文章编号:1671-4598(2020)04-0185-06 DOI:10.16526/j.cnki.11—4762/tp.2020.04.038 中图分类号:G647 文献标识码:A

基于 AOP 实现冲突动态检测的实验室预约系统设计

张 亮[1],李正卫[1],蒋 烨[2]
(1.浙江工业大学 管理学院,杭州 310023;2.浙江工业大学 经济学院,杭州 310023)

摘要:在实验室预约管理工作中,手工操作效率低、出错率高,而使用成品软件难以进行功能扩展与二次开发;针对这种情况,以浙江工业大学经济管理实验教学中心为背景,利用面向切面(aspect—oriented programming,AOP)技术实现能够动态检测预约冲突并满足实际应用需求的新型实验室预约系统;系统基于 MVC 设计模式,通过异常处理与事务管理机制检测预约冲突,并在运行时将检测程序以 AOP 切面方式嵌入业务处理流程中,对预约操作实行动态拦截,实现预约冲突的自动化检测与处理;系统提供线上预约与信息管理服务,便于功能扩展与数据对接,其开发技术具有通用性,对优化架构设计提升系统灵活性有一定的借鉴作用。

关键词:实验室预约;MVC 模式;面向切面设计;冲突动态检测;架构优化

Design of Laboratory Reservation System for Dynamic Conflict Detection Based on AOP

Zhang Liang[1], Li Zhengwei[1], Jiang Ye[2]
(1. School of Management, Zhejiang University of Technology, Hangzhou 310023, China;
2. School of Economics, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract: In the management of laboratory reservation, manual operation is inefficient and with high error rate, besides it is difficult to expand and redevelop functions by using existing software. In view of this situation, a new type of laboratory reservation system, which can dynamically detect reservation conflicts and meet the practical application needs, is implemented by using AOP (aspect oriented programming) technology in the experimental center of economics and management in Zhejiang university of technology. Based on MVC design pattern, the system detects reservation conflicts through exception handling and transaction management mechanism, and embeds the detection program into the business process using AOP at runtime. It dynamically intercepts the reservation operation to realize automatic detection and processing of reservation conflicts. The system provides on—line reservation and information management services, facilitates function expansion and data docking. Its development technology is universal, and can be used for reference to optimize the architecture design and enhance the flexibility of the system.

Keywords: laboratory reservation; MVC; AOP (aspect—oriented programming); dynamic conflict detection; optimized design

0 引言

随着高校经管类各专业开设的实验课程越来越多,实验教学所占的比重逐渐增大,如何根据课程需要合理调配有限的实验室资源,成为当前经管类实验室管理工作亟需解决的重要问题。传统使用 EXCEL 电子表格记录实验室预约信息的方式存在操作效率低、工作量大、出错率高等问题,经常造成课程时间冲突,上课人数与实验室机位数量不符,实验室软硬件配置与教师授课需求不符,极大影响了教师与学生的课堂教学体验。而购买厂商的相关软件产品由于其价格昂贵,后期维护成本较高,同时又难以进行个性化定制与功能扩展,无法适应各个高校在实验室管理工作方面的实际需求。

通过总结工作经验,分析存在的问题,提出利用软件工程领域的前沿技术设计与开发具有较好的可扩展性的新型经管类实验室预约系统,优化实验教学管理模式,方便教师在线进行实验室预约,同时允许师生对实验室硬件配置、软件资源、数字资料、面向专业与课程安排等信息进行实时查询。

考虑到与教务处、人事处系统进行对接,实现数据的同步与共享,以及后期的管理维护与功能扩展等需要。采用 MVC 设计模式,将系统划分为模型、视图、控制器三层进行分层开发,实现数据与操作之间的解耦,提高程序代

