



Java EE 架构与应用

基于 SpringMVC 的体育健身网站设计报告

小组成员：

覃依依 16301087

王佳乐 16301076

创建日期：

2019/4/29

目录

1.概述.....	3
2.总体设计.....	3
2.1 开发环境	3
2.2 基本涉及描述	3
3.详细设计.....	3
3.1 主要界面流程描述	4
3.1.1 主页	4
3.1.2 注册	5
3.1.3 登录	6
3.1.4 查看或搜索课程内容	8
3.1.1 查看或搜索教练	9
3.1.5 查看体育馆位置	11
3.2 数据库设计	11
3.3 缓存功能	13

1. 概述

该项目为基于 SpringMVC 架构的体育健身网站，目前实现主要功能为：

- 新用户的注册（详见 3.1.2）
- 旧用户的登录（详见 3.1.3）
- 查看或搜索课程内容（详见 3.1.4）
- 查看或搜索教练（详见 3.1.5）
- 查看体育馆位置（详见 3.1.6）
- 多表链接的查询（详见 3.1.5）
- 查询信息的分页（详见 3.1.5）
- 多数据源的查询（详见 3.2）
- 缓存功能（详见 3.3）

2. 总体设计

2.1 开发环境

该项目涉及到的开发环境及工具如下表所示。

开发工具	作用
Eclipse Mars.2	Java EE 开发平台
Springframework 2.1.3	Springboot 架构
MySQL 8.0.16	数据库工具
Hibernate	关系映射框架
Tomcat 8.5.40	Java Web 服务器
Redis 3.2.100	可基于内存亦可持久化的 Key-Value 数据库，用于提供缓存

表 2-1：开发环境

2.2 基本设计描述

该项目基于 Spring 实现了 Web MVC 设计模式的请求驱动类型，大致分别为 Web 层（使用了 MVC 架构模式的思想进行职责解耦）、业务层、数据访问层和底端 JDBC，其中数据访问层可分为 DAO 层和持久层。项目包架构和各个包的作用如下所示。

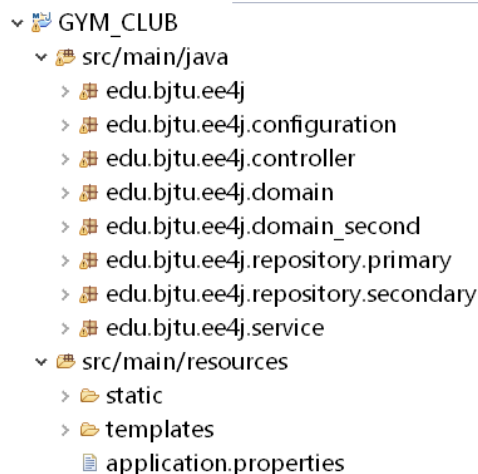


图 2-1：项目包架构

包名	作用
.configuration	用于存放项目的配置文件，实现依赖注入
.controller	用于存放用于 url 映射的控制器
.domain	用于存放第一数据库的实体类
.domain_second	用于存放第二数据库的实体类
.repository.primary	用于存放第一数据库涉及的接口类，通过注解写入方法
.repository.secondary	用于存放第二数据库涉及的接口类，通过注解写入方法
.service	用于实现数据库的接口方法

表 2-2：src/main/java 中的包介绍

包名	作用
.static	用于存放 html 中所使用的静态元素
.templates	用于存放网页文件

表 2-3：src/main/resource 中的包介绍

3. 详细设计

3.1 主要界面流程描述

3.1.1 主页

如下所示，当用户访问 <http://localhost:8080> 或 <http://localhost:8080/home> 时，Dispatchcher Servlet 接收到该 GET 请求并找到对应的 url 映射分发给 Controller，其将用户导航至 index.html 页面。

