文章编号:1007 - 2985(2013)05 - 0031 - 06

基于 Cloud Foundry 云平台的网络考试系统实现*

丁振凡
(华东交通大学信息工程学院,江西 南昌 330013)

摘 要:介绍了基于 Cloud Foundry 云平台的网络考试系统设计. 在 STS 环境下通过添加“Cloud Foundry Integration”的扩展插件,开发者可通过虚拟机实现对 Cloud Foundry 云平台应用的可视化部署与管理. 应用借助云服务可访问云上 MySQL 数据库. 重点围绕考试系统的组卷、试卷显示、交卷评分处理等环节,介绍了用 Spring MVC 框架设计相应的控制器、模型及视图显示的编程处理方法. 系统采用 Json 串实现试卷信息在页面之间传递处理.

关键词:Cloud Foundry;云数据库访问;考试系统;MVC 开发模式
中图分类号:TP393 **文献标志码**:A **DOI**:10.3969/j.issn.1007 - 2985.2013.05.008

云计算作为一种新型的资源共享和管理模式,成为近年来研究和应用的热点[1]. 将软件部署在云的虚拟化环境中,开发人员专注于应用设计,而不用维护基础设施,可使软件部署变得快速方便. Cloud Foundry 是 VMware 发布的业内第 1 个开源 PaaS 项目. Cloud Foundry 云平台提供了软件开发、测试和运行维护一整套环境,确保安全性、可管理性、高可用性. 该系统通过给 STS(Springsource Tool Suite)添加“Cloud Foundry Integration Extension”,实现在 STS 开发环境下对云上应用的部署与调试.

网络考试是网络教学平台中较为复杂的一项功能[2]. 完整的考试系统应支持较丰富的题型. 系统采用 Spring MVC 模型进行开发,系统功能由 Spring 的控制器、模型、服务业务逻辑、视图协作完成[3]. 为简单起见,笔者仅讨论组卷、试卷显示、解答评分处理的实现,且题型仅考虑单选题和多选题.

1 通过 STS 实现云应用开发部署过程

1.1 利用 STS 环境云虚拟机部署应用

在 STS 环境,是依托云虚拟机实现应用的部署和管理. 为了连接云服务器,第 1 步,在 http://my.cloudfoundry.com/signup 站点注册一个帐户;第 2 步,在 STS 环境中添加“Cloud Foundry Integration”的插件;第 3 步,通过 STS 的 Server 添加功能,添加对应云虚拟机的服务器,连接服务器将需要注册的帐户和密码;第 4 步,应用运行时选择云服务器,就可以自动将应用部署到云环境.

1.2 云上 MySQL 数据库的使用配置

在 Cloud Foundry 云上支持 MySQL 服务,应用中可使用云的 MySQL 数据库实现数据存储. 首先,要添加 Mysql 服务,双击 Server 窗体的 Cloud Foundry server,在其控制面板中,点击“add service”图标,在弹出的对话框中,选择数据源的类型和输入服务的名称,点击“Finish”按钮即可. 然后,将 Services 面板的

* **收稿日期**:2013 - 05 - 17
作者简介:丁振凡(1965 -),男,江西丰城人,华东交通大学信息工程学院教授,硕士生导师,主要从事云计算、语义 Web、计算机辅助教学研究.

数据源服务拖到 Application Services 面板. 这样,在应用中才能通过标识连接到该数据源.

系统采用 Spring 框架进行开发,系统开发的技术框架如图 1 所示,用 Spring Security 进行用户认证设计[4]. 在应用配置文件中引用云上定义的数据源,要用到 cloud 名空间的标签. 以下配置代码引用云上标识为"mysql"的数据源,并根据该数据源建立 JdbcTemplate 对象[5],将与数据源连接的 jdbcTemplate 定义为一个 Bean:

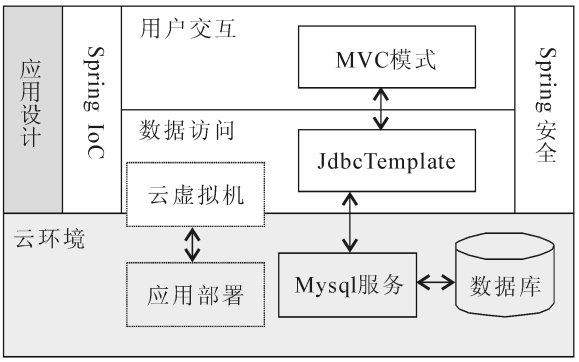

图 1 系统开发环境及技术

```
<cloud:data-source id="mysql" />
<beans:bean id="jdbcTemplate"
  class="org.springframework.jdbc.core.JdbcTemplate">
    <beans:property name="dataSource" value="#{mysql}" />
</beans:bean>
```

其他 Bean 要访问数据库可通过属性依赖引用 jdbcTemplate 即可.