收稿日期:2019-09-07; **修回日期**:2019-10-10。
基金项目:浙江省高等教育"十三五"教学改革研究项目(jg20180048);浙江工业大学创新性实验项目(syxm1725);浙江工业大学创新性实验项目(syxm1726)。
作者简介:张 亮(1983-),男,浙江杭州人,硕士,实验师,主要从事软件工程、Web 开发技术、数据挖掘方向的研究。
李正卫(1970-),男,江苏江都人,博士,教授,博士生导师,主要从事创新管理方向的研究。

码的复用性与可扩展能力，满足松散耦合的设计标准[1-2]。

由于每学期的课程安排比较紧凑，教师与学生人数较多，经常会出现课程时间安排冲突导致教学资源浪费的情况发生[3-4]。为解决这个问题，实现预约冲突的智能化检测与处理[5-6]，在 MVC 分层开发的基础上引入面向切面（aspect－oriented programming，AOP）设计思想，将冲突检测程序作为通用功能从主要的业务流程中抽离出来进行模块化封装。在系统运行期，根据实际需要，通过预编译和动态代理的方式，将冲突检测程序自动切入当前执行的业务处理过程中，对预约操作实行动态拦截。这种设计方式能够有效实现逻辑功能的分离与解耦，避免各模块相互之间的影响与制约，从而进一步降低程序耦合度，实现系统化、智能化管理。

1 系统架构设计及原理

1.1 MVC 设计模式

MVC 设计模式（Design pattern），即模型、视图、控制器（Model、View、Controller），是将应用程序的业务逻辑层、视图显示层进行分层设计，并通过控制器层进行连接调度的一种开发模式，最早由 Trygve Reenskaug 提出，为施乐帕罗奥多研究中心（Xerox PARC）的 Smalltalk 面向对象编程语言所采用的一种开发模式。其意义在于实现用户界面与业务逻辑的分离，提高程序代码的灵活性与复用率[7]。

模型层（Model）表示业务数据与逻辑规则，用于实现数据操作与业务逻辑功能。模型层通过与数据库进行交互，实现数据的读取及写入等操作。其相对于数据来说保持中立，即与数据格式及数据库类型无关，无论使用哪一种数据库，都返回相同格式的数据，实现数据与表现的解耦。

视图层（View）即用户显示界面，对 Model 返回的数据进行格式化显示输出，也用于收集用户输入信息。视图层通过浏览器（Web Browser）提供与用户进行互动交流的界面，是系统业务逻辑与用户之间沟通的桥梁。视图层的组成元素包括：HTML、JavaScript、CSS、XML、Web Services 等。

控制器层（Controller）负责协调模型与视图，即根据 HTTP 请求 URL 映射获取当前需调用的业务逻辑方法，操作完成后选择相应的视图，通过渲染视图对操作返回的数据进行格式化，最后在客户端浏览器中显示输出，完成用户请求。

1.2 AOP 面向切面设计

AOP 面向切面设计模式，即 Aspect－Oriented Programming，将通用功能从业务逻辑程序中分离出来，对其进行独立编码实现；在系统运行时，将独立程序动态切入到当前操作对象的方法执行过程中，实现业务功能的单独管理与动态组合。

AOP 模式允许不同业务处理程序共享相同的行为，实现应用程序各部分之间低耦合的分离效果，业务逻辑程序的改变不影响以 AOP 切面进行封装的通用功能，从而进一步提高代码的复用率，降低程序之间的耦合度。

AOP 模式涉及到的相关概念包括切面（Aspect）、连接点（JointPoint）、处理逻辑（Advice）和切点（Pointcut）。具体含义如下：

1）切面（Aspect）：是将业务程序中共同的、重复的部分进行横向切分并单独实现，实现 cross－cutting 功能。

2）连接点（JointPoint）：是切面嵌入业务流程的触发点，可以是在抛出异常时，方法调用时，或者修改某个变量时插入切面代码，执行新的行为。当程序正常流程执行到切面连接点时，自动调用相应的处理逻辑（Advice）。

