预约挂号系统

详细设计说明书

2017年10月

修订历史纪录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 说明 | 作者 |
| 2017-10-10 | 1.0 |  |  |
| 2017-10-14 | 2.0 |  |  |
| 2017-10-18 | 3.0 |  |  |
|  |  |  |  |

目录

[1. 引言 - 1 -](#_Toc31129)

[1.1 目的 - 1 -](#_Toc13159)

[1.2 参考资料 - 1 -](#_Toc19607)

[1.3 名称解释 - 1 -](#_Toc30184)

[1.4 关键字 - 1 -](#_Toc323)

[2. 功能概述 - 2 -](#_Toc22393)

[2.1系统管理员用例 - 2 -](#_Toc371)

[2.1.1 医院管理 - 2 -](#_Toc32098)

[2.1.2 医院管理员管理 - 9 -](#_Toc23794)

[2.1.3 系统设置 - 16 -](#_Toc16730)

[2.1.4 后台登录 - 22 -](#_Toc21324)

[2.2医院管理员用例 - 22 -](#_Toc18088)

[2.2.1 医院信息管理 - 22 -](#_Toc31979)

[2.2.2 科室管理 - 27 -](#_Toc17735)

[2.2.3 专家管理 - 33 -](#_Toc5688)

[2.2.4 挂号设置管理 - 40 -](#_Toc8969)

[2.2.5 黑名单功能 - 46 -](#_Toc17559)

[2.2.5 数据报表管理 - 50 -](#_Toc29466)

[2.3医院专家用例 - 53 -](#_Toc11386)

[2.3.1 坐诊时间管理 - 53 -](#_Toc30288)

[2.3.2 挂号信息管理 - 57 -](#_Toc10465)

[2.3.3 个人管理 - 60 -](#_Toc19231)

[2.4 患者用例 - 64 -](#_Toc14337)

[2.4.1 登录 - 64 -](#_Toc12859)

[2.4.2 预约挂号 - 68 -](#_Toc3552)

[2.4.3 预约查询 - 75 -](#_Toc13604)

[2.4.4 个人管理 - 79 -](#_Toc1329)

[2.4.5 系统公告 - 82 -](#_Toc9775)

