
```

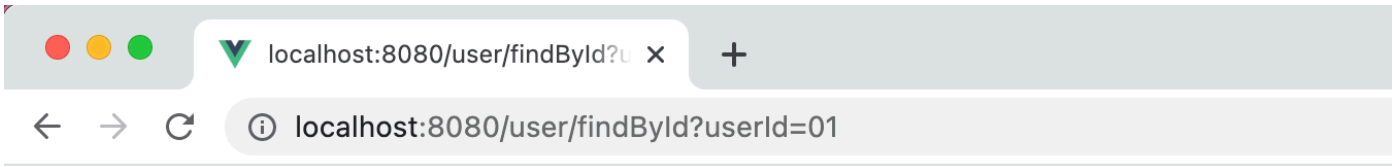

测试Read

在浏览器地址栏中输入：`http://localhost:8080/user/findAll`，查询全部数据。




```
[{"userId":"01","name":"foo","age":20,"createTime":"2021-08-28T02:14:08.079+0000"},
{"userId":"02","name":"bar","age":30,"createTime":"2021-08-28T02:15:21.942+0000"}]
```

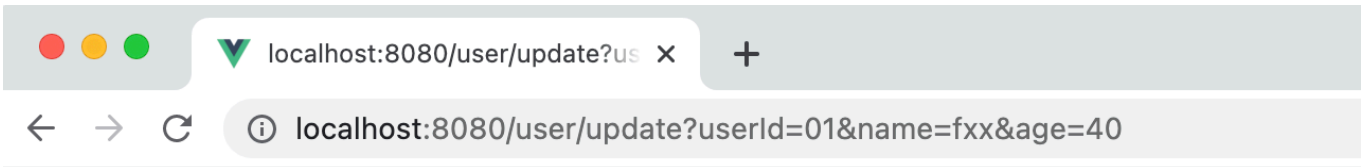

在浏览器地址栏中输入：`http://localhost:8080/user/findById?userId=01`，查询`userId=01`的数据。



{"userId":"01","name":"foo","age":20,"createTime":"2021-08-28T02:14:08.079+0000"}

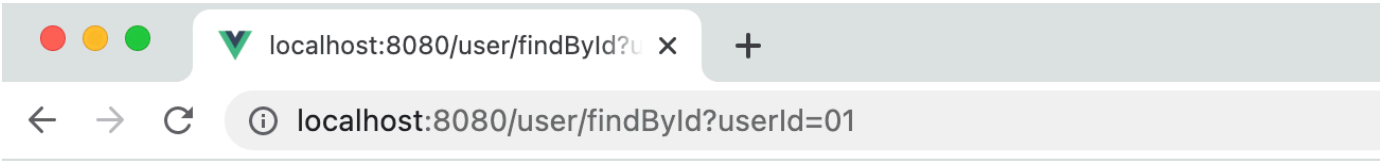
测试Update

在浏览器地址栏中输入：`http://localhost:8080/user/update?userId=01&name=fxx&age=40`，修改`userId=01`数据。



{"userId":"01","name":"fxx","age":40,"createTime":"2021-08-28T02:22:09.902+0000"}

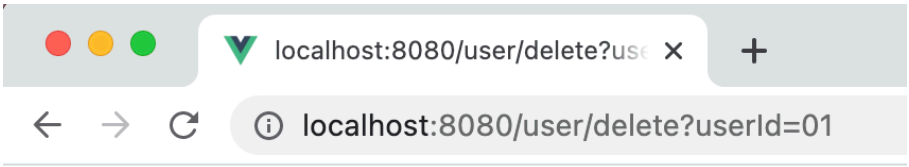
在浏览器地址栏中输入：`http://localhost:8080/user/findById?userId=01`，查询`userId=01`的数据，可以看到数据已经被修改了。



{"userId":"01","name":"fxx","age":40,"createTime":"2021-08-28T02:22:09.902+0000"}

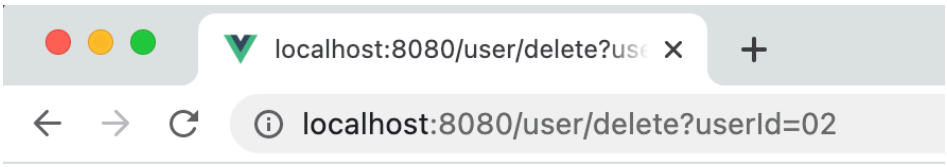
测试delete

在浏览器地址栏中输入：`http://localhost:8080/user/delete?userId=01`，删除`userId=01`数据。



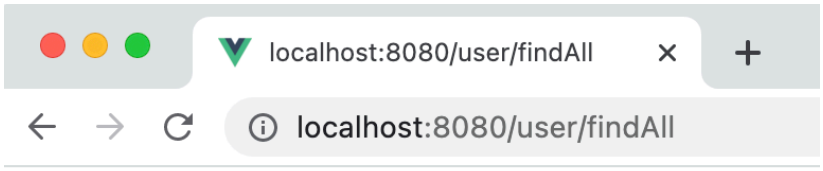
true

在浏览器地址栏中输入：`http://localhost:8080/user/delete?userId=02`，删除`userId=02`数据。




```
true
```

在浏览器地址栏中输入：`http://localhost:8080/user/findAll`，查询全部数据，数据全部被清空了。




```
[]
```

作业

用互联网查询资料，为MongoDB的CRUD应用开发用户界面，要求：

1. 采用前后端分离模式，前端通过Ajax请求访问后端的CRUD Restful服务
2. 前端采用JavaScript框架实现，例如采用Vue
3. 示例代码没有解决跨域调用问题（CROS），请查找资料，解决该问题