3）处理逻辑（Advice）：用于实现切面功能，在 JointPoint 处插入到业务处理流程中，并告知程序有新的行为将被执行。

4）切点（Pointcut）：用于控制在 JointPoint 上被调用的 Advice。

经管类实验室预约系统将预约冲突检测程序以 AOP 面向切面的方式嵌入业务处理过程中。在执行实验室预约操作前，系统通过依赖注入方式动态切入并调用检测程序，利用事务管理和异常处理机制，实现预约冲突的自动检测，如图 1 所示。

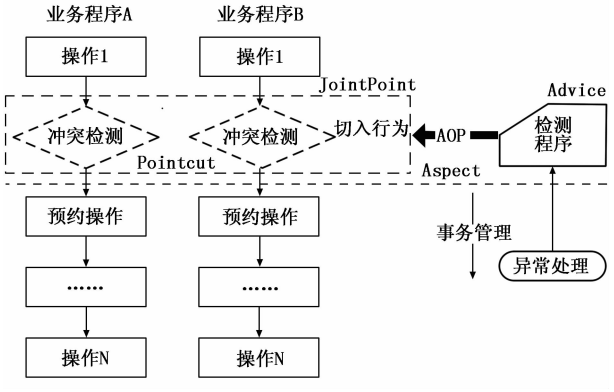

图 1 AOP 面向切面设计

将冲突检测功能从主程序中分离出来，能够在不修改业务流程的情况下，单独对检测程序进行修改与补充。这样，在开发过程中能够实现合理分工，各功能模块职责明确，避免了相互之间的影响与制约，降低维护成本，提高开发效率。

1.3 基于 YII2 MVC 架构的 AOP 模式

系统基于 YII2 MVC 架构搭建，在 MVC 分层结构的基础上引入 AOP 面向切面设计理念，其具体实现方法是在控制器（Controller）与模型（Model）之间加入过滤器（Filter）验证功能，当调用 Action 方法执行预约操作时，先触

发行为（Behavior），在行为程序中执行过滤器验证，即检测预约冲突，检测通过则继续执行 Action，完成预约操作；检测未通过则停止执行 Action，将冲突信息返回给视图。基于 YII2 MVC 架构的 AOP 模式如图 2 所示。

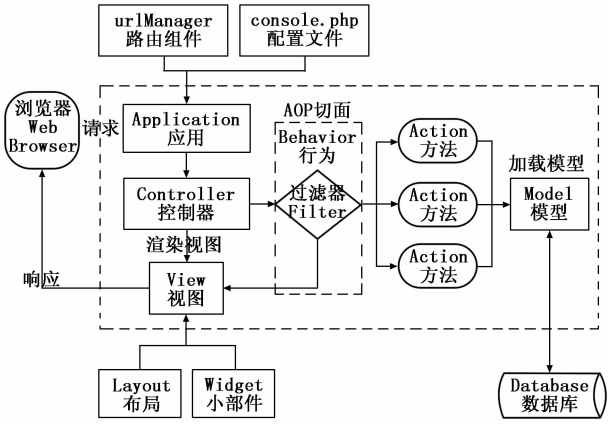

图 2　基于 YII2 MVC 架构的 AOP 模式

行为（Behavior）是 yii \ base \ Behavior 类及其子类的实例化对象，表示方法执行过程中自动执行的功能程序。使用行为前，需要先将其与组件进行绑定，当组件运行时，行为将其自身所包含的属性与方法通过动态注入的方式附加到组件上，使得在组件中运行行为就像在执行组件自己的方法一样。行为通过与组件的绑定能够对触发事件（Event）做出响应，从而实现对组件运行流程的动态调整。