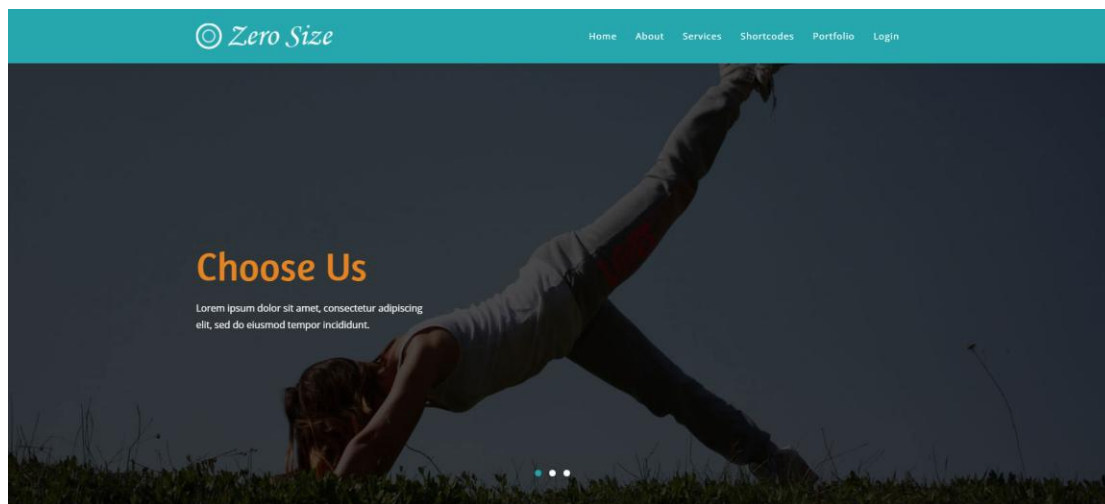


图 3-1：主页

3.1.2 注册

如下所示，用户点击主页的 Login 按钮，跳转至用户注册界面，用户填写完表单后点击“Save”按钮，前端验证用户输入是否合法，如若合法将以 POST 请求提交该表单，Dispatcher Servlet 接收到该 POST 请求并找到对应的 url 映射分发给 Controller，Controller 进行调用相应的数据库操作。

图 3-2：注册页面

图 3-3：格式验证提示

```
@CachePut(value = "person", key = "#person.id",unless="#judge=false")
@RequestMapping(value = "/register", method = RequestMethod.POST)
//ResponseBody
public String Register(@Valid Person person, BindingResult bindingResult, Model model,HttpServletResponse response,
    boolean judge=true;
    if (bindingResult.hasErrors()) {
        judge=false;
        return "/contact";
    }
}
if(this.PersonService.getUser(person.getEmail())!=null){
    model.addAttribute("err", "The email has been registered!");
    judge=false;
    return "/contact";
}
else if(this.PersonService.getUser1(person.getPhone_no())!=null){
    model.addAttribute("err1", "The phone has been registered!");
    judge=false;
    return "/contact";
}
this.PersonService.savePerson(person);
try {
    this.addCookie(person.getPhone_no(),"", response, request);
    this.addCookie(person.getEmail().replace("@",""),"", response, request);
} catch (UnsupportedEncodingException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
return "redirect:/contacts/login1";
}
```

图 3-4：核心代码

3.1.3 登录

如下所示，用户点击主页的 Login 按钮，跳转至用户登录界面，用户填写完表单后点击“Save”按钮，前端验证用户输入是否合法，如若合法将以 POST 请求提交该表单，Dispatchcher Servlet 接收到该 POST 请求并找到对应的 url 映射分发给 Controller，Controller 进行调用相应的数据库操作判断该用户验证是否合法，如用户名密码一致系统便将其导航至主页。