# 引言

## 1.1 目的

此设计说明书是预约挂号开发的主要依据，描述程序系统结构，并进行程序设计，为开发人员提供工作基准文件，并对后续阶段的工作起指导作用。

预期读者为设该系统的系统分析员、程序员、测试员、实施人员等。

## 1.2 参考资料

## 1.3 名称解释

|  |  |
| --- | --- |
| **名称** | **解释** |
| 预约挂号 | 网上预约专家 |
|  |  |

## 1.4 关键字

高级搜索，擅长诊治，提前预约时间，门诊日期，门诊时间，约满，停诊，赴约，失约。

# 功能概述

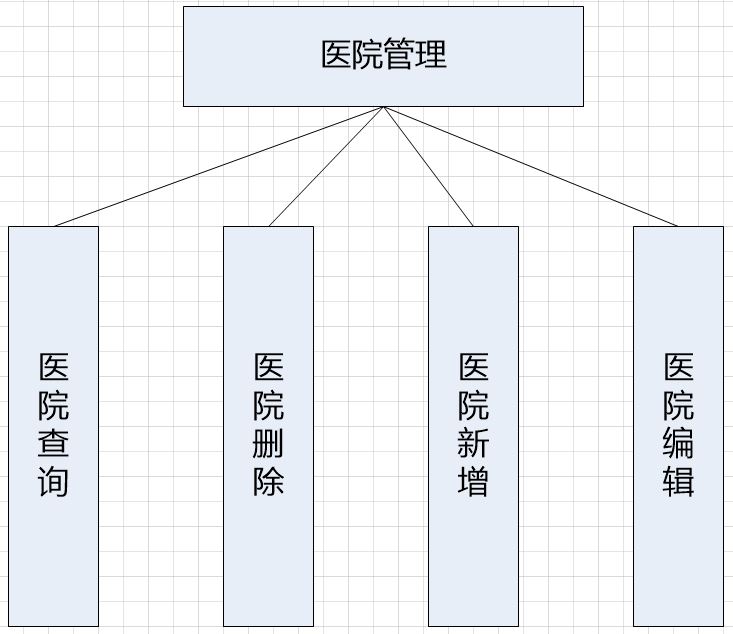
## 2.1系统管理员用例

### 2.1.1 **医院管理**

1. 需求概述：

此用例允许系统管理员登录到系统之后对医院进行增加，删除，修改，查询操作，维护各个医院的大致信息。

2. 软件结构：



3.具体描述：

1. 医院查询：

功能：根据系统管理员输入的查询条件，查询出符合要求的医院信息，如果没有查询到相关医院的信息，则提示用户没有相关搜索结果。

输入：医院代码\名称，行政区划，医院级别

输出：医院代码，医院名称，行政区划，级别

1. 医院删除：

功能：系统管理员可通过医院查询用例查询出想要删除的医院，然后可点击删除操作。

输入：无。

输出：医院删除成功提示信息。

（3）医院新增：

功能：系统管理员填写医院增加表单，从而完成医院增加用例。

输入：医院代码，医院名称，行政区划，医院级别

输出：医院增加成功提示信息。

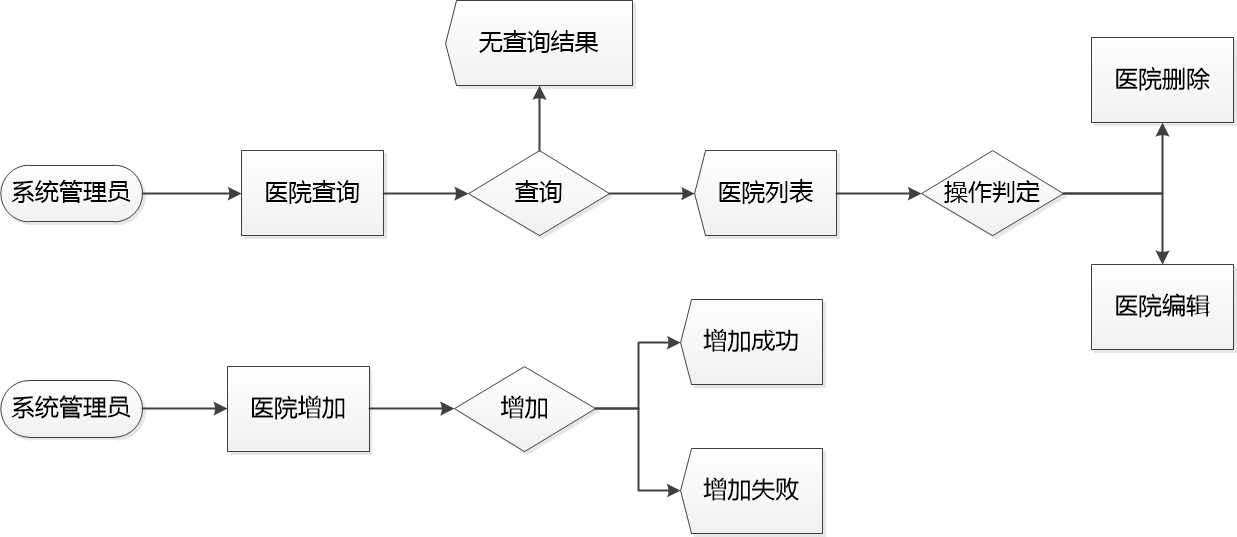
（4）医院编辑：

功能：系统管理员可通过医院查询用例查询出想要编辑的医院，然后可点击编辑操作，从而跳转到编辑页面，同时编辑页面也会加载相应的医院信息。

输入：行政区划，医院级别（医院代码，医院名称不能修改）

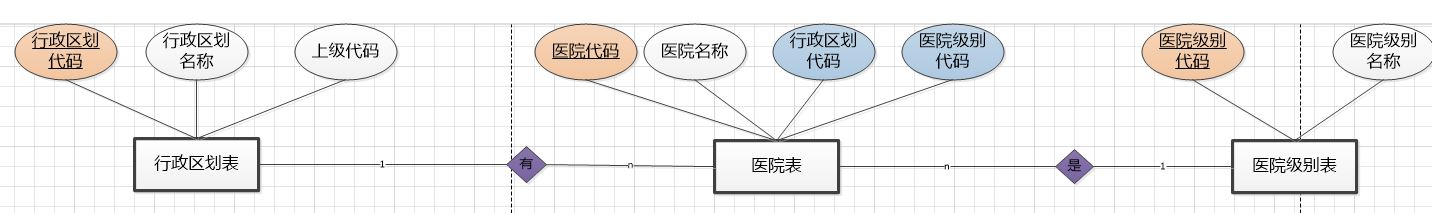
输出：医院编辑成功提示信息。

1. 流程图：



1. 数据库设计

ER图：



6.程序界面

医院查询，删除页面：



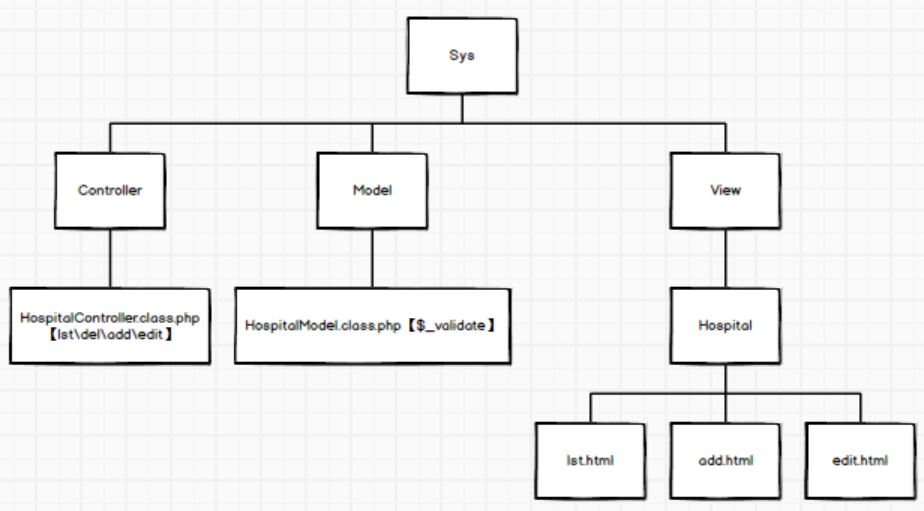
医院新增页面：



医院编辑页面：



7.代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Sys\Controller | HospitalController.class.php  【lst(),del(),add(),edit() 】 | 医院控制器类 |