过滤器（Filter）是在 Action 方法执行前后运行的程序，通过行为（Behavior）进行调用，常用于配置控制器权限（即 RBAC － Role－Based Access Control 权限管理）、客户端/页面缓存、用户认证、内容格式检测、HTTP 请求方式验证、跨域资源共享等操作。过滤器包括预切入过滤器（在 Action 方法执行前运行的程序）、后切入过滤器（在 Action 方法执行后运行的程序）。在行为中可以部署多个过滤器，并分别将其与不同的 Action 方法进行绑定。过滤器的精心设计与合理利用能够有效提升系统开发的灵活性，通过将多个功能封装在不同的过滤器中，实现通用功能与主要业务流程之间的解耦，提高代码复用率。

架构运行流程如下：

1）用户通过浏览器发送请求，交由 web/index. php 入口程序处理，在入口程序中加载 console. php 应用配置文件，根据配置信息创建 Application 实例。

2）Application 实例通过 urlManager 路由组件对请求 URL 进行解析，根据解析结果定位目标 Controller 类与 Action 方法，同时创建 Controller 实例，Controller 实例调用 runAction 方法执行 Action。部分实现代码如下：

```
//创建 Controller 实例
parts = this－>createController(route);
if (is_array(parts)) {
list(controller, actionID) = parts;
Yii::app－>controller = controller;
//执行 Action
result=controller－>runAction(actionID, params);
...
}
```

3）当 Controller 实例调用 runAction 方法时，会首先执行 beforeAction 方法，根据 beforeAction 的返回值决定是否继续执行当前的 Action。

4）在 beforeAction 内，通过调用 ensureBehaviors 方法将 Behavior 行为绑定到 Controller 实例上，这样在执行 Action 前，会先触发 Behavior 行为，执行在行为中部署的过滤器（Filter）。若某个过滤器返回 false，则取消执行 Action。当所有过滤器均验证通过，则继续执行 Action。ensureBehaviors 方法实现代码如下：

```
public function ensureBehaviors() {
if (this－>_behaviors === null) {
this－>_behaviors = [];
foreach (this－>behaviors() as name => behavior) {
//绑定 Behavior 行为
this－>attachBehaviorInternal(name, behavior);
}
}
}
```

5）过滤器验证通过后，执行 Action 方法，在 Action 中加载模型（Model），通过 DAO 数据库访问层对 MySQL 数据库进行 CRUD 操作，完成业务逻辑处理操作。

6）Controller 实例调用 render 方法渲染视图（View），即将数据处理结果传递给视图，结合 Layout 布局文件与 Widget 小部件构成完整的页面。调用 render 方法的代码如下：

```
return this－>render('index', ['param' => '...']);
```

7）在 Response 响应对象中封装视图渲染结果，并以 HTML 格式返回给浏览器客户端。

2　系统运行环境

系统采用 WAMP 集成环境，实现 APACHE、MySQL、PHP 的高效整合，摆脱环境配置的烦恼，提高开发效率。WAMP 集成 PHPMYADMIN 数据库管理工具，允许管理者直接通过 Web 接口对 MySQL 数据库进行操作与管理，无需安装其他客户端软件。开发工具选用 EDITPLUS 5.0，EDITPLUS 是一款轻量级且功能强大的文本编辑器，支持语法高亮、代码折叠等功能，具有启动速度快、界面简洁等特点，能够提高程序代码的编写效率。

为提升系统性能，提高运行平台与 Windows Server 操作系统的兼容性，采用 IIS7.5 作为系统运行平台，通过 FastCGI 方式配置 PHP，使得应用系统在 IIS 平台上进行部署的同时，又能够通过 WAMP 进行配置与管理。

3 系统功能设计

经管类实验室预约系统的功能模块如图 3 所示，主要分为实验室管理模块、时间管理模块、课程管理模块、教师管理模块、实验室预约模块、冲突检测模块与数据同步模块[8]。