用户可以使用邮箱或注册手机号登录。

此处我们添加了记住密码功能，将用户名和密码存入浏览器 cookie 中。

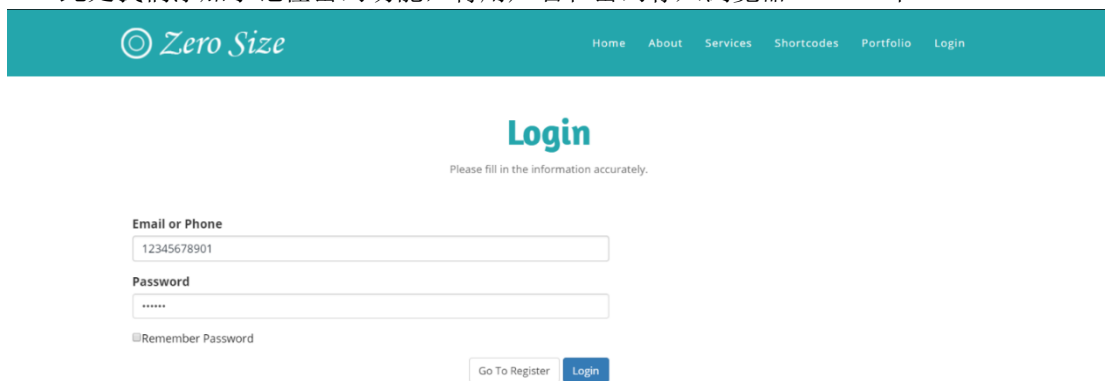


图 3-5：登录页面



图 3-6 提示信息

```
@RequestMapping(value = "/login2",method = RequestMethod.POST)
public String viewPerson(@Valid Person person, BindingResult bindingResult, Model model,HttpServletResponse respons
HttpSession session=request.getSession());
boolean judge=true;
if(session.getValue("name")==null){
    if(person.getEmail()!=null){
        String mailRegex,mailName,mailDomain;
        mailName="^[0-9a-z]+\\w*"; //^表明一行以什么开头;^[0-9a-z]表明要以数字或小写字母开头;\\w*表明匹配任意个大写小写字母
        mailDomain="([0-9a-z]+\\.)+[0-9a-z]+$"; //***.***.***格式的域名,其中*为小写字母或数字;第一个括号代表有至少一个
        mailRegex=mailName+"@"+mailDomain; //邮箱正则表达式 ^[0-9a-z]+\\w*@([0-9a-z]+\\.)+[0-9a-z]+$
        Pattern pattern=Pattern.compile(mailRegex);
        Matcher matcher=pattern.matcher(person.getEmail());
        if(!matcher.matches()){
            model.addAttribute("err1", "Wrong Email.");
            judge=false;
            return "/login";
        }
    }
    if(this.PersonService.getUser(person.getEmail()).equals(person.getPassword())){
        session.putValue("name", person.getEmail());
        String str[]=request.getParameterValues("pass");
        if(str!=null && str[0].equals("on")){
            System.out.println("保存了密码:");
            try {
                this.addCookie(person.getEmail().replace("@",""),person.getPassword(), response, request);
            } catch (UnsupportedEncodingException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
    else{
        try {
            this.addCookie(person.getEmail().replace("@",""), "", response, request);
        } catch (UnsupportedEncodingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    return "/services";
}
else{
    model.addAttribute("err", "Wrong Password!");
    judge=false;
    return "/login";
}
}
else{
    session.putValue("name", person.getPhone_no());
    if (person.getPhone_no().length()!=11) {
        model.addAttribute("err1", "Mobile no. must be 11 digits.");
        judge=false;
        return "/login";
    }
    else if(this.PersonService.getUser1(person.getPhone_no()).equals(person.getPassword())){
        if(this.PersonService.getUser(person.getPhone_no()).equals(person.getPassword())){
            String str[]=request.getParameterValues("pass");
            if(str!=null && str[0].equals("on")){
                System.out.println("保存了密码:");
                try {
                    this.addCookie(person.getPhone_no(), person.getPassword(), response, request);
                } catch (UnsupportedEncodingException e) {
                    // TODO Auto-generated catch block

```

```
        e.printStackTrace();
    }
}
else{
    try {
        this.addCookie(person.getPhone_no(),"", response, request);
    } catch (UnsupportedEncodingException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
return "/services";
}
else{
    model.addAttribute("err", "Wrong Password!");
    judge=false;
    return "/login";
}
}
}
else{
    return "/services";
}
return "/services";
}
```

图 3-7：核心代码

```
public void addCookie(String userName,String password,HttpServletResponse response, HttpServletRequest request) throws
//创建cookie
Cookie nameCookie = new Cookie(userName, password);
nameCookie.setPath(request.getContextPath()+"/");//设置cookie路径
//设置cookie保存的时间 单位: 秒
nameCookie.setMaxAge(7*24*60*60);
//将cookie添加到响应
response.addCookie(nameCookie);
}
```