连接云数据库后,需要编写相应的程序执行 SQL 语句建立表格,并通过程序将数据导入到表格中. 文中程序所涉及的各表的字段含义解释如下:

(1) 单选题表(danxuan)、多选题表(mxuan)结构相同,含字段有 number 为题号,content 为试题内容,diff 为难度,knowledge 为所属知识点,answer 为答案.

(2) 考试参数配置表(config)含字段有 knowledges 为考核知识点的集合,sxamount 为单选题的数量,sxscore 为单选题的小题分数……. 其中,knowledges 为一个文本串,列出所有考核知识点,每个知识点用单引号括住,知识点之间用逗号分隔.

1.3 Cloud Foundry 上 Web 应用的工程要求

Cloud Foundry 不支持动态 Web 工程,需要利用 Maven Web 工程或者利用 Spring MVC 模板工程来创建工程. 在工程的依赖管理项目中,除了添加 Spring 框架、Spring 安全框架、AOP 以及 Web 应用所需典型 Jar 包依赖项外,为支持 Cloud Foundry,还需要在工程的 pom.xml 文件中添加工件标识为"cloudfoundry-runtime"的依赖项,并在 pom.xml 文件中添加 Spring 框架的 Milestone 远程仓库,该仓库含有 cloudfoundry 运行时的 jar 包.

2 组卷处理及试卷显示设计

在试卷处理不同阶段,需要试卷的不同信息. 例如:组卷阶段只要记录大题类型、各小题编号;显示试卷时则需要大题名称、各小题的内容;评卷阶段只要大题的每小题分值、各小题答案. 因此,应用设计中对试卷信息进行各自的封装设计,试卷传递只传递基本信息,其他信息可根据基本信息查阅数据库得到.

实现试卷在应用各功能之间传递有多种方法,例如用 session 对象、用 Cookie 变量. 该系统选择采用 URL 参数传递,其好处是不用消耗客户方和服务方的资源. 但用 URL 传递试卷对象需要将对象转换为字符串,否则,对象不能直接作为 URL 参数. 该系统是用 Google 的 Gson 工具实现对象与 Json 串的变换[6]. 另外,还需要对变换后的内容进行 URL 编码处理.

2.1 组卷相关数据对象的封装设计

定义以下类用来封装记录组好的试卷的相关信息. 整个试卷由若干题型构成,每个题型由若干试题构成. 系统假定每道大题的各小题分配相同分值.

(1) 用 Question 类记录下某类题型的抽题信息. 包括题型编码、每小题分值以及抽到的试题编号构成的数组.

```
public class Question {
    int bh[]; //各小题编号
    int score ; //每小题分数
```

```
    int type; //题型,值为 1 表示单选,为 2 表示多选
}
```

(2) 用 ExamPaper 类中记录整个抽取的试卷. 用 ExamPaper 封装整个试卷是为了方便后面的 Json 包装处理. 将 Json 串转换为对象时,可通过 ExamPaper 类指示要转换的目标.

```
public class ExamPaper {
    List<Question> allst = new ArrayList<Question>();    //存放各类大题的抽题信息
}
```

2.2 组卷业务逻辑程序

根据组卷参数要求,组卷程序从数据库抽取试题组卷. 在类 ExamPaper 中设计了 2 个方法. genPaper 方法将根据数据库存储的组卷参数要求进行组卷,它会调用 pickst 方法实现具体某个题型的抽题处理.

```
public static ExamPaper genPaper(JdbcTemplate jdbcTemplate){
    ExamPaperpaper=new ExamPaper();
    String sql="select knowledges from config"; //查考核知识范围
    String knowledges=jdbcTemplate.queryForObject(sql,String.class);
    sql="select sxamount from config";  //查配置得到单选数量
    int amount=jdbcTemplate.queryForInt(sql);
    if (amount>0) {
      sql="select sxscore from config";  //查配置得到每小题分数
      int score=jdbcTemplate.queryForForObject (sql,Integer.class);
      Question q=new Question();
      q.type=1;  //单选题型
      q.score= score;
      q.bh= pickst(jdbcTemplate,"danxuan",amount,knowledges); //组卷
      paper.allst.add(q); //将单选题的组卷选题加入试卷中
    }
      ……  //其他类试题的选题略
    returnpaper;
}
```

pickst 方法在知识点范围随机选题,使用 SQL 的 in 关键词选取,这里未考虑难度要求. 算法可自动适应课程的实际试题数量,数量不足时按实际数量选取. 方法的参数包括数据库表格名、选题数量、知识点范围等,方法的返回结果为选中试题编号构成的数组. 限于篇幅,该方法代码略.