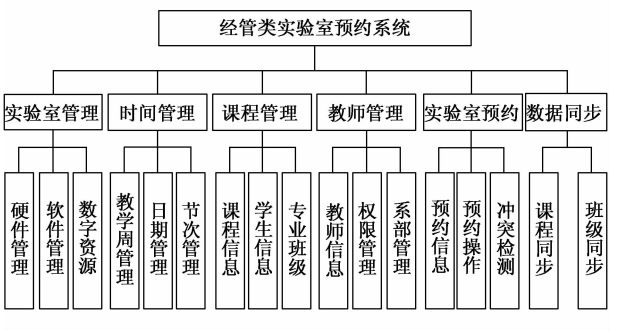

图 3 系统功能设计

1）实验室管理模块，包括对实验室硬件配置、软件资源、数字资源、机位数量与当前使用情况等信息的查询与管理，便于师生随时随地地了解实验室相关信息，有利于实验教学课程的顺利进行。MySQL 数据库中提取的数据信息通过 Bootstrap 前端技术进行栅格式与响应式的布局排版，以适应手机、平板等移动便携设备。

2）时间管理模块，包括对学期信息、教学周信息、教学周日期、上课时间与上课节次等信息的查询与管理。每学期包含 16 个教学周，每周包含 5 个教学工作日，每个工作日包含 12 个上课节次，上午 1～5 节课，下午 6～9 节课，晚上 10～12 节课。系统可根据学期教学周第一天的日期自动计算并生成 1～16 周每一天的具体日期，极大方便了管理员进行日常管理与维护，避免了手动输入操作出错率高且效率低的问题。

3）课程管理模块，包括对课程信息、任课教师、授课资料、上课地点、上课人数、学生信息与专业班级等信息的查询与管理。

4）教师管理模块，包括对教师个人简介、所在系部、专业背景、教学成果以及学术成就的查询与管理，允许教师使用个人账号登录系统编辑信息，便于学生对教师有更详细的了解。

5）实验室预约模块，包括预约信息查询、预约操作管理与冲突检测功能。允许教师根据教学计划制定的实践教学环节与课程实际需要，结合实验室现有的软硬件设备条件，通过系统提供的在线预约平台进行实验室预约操作。

6）数据同步模块，实现课程、班级信息的定期同步更新。MySQL 数据库服务器负责与教务处等相关部门进行数据对接，利用 MySQL 存储过程定期监测教务处数据的变动情况，并根据数据变化同步更新数据信息。

3.1 实验室预约模块

实验室预约模块是经管类实验室预约系统的核心。教师登录系统，通过预约表单填写预约信息，表单元素包括机房选择框、教学周选择框、时间选择框、节次选择框、课程输入框、学生人数输入框等。为方便教师进行预约操作，表单设计要尽可能做到界面简洁、功能清晰，同时能够适应手机界面显示。采用 Bootstrap 技术对 HTML 表单元素进行重新排版与布局，以适应教师的使用需求。实验室预约模块主界面如图 4 所示。

图 4 实验室预约模块主界面

教师根据课程实际需要，选择机房、教学周、上课时间与上课节次，输入课程名称、学生人数与专业班级，点击“预约”按钮，启动预约处理流程。预约信息通过互联网传输至服务器，在服务器内处理预约请求。

在服务器内执行的预约操作处理流程如下：

1）调用 model－＞validate（）方法，通过 Rules 验证器对预约信息各个字段的输入值进行合法性验证。

2）验证通过后，运行 beginTransaction（）开启事务（Transaction）。

3）在事务处理的操作序列中，执行预约操作 Action。

4）预约操作 Action 执行时，会首先触发与其绑定的 Behavior 行为，对 Action 执行过程实施动态拦截。行为程序以 AOP 切面形式自动嵌入 Action 执行过程中，并运行部署在行为中的冲突检测过滤器（Filter），进行预约冲突检测。

5）预约有冲突，则终止当前 Action 的执行，抛出异常（Exception），并执行事务回滚（rollback），取消先前执行的所有预约操作。同时捕获异常，显示冲突提示信息。