图 3-8：记住密码核心代码

3.1.4 查看或搜索课程内容

如下所示,用户可在主页底部实现查看或搜索课程内容,点击课程内容可进入课程详情。



图 3-8：查看或搜索课程内容

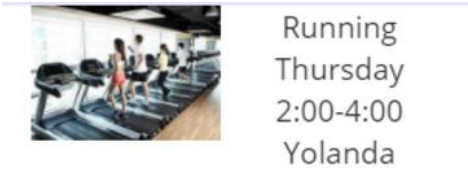


图 3-9：课程详情


```
@Cacheable(value = "courses")
@RequestMapping(value = "/query1/Course", method = RequestMethod.POST)
@ResponseBody
public String course_query(@Valid Course Course,BindingResult bindingResult,Model model) {

    Collection<Course> courses = new ArrayList<Course>();

    Iterable<Course> iterable=this.CourseService.getAllCourses();
    Iterator<Course> iterator1=iterable.iterator();
    while (iterator1.hasNext()) {

        Course c= iterator1.next();
        if(c.getCourse_name().indexOf(Course.getCourse_name())!=-1){
            courses.add(c);
        }
    }
    model.addAttribute("course",courses);

    return "/index";
}

@Cacheable(value = "course_detail", key = "#course_name")
@RequestMapping(value = { "", "/course_detail/{course_name}"})
public String index1(@PathVariable String course_name,Coach coach,Course course, Model model) {
    model.addAttribute("activePage", "contacts");

    Course cour=this.CourseService.getCourse(course_name);
    model.addAttribute("cour", cour);

    return "/index2";
}
```

图 3-10：核心代码

3.1.5 查看或搜索教练

如下所示，用户可在主页底部实现分页查看或搜索教练信息。

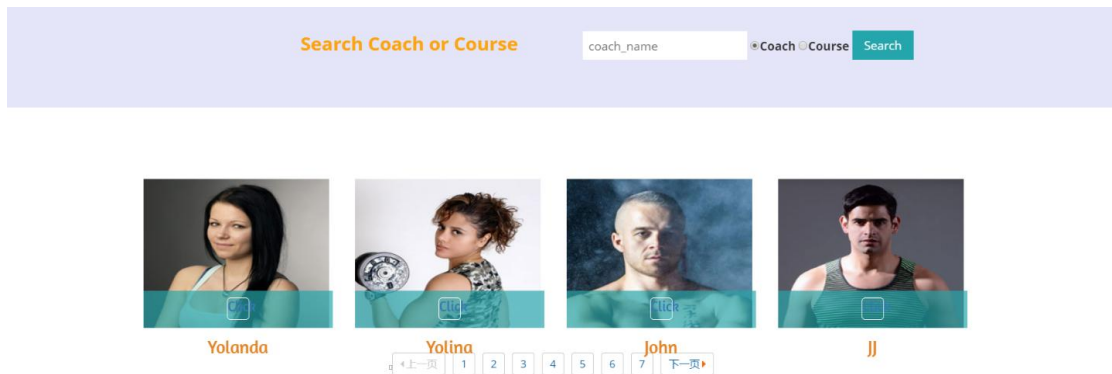


图 3-11：分页查看搜索教练信息

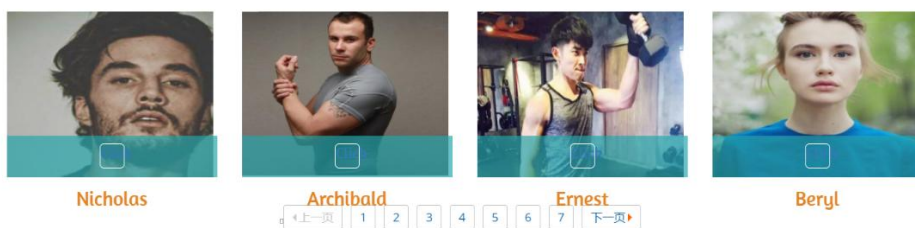


图 3-12：分页的切换

点击教练头像,可查看该教练的具体信息,此处使用多表链接,将数据库 PP1 中的 coach 和 course 进行 join 操作,实现查看该教练所教授的科目的功能。