| Sys\Model | HospitalModel.class.php | 医院模型类 |
| Sys\View\Hospital | lst.html | 医院列表模板 |
| Sys\View\Hospital | add.html | 医院增加模板 |
| Sys\View\Hospital | edit.html | 医院修改模板 |

|  |  |
| --- | --- |
| Model类设计 | //hospital表的模型类，用于自动验证  class HospitalModel extends Model{} |
| 处理逻辑 | //用于自动验证  protected $\_validate=array(  array("hos\_code","checkLength","医院代码长度应该在20个字内!",0,"callback",3,array(20)),  array("hos\_name","checkLength","医院名称长度应该在40个字内!",0,"callback",3,array(40)),  );  function checkLength($str,$maxlength){  if(strlen($str)<=$maxlength){  return true;  }else{  return false;  }  } |

|  |  |
| --- | --- |
| 方法设计 | //医院列表展示  //1.用于第一次刷出这个页面时的整表展示  //2.用于用户输入查询条件时的展示  public function lst(){} |
| 处理逻辑 | 梳理：用户可以传值的字段有3个：医院代码\名称，行政区划，医院级别  医院代码\名称有：  行政区划有：  医院级别有：  医院级别无：  行政区划无：  医院级别有：  医院级别无：  医院代码\名称无：  行政区划有：  医院级别有：  医院级别无：  行政区划无：  医院级别有：  医院级别无：  $sql="select \* from hospital";  $selectModel=new Model();  $selectData=$selectModel->query($sql);  $this->assign("selectData",$selectData);  $this->display(); |