6）预约无冲突，则继续执行预约操作 Action。

7）当一次预约请求过程中的多个预约操作 Action 都执行完成（未出现预约冲突），则调用 transaction－＞commit（）提交事务，将预约信息存入 MySQL 数据库，同时更新

数据缓存。

预约操作处理流程如图 5 所示。

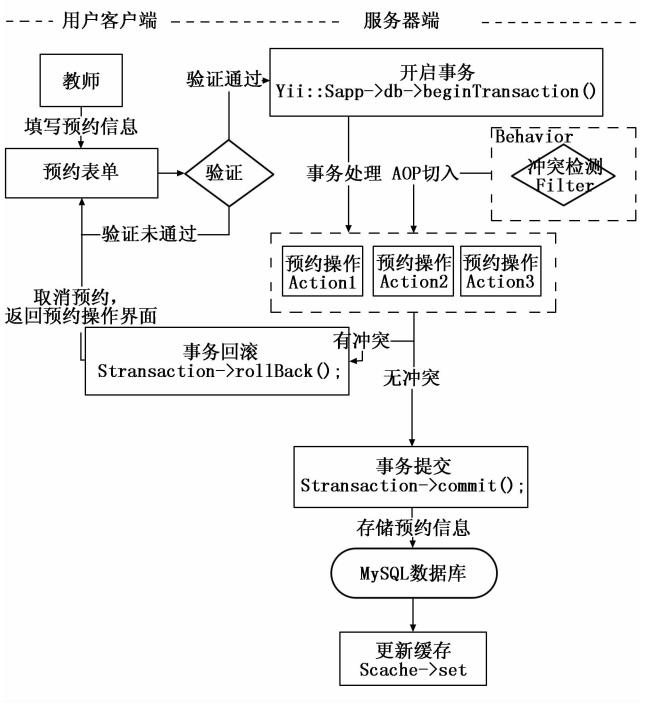

图 5　预约操作处理流程

3.2　基于 AOP 实现冲突动态检测

在实验室预约管理工作中，经常会出现预约时间冲突，即同一实验室在某一时间段已存在预约，无法再次进行预约的情况。因此采用相应的冲突检测机制，自动识别预约冲突，并在客户端显示提示信息，便于教师合理安排上课时间。常用的冲突检测方式包括客户端检测方式与服务端检测方式。

1）客户端检测方式：需要预先从数据库中获取已存在的实验室预约记录，然后在客户端浏览器中利用 jQuery 与当前预约信息进行比较，检测预约冲突。这种方式需要预先从服务器中获取大量的数据，对网络带宽有较高的要求，容易造成客户端运行卡顿，页面响应速度慢等问题。

2）服务端检测方式：预约请求通过网络传递给相应的 Action 方法，在 Action 方法内获取数据库数据，执行预约冲突检测，如存在冲突，则返回提示信息。数据读取和检测处理操作都在服务器端进行，大大减少了网络的数据传输量，同时由于服务器端的数据处理与运算能力远远超过客户端，因此在服务器端进行检测处理操作，效率更高，速度更快。

系统采用服务端检测方式，基于 AOP 面向切面编程技术，在运行时动态加载检测程序[9]。

设置命名空间 namespace app \ components，导入 yii \ base \ ActionFilter 基础类，通过继承 ActionFilter 创建过滤器（Filter）。ActionFilter 类包含 beforeAction 与 afterAction 方法，分别表示在 Action 执行前与执行后调用的代码块。重写 Filter 的 beforeAction 方法，并在 beforeAction 方法内执行冲突检测程序。关键代码如下：

```
class CheckFilter extends ActionFilter {
public function beforeAction(action) {
s_data = Yii::app->request->post();
  //检测预约冲突
cr = checkConflict(s_data);
if(cr){ //检测无冲突
return parent::beforeAction(action);
}else{ //检测有冲突
this->redirect(['room/dealconflict','cr'=>cr]);
return false;
}
}
}
```