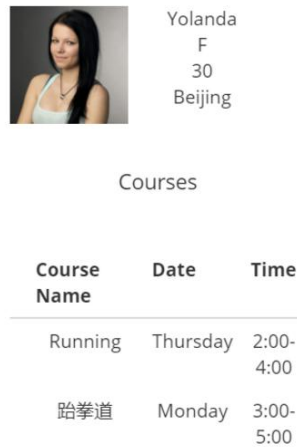


图 3-13: 教练信息

```
@Cacheable(value = "coaches")
@RequestMapping(value = "/query/{page}")
@ResponseBody
public String Coach_query(@Valid Coach coach, BindingResult bindingResult, Model model, @PathVariable("page") int page) {
    Collection<Coach> a = new ArrayList<Coach>();
    PageRequest pageRequest = PageRequest.of(page-1, 4);
    Page<Coach> coachPage = this.CoachService.getAllCoaches(pageRequest);
    if(coach.getCoach_name()==null||coach.getCoach_name().equals("")){
        for (int i = 0; i < coachPage.getContent().size(); i++) {
            Coach c=coachPage.getContent().get(i);
            a.add(c);
        }
    }
    else{
        for (int i = 0; i < coachPage.getContent().size(); i++) {
            Coach c=coachPage.getContent().get(i);
            if(c.getCoach_name().indexOf(coach.getCoach_name())!=-1){
                a.add(c);
            }
        }
    }
    model.addAttribute("coaches",a);
    return "/index";
}

@Cacheable(value = "coach_detail", key = "#coach_name")
@RequestMapping(value = { "", "/coach_detail/{coach_name}"})
public String index1(@PathVariable String coach_name,Coach coach,Course course, Model model) {
    model.addAttribute("activePage", "contacts");
    System.out.println(coach_name);
    Coach co=this.CoachService.getCoach(coach_name);
    model.addAttribute("co", co);

    return "/index1";
}
```

图 3-14: 教练查询页核心代码

```
@ManyToOne
@JoinColumn(name = "coach_id")
private Coach coach;
```

```
@OneToMany(mappedBy = "coach", cascade = CascadeType.ALL, fetch=FetchType.EAGER)
private Set<Course> courses;
public Set<Course> getCourses() {
    return courses;
}
```

图 3-15: 多表链接核心代码

```
PageRequest pageRequest = PageRequest.of(0, 4);
if(coach.getCoach_name()==null||coach.getCoach_name().equals("")){
    Page<Coach> coachPage = this.CoachService.getAllCoaches(pageRequest);
    for (int i = 0; i < coachPage.getContent().size(); i++) {
        Coach c=coachPage.getContent().get(i);
        a.add(c);
    }
}
else{
    Page<Coach> coachPage = this.CoachService.getAllCoaches(pageRequest);
    for (int i = 0; i < coachPage.getContent().size(); i++) {
        Coach c=coachPage.getContent().get(i);
        if(c.getCoach_name().indexOf(coach.getCoach_name())!=-1){
            a.add(c);
        }
    }
}
model.addAttribute("coaches",a);
```