2.3 组卷 MVC 控制器

组卷控制器将调用组卷算法完成组卷,并设置试卷显示视图需要的模型参数. 为方便考卷的解答显示处理,引入类 DisplayPaper 封装试卷显示所需的信息. 考虑到既要传递组卷给后续页面,又要显示试卷,在模型中分别用 paper 和 disppaper 属性记录组卷和显示试卷内容. 因为传递给视图的 paper 要通过在后续页面中用表单的 URL 传递,所以除了要进行串行化处理外,还需要进行 URL 编码处理. 另一种办法是采用表单的隐含域传递,则可以不必进行 URL 编码处理.

```
public classDisplayPaper {
    List<String> content = new ArrayList<String>(); //试题文本内容
    String name; // 该类试题名称
    int type;// 试题类型
    int score;//小题分值
}
```

以下为 ExamController 控制器的 mapping 方法:

```
@RequestMapping(value = "/disppaper",method = RequestMethod.GET)
public String display(Model model,HttpServletRequest request) {
    ExamPaper paper= ExamPaper.gen(jdbcTemplate); //调组卷算法组卷
```

```
    int len = paper.allst.size();
    DisplayPaper[] disp =new DisplayPaper[len];
    for (int k=0;k<len;k++) {
      Question q= paper.allst.get(k);//第 k 个题型
      disp[k]=new DisplayPaper();
      disp[k].type=q.type;
      disp[k].name=getTxName(q.type); //获取题型名称
      disp[k].score=q.score;
      for (int i=0;i<q.bh.length;i++) //获取此类试题的各题内容
        disp[k].content.add(getContent(q.bh[i],q.type));
    }
    Gson gson = new Gson(); //使用 Google 的 Gson 将试卷包装为 Json 串
    String jsonpaper="";
    try { jsonpaper =URLEncoder.encode(gson.toJson(paper),"utf-8");
    } catch (UnsupportedEncodingException e) {}
    model.addAttribute("paper",jsonpaper);//传递 Json 试卷用于判分处理等
    model.addAttribute("disppaper",disp);//用于显示的试卷
    return "display";
  }
```

其中,在控制器的逻辑中还用到 2 个私有方法:一个是 getTxName,根据题型 type 得到题型的文字描述,可以根据题型数量扩展;另一个是 getContent,根据题型和试题编号得到试题内容.

2.4 试卷显示视图(display.jsp)

视图文件给出试卷的显示模板,试卷显示除了解决试卷内容的显示外,还需要提供用户解答控件,如图 2 所示. 这里用户解答控件的命名按“data”+大题号+“-”+小题号的拼接方式. 程序中在显示处理上使用了一些技巧,包括通过 EL 变量实现试题序号的显示,用 JSTL 的 forTokens 标签实现解答选项的输出,使用 JSTL 的函数库中 substring 方法提取文字串的中文数字等.

图 2 试卷解答界面

```
<%@page contentType="text/html; charset=UTF-8"%>
<%@ taglib uri=" http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<html><body><c:set value="一二三四五六" var="s" />
<form action="givemark/${paper}" method=post>
<c:set value="1" var="k" />
<c:forEach items="${disppaper}" var="st">
<font size=4 face="黑体" color=red>${fn:substring(s,k-1,k)}.${st.name}</font>
<font size=3 face="宋体" color=green>(每小题${st.score}分)</font> <br>
<c:set value="1" var="x" />
<c:forEach items="${st.content}" var="content">
<table width=98% align=center style="word-break:break-all"><tr>
<td align=left valign=top width=20><b><font color=blue>${x}.</font></b></td>
<td> <pre><font >${content}</font></pre></td></tr></table>
<table width=50% align=center><tr>
<c:if test="${st.type<=2 }">
```

```
<c:forTokens items="A,B,C,D,E"delims=","var="item" >
<td align=right> ${item} </td> <td align=left>
  <c:choose>
    <c:when test=" ${st.type==1}">
      <input type=radio size="30" name="data${k}-${x}" value= ${item}>
    </c:when>
    <c:when test=" ${st.type==2}">
      <input type=checkbox size="30" name="data${k}-${x}" value= ${item}>
    </c:when>
  </c:choose>
</td>
</c:forTokens>
</c:if>
</table>
<c:set var="x" value=" ${x+1}"/>
</c:forEach>
<c:set var="k" value=" ${k+1}"/><br>
</c:forEach>
<p align="center"><input type="submit" name="button" value="交 卷 " >
</form></body></html>
```

3　阅卷处理设计

3.1 阅卷方法设计

阅卷处理可以不必引入额外类记录试卷信息,而是在前面组卷传递的试卷信息的基础上进行处理,只是各小题的答案要根据试题编号和题型从数据库获得.评阅某个大题时,可以将小题标准答案放入数组中,与用户输入解答构成的数组元素逐个比较.