在控制器中配置行为，然后在行为中部署过滤器，通过 class 属性指定当前配置的过滤器类名，再通过 only 属性将过滤器与相应的 Action 方法进行绑定。当执行预约操作时，触发行为，通过行为加载并调用过滤器，执行预约冲突检测程序。部署过滤器的关键代码如下：

```
public function behaviors() {
return [
[ //过滤器配置
'class'=>'app\components\CheckFilter',
'only'=>['reserve'],
],
];
}
```

预约冲突检测程序的运行时动态加载过程如下：

1）服务端在收到教师提交的实验室预约请求后，创建控制器实例，调用 runAction 方法。

2）在 runAction 内，先调用 createAction 方法创建执行实验室预约操作的 Action 动作实例，然后执行 this->beforeAction（action），此处的 this 表示控制器实例，运行控制器的 beforeAction 方法。

3）在控制器的 beforeAction 方法内，触发 trigger 函数，通过 ensureBehaviors 绑定行为（Behavior），即将冲突检测程序与执行预约操作的 reserveAction 动作进行绑定，实现 AOP 动态切入[10]。

4）调用 ActionFilter 基类中的 beforeFilter 方法，在 beforeFilter 中执行其子类 CheckFilter（自定义过滤器，部署在行为中，用于执行预约冲突检测）的 beforeAction 方法，此处的 beforeAction 与上述通过控制器调用的不同，通过控制器调用的 beforeAction 方法用于绑定行为，而通过过

滤器调用的 beforeAction 方法用于执行实验室预约冲突检测操作。

5）在 CheckFilter 过滤器的 beforeAction 方法中，调用 checkConflict 执行预约冲突检测操作，如图 6 所示。

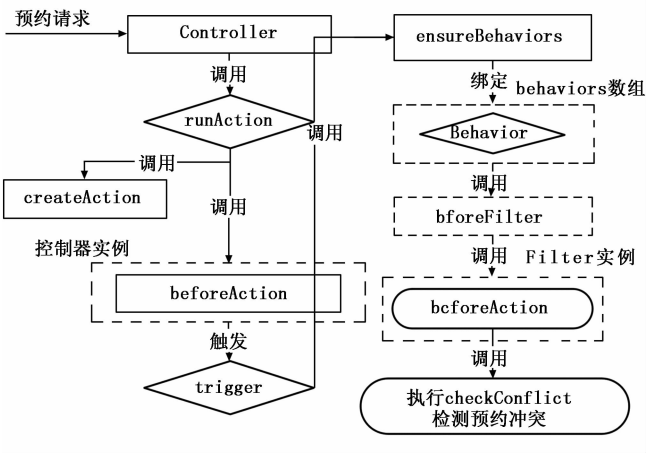

图 6 冲突检测程序的运行时动态加载过程

4 实验结果与分析

系统提供信息查询与线上预约等功能，允许师生在线查询实验室预约信息、课程信息、课程面向专业、学生人数。系统界面采用 Bootstrap 实现，通过“教学实验室”列表，查看实验室各个机房的课程预约信息；通过“教学周”列表查看每周的课程安排；通过“学期”列表，查看各个学期课程预约历史记录。在课表页面点击课程名称，弹出 Modal 详情页，显示该课程的详细介绍，包括：实验大纲、实验卡片、实验授课计划、实验指导数等相关资料。实验室预约信息查询界面图 7 所示。