图 3-16: 分页功能核心代码

3.1.5 查看体育馆位置

如下所示，项目调用了百度地图 API，用户可点击主页的“GYM”按钮进入百度地图查看体育馆的位置。

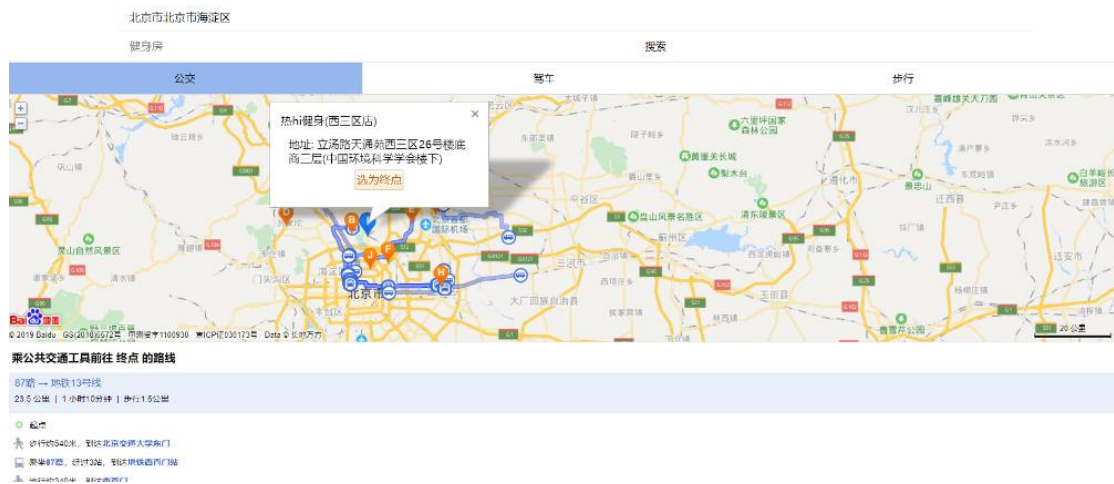


图 3-17: 查看体育馆位置

```
//搜索路线
function searchRoute() {
    if ($("#end").val().length != 0) {
        var end = ($("#end").val());
        $("#end").val("");
    } else {
        var pe = window.end_point.split(",");
        var end = new BMap.Point(pe[0], pe[1]);
    }
    var ps = window.start_point.split(",");
    var start = new BMap.Point(ps[0], ps[1]);
    var starIcon = new BMap.Icon("../images/起点.png", new BMap.Size(22, 32));
    var endIcon = new BMap.Icon("../images/终点.png", new BMap.Size(22, 32));
    if (type == "bus") {
        var transit = new BMap.TransitRoute(map, {renderOptions: {map: map, panel: "r-result", autoViewport: false}});
        transit.search(start, end);
        transit.setMarkersSetCallback(function(result) {
            result[0].marker.setIcon(starIcon);
            result[1].marker.setIcon(endIcon);
        });
    } else if (type == "driver") {
        var driving = new BMap.DrivingRoute(map, {renderOptions: {map: map, panel: "r-result", autoViewport: false}});
        driving.search(start, end);
        driving.setMarkersSetCallback(function(result) {
            result[0].marker.setIcon(starIcon);
            result[1].marker.setIcon(endIcon);
        });
    } else if (type == "walk") {
        var walking = new BMap.WalkingRoute(map, {renderOptions: {map: map, panel: "r-result", autoViewport: false}});
        walking.search(start, end);
        walking.setMarkersSetCallback(function(result) {
            result[0].marker.setIcon(starIcon);
            result[1].marker.setIcon(endIcon);
        });
    }
    //walking.search("天坛公园", "故宫");
}
}
```

图 3-18: 核心代码

3.2 数据库设计

该项目中使用了 MySQL 环境下的两个不同的数据库: pp 和 pp1, 其中 pp 的 person 关系用于存放用户信息; pp1 中的 coach 关系用于存放教练信息, course 关系用于存放课程信息。

```
CREATE TABLE `person` (
  `nick_name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `password` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `email` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `phone_no` varchar(11) NOT NULL,
  `address` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `id` int(11) NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

nick_name	password	email	phone_no	address	id
123	123456	2@163.com	666666666666	123	5
123	333333	2@bjtu.edu.cn	333333333333	4	6
123	123456	1@bjtu.edu.cn	444444444444	1	7
123	123456	0@bjtu.edu.cn	000000000000	3	8
Yolanda	098765	16301076@bjtu.edu.cn	15801625826	北京	9
Yolanda	098765	16301076@bjtu.edu.cn	15801625826	北京	10
jj	123456	211@bjtu.edu.cn	15801625820	北京	11
QinYiYi	123456	2@163	12345678901	beijing	12

图 3-19: person 关系表及其约束

```
CREATE TABLE `coach` (
  `coach_name` varchar(255) NOT NULL,
  `sex` varchar(255) NOT NULL,
  `age` int(11) NOT NULL,
  `address` varchar(255) NOT NULL,
  `id` int(11) NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

coach_name	sex	age	address	id
Yolanda	F	30	Beijing	1
Yolina	F	27	Beijing	2
John	M	25	Beijing	3
JJ	M	20	Beijing	4
Nicholas	M	35	Haidian	5
Archibald	M	26	Haidian	6
Ernest	M	23	Haidian	7
Beryl	F	26	Beijing	8
Paula	F	26	Beijing	9

图 3-19: coach 关系表及其约束

```
CREATE TABLE `course` (
  `course_name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci DEFAULT NULL,
  `date` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci DEFAULT NULL,
  `time` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci DEFAULT NULL,
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `coach_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

course_name	date	time	id	coach_id
Running	Thursday	2:00-4:00	2	1
跆拳道	Monday	3:00-5:00	3	1
器械健美	Tuesday	5:00-7:00	4	2
健美操	Friday	3:00-5:00	5	2

图 3-19: course 关系表及其约束

3.3 缓存功能

该项目采用 Redis 3.2.100 实现缓存功能，由于涉及的代码过于分散，此处便不另外贴出核心代码，具体实现可参见项目源码。