在 ExamPaper 类增加一个 givescore 方法对阅卷逻辑进行封装.givescore 方法的参数有题型、标准答案、学生解答、小题分数,它将对某类题型的解答进行判分.方法的返回结果为一个含 2 个元素的数组,分别为某个题型的学生得分和总分.

3.2 阅卷控制器的逻辑

阅卷控制器是与用户的接口,控制器通过 REST 风格的服务处理用户提交的请求.用户通过页面表单提交解答,并通过 URL 参数传递用 Json 串封装的试卷.阅卷控制器工作包括解开 Json 试卷,对试卷的各大类试题循环处理,通过 HttpServletRequest 对象的 getParameter("控件名")方法得到学生的解答.将解答与根据试题编号获取的答案进行比对评分,并计算总得分.最后登记学生的考卷和解答.

3.3 学生得分显示视图

得分显示视图中用 EL 表达式获取来自模型的分数,经 Javascript 脚本弹出对话框显示学生得分,并通过执行页面重定向将页面导向到系统首页,防止学生回退[7],从而避免反复交卷试出答案.

4　结语

介绍了一个采用 Spring MVC 设计的网络考试系统,用 maven 工程构建应用,数据存储采用来自云的 mysql 数据库.重点讨论了组卷、解答试卷、评阅试卷等环节所涉及的试卷信息的模型设计以及视图显示处理方法,实际系统中还提供了学生查卷等功能,并提供了更为多样的题型,限于篇幅未展开.该系统已部署在 Cloud Foundry 云环境,在 Java 等课程的考试中应用,希望对基于 Cloud Foundry 云环境的 Web 数据库应用开发有所参考.

参考文献:

[1] 林 利,石文昌.构建云计算平台的开源软件综述 [J].计算机科学,2012,39(11):1-6.

[2] 丁振凡.基于 Spring 的网络考试系统的服务设计 [J].吉首大学大学学报:自然科学版,2013,34(1):21-25.

[3] 丁振凡.用 Spring MVC 实现数据分页显示处理 [J].智能计算机与应用,2012,2(5):20-22.

[4] 丁振凡.基于 Spring Security 的 Web 资源访问控制 [J].宜春学院学报,2012,34(8):71-74.

[5] 丁振凡,李馨梅.基于 JdbcTemplate 的数据库访问处理 [J].智能计算机与应用,2012,2(3):29-32.

[6] 丁振凡.Spring REST 风格 Web 服务的 Json 消息封装及解析研究 [J].智能计算机与应用,2012,2(2):9-11.

[7] 丁振凡.Web 编程实践教程 [M].北京:清华大学出版社,2011:261-262.

Implementation of Network Examination System Based on Cloud Foundry Platform

DING Zhen-fan

(School of Information Engineering, East China Jiaotong University, Nanchang 330013, China)

Abstract: The design of network examination system based on Cloud Foundry platform is introduced. By adding a "Cloud Foundry Integration" extension in the STS environment, developers can visually deploy and manage the applications in Cloud Foundry platform through the virtual machine. By using cloud services, application can access the MySQL database on the cloud. Focusing on test paper generation, test paper display, and scoring process in the examination system, the author introduces programming methods about the controller, model and view based on Spring MVC framework. The system uses the Json string to transfer test paper information between web pages.

Key words: Cloud Foundry; access to Cloud database; examination system; MVC development model

(责任编辑 向阳洁)

(上接第 10 页)

Reference:

[1] KRASNOSELSKII M A, POKROVSKII A V. Systems with Hysteresis [M]. Moscow: Nauka, 1983.

[2] PAVEL KREJCI. Hysteresis and Periodic Solutions of Semilinear and Quasilinear Wave Equations [J]. Mathematische Zeitschrife, 1986, 193: 247-264.

[3] PAVEL KREJCI. Periodic Solutions of a Parabolic Equation with Hysteresis [J]. Mathematische Zeitschrife, 1987, 194: 61-70.

[4] MAYERGOYZ I D. Mathematical Models of Hysteresis [M]. New York: Springer-Verlag, 1991.

[5] LADYZENSKAAJA O A. Linear and Quasi-Linear Equations of Parabolic Type [M]. American: The American Mathematical Society, 1968: 204-210.

一个具 Hysteresis 的电报方程

徐龙封[1],薛 亮[1,2]

(1.安徽工业大学数理学院,安徽 马鞍山 243002;2.安徽工业职业技术学院基础部,安徽 铜陵 244000)

摘 要:讨论一个具 Hysteresis 的电报方程.首先给出 Ishlinskiĭ hysteresis 算子的一些性质,然后得到这个方程 $C^{1+\alpha,\alpha}$ 强解存在性.

关键词:Ishlinskiĭ hysteresis 算子;电报方程;单调算子;强解

中图分类号:O175.29 文献标识码:A

(责任编辑 向阳洁)