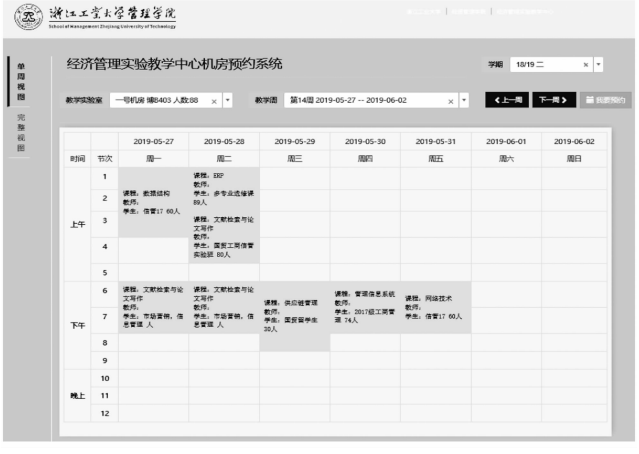

图 7 实验室预约信息查询界面

目前，经管类实验室预约系统已经在浙江工业大学管理学院与经济学院进行了部署与应用，为师生查询实验室信息、教学资源、课程信息与课程安排，教师预约实验室提供了便捷的线上服务平台。系统面向专业包括国际经济与贸易、市场营销、旅游管理、工商管理、财务管理、信息管理与信息系统、金融学、工程管理等 8 个本科专业；研究生、MBA 及各类工程硕士；建筑工程学院、药学院的一体化双专业以及全校的经济管理类二专业和选修课。涉及的实践教学包括 44 门实验课程、172 个实验项目、21 门课程设计。通过信息化手段的使用，将管理学院与经济学院的 6 个学科和 8 个本科专业的实验教学纳入到统一的实验课程预约管理综合平台中，实行“大平台，统一管理”。

5 结语

基于当前实验教学管理工作存在的问题，分析使用成品软件在功能扩展与二次开发方面的短板，整合现有实验教学资源，基于 MVC 设计模式开发经管类实验室预约系统，系统采用 AOP 面向切面编程技术实现预约冲突的动态检测，有效避免出现排课冲突、重复排课等教学事故。在满足功能需求的同时，优化系统架构设计，降低各模块之间的耦合度，提高灵活性，便于功能扩展与个性化定制。同时与本科论文管理系统、教师数据中心等其他应用进行无缝对接，只需登陆一个账户便可同时使用多项应用，方便教师开展工作。系统架构设计灵活，只需更改相应模块里的功能代码即可应用于其他行业领域，在系统架构优化方面有一定的参考意义。

参考文献：

[1] 卢肖霞. SSH 框架在 Web 项目开发中的设计与实现 [J]. 计算机测量与控制，2018，26 (10)：122 - 127.

[2] 张丽晔，蔡斐华，褚厚斌，等. 面向飞行器的自主保障寿命管理软件设计 [J]. 计算机测量与控制，2019，27 (7)：115 - 118，127.

[3] 彭小明，叶洁邢. 基于 3G/WIFI 的远程指纹考勤系统的设计与实现 [J]. 计算机应用与软件，2016，33 (5)：102 - 107.

[4] 祝杨军. 高校创业实验室的建设与探索 [J]. 实验技术与管理，2016，33 (11)：259 - 262.

[5] 冯建周，王晓寰，张 莹. 新工科背景下的创新创业教育实验室建设 [J]. 教育教学论坛，2018 (44)：273 - 274.

[6] 陈 晰，朱雅各，张著森，等. 创新创业教育背景下高校实验室管理改革探讨 [J]. 龙岩学院学报，2018，36 (5)：106 - 109.

[7] 王圣辉. 浙江省护林员考勤巡查系统设计与应用 [J]. 计算机应用与软件，2018，35 (11)：136 - 141.

[8] 季知祥，邓春宇，吴江宁，等. 面向智能电网的数据资产管理系统设计与应用 [J]. 计算机应用与软件，2019，36 (4)：118 - 123.

[9] 欧阳宏基，宋笑雪，李 红. 整合 ESMSH 框架的 Java EE 应用架构 [J]. 计算机测量与控制，2018，26 (10)：230 - 234.

[10] 张 时，王向东，李树江. 在线油烟实时监测系统的设计与实现 [J]. 计算机测量与控制，2019，27 (8)：35 - 39.