```
C:\Users\Lenovo\Documents\Downloads\Redis-x64-3.2.100\redis-server.exe
[7752] 30 Apr 09:02:11.412 # Warning: no config file specified, using the default config. In order to specify a config file use C:\Us
loads\Redis-x64-3.2.100\redis-server.exe /path/to/redis.conf

Redis 3.2.100 (00000000/0) 64 bit

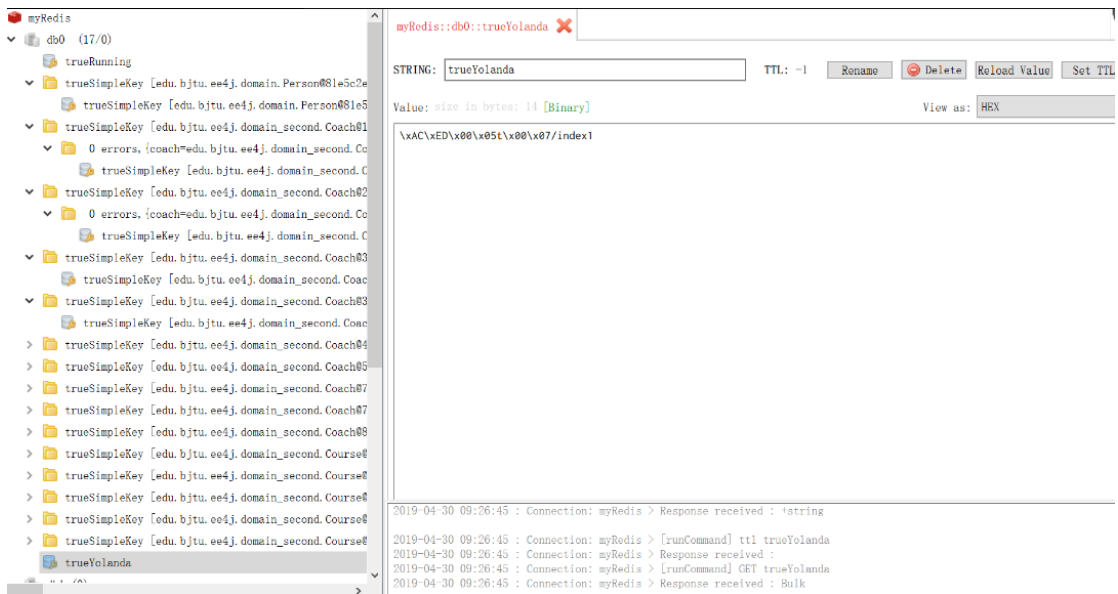
Running in standalone mode
Port: 6379
PID: 7752

http://redis.io

[7752] 30 Apr 09:02:11.418 # Server started, Redis version 3.2.100
[7752] 30 Apr 09:02:11.419 * DB loaded from disk: 0.000 seconds
[7752] 30 Apr 09:02:11.419 * The server is now ready to accept connections on port 6379
[7752] 30 Apr 10:02:12.055 * 1 changes in 3600 seconds. Saving...
[7752] 30 Apr 10:02:12.121 * Background saving started by pid 5492
[7752] 30 Apr 10:02:12.424 # fork operation complete
[7752] 30 Apr 10:02:12.426 * Background saving terminated with success
[7752] 30 Apr 11:02:13.003 * 1 changes in 3600 seconds. Saving...
[7752] 30 Apr 11:02:13.074 * Background saving started by pid 6648
[7752] 30 Apr 11:02:13.275 # fork operation complete
[7752] 30 Apr 11:02:13.276 * Background saving terminated with success
[7752] 30 Apr 12:02:14.014 * 1 changes in 3600 seconds. Saving...
[7752] 30 Apr 12:02:14.103 * Background saving started by pid 1608
[7752] 30 Apr 12:02:14.305 # fork operation complete
[7752] 30 Apr 12:02:14.306 * Background saving terminated with success
```

激活
转到

图：Redis 运行截图



图：Redis 数据可视化