|  |  |
| --- | --- |
| 方法设计 | //删除某一条hospital表中的记录  public function del(){} |
| 处理逻辑 | $hos\_code=I("get.hos\_code");  $deleteModel=M("hospital");  $deleteModel->delete($hos\_code); |

|  |  |
| --- | --- |
| 方法设计 | //向hospital表中添加一条记录，并且会自动验证添加内容  public function add(){} |
| 处理逻辑 | if(IS\_POST){  $addData["xxx"]=I("post.xxx");  $addModel=D("hospital");  //这里用了自动验证  if(!$addModel->create($addData)){  $this->error($addModel->getError());  }else{  $addModel->add();  }  }  $this->assign();  $this->display(); |

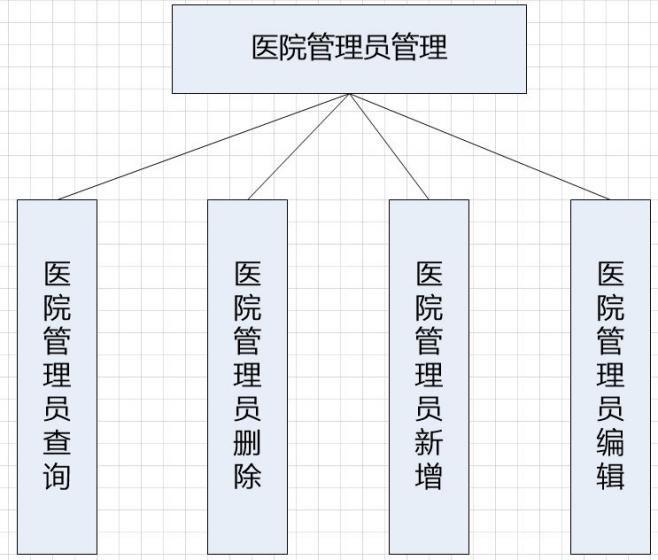
|  |  |
| --- | --- |
| 方法设计 | //医院编辑  public function edit(){} |
| 处理逻辑 | //如果用户点击了编辑提交按钮，那么则将信息保存在数据库  if(IS\_POST){  $updateData["xxx"]=I("post.xxx");  $updateModel=D("hospital");  //这里用了自动验证  if(!$updateModel->create($updateData)){  $this->error($updateModel->getError());  }else{  $updateModel->where("hos\_code=".I("get.hos\_code"))->save($updateData);  }  }  //获取  $hos\_code=I("get.hos\_code");  $selectModel=M("hospital");  $selectData=$selectModel->where("hos\_code=".$hos\_code)->find();  //展示  $this->assign("selectData",$selectData);  $this->display(); |

### 2.1.2 **医院管理员管理**

1. 需求概述：

本用例允许系统管理员针对每一个医院设置一个管理员并且可以删除，修改，查询设置的医院管理员，医院管理员可以设置各自医院的详细信息（医院图片，医院地址，医院简介，医院电话信息）。

1. 软件结构：



1. 具体描述：
2. 医院管理员查询：

功能：系统管理员可通过该用例完成医院管理员查询操作。

输入：医院管理员名字，管理医院。

输出：医院管理员名字，医院管理员手机，管理医院。

（2）医院管理员删除：

功能：系统管理员可通过医院管理员查询用例查询出想要删除的医院管理员，然后可点击删除操作。

输入：无。

输出：医院管理员删除成功提示信息。

（3）医院管理员新增：

功能：系统管理员可通过该用例完成医院管理员的增加操作。

输入：医院管理员名字，密码，手机，管理医院。

输出：医院管理员增加成功提示信息。

备注：一个医院只能设置一个医院管理员。

（4）医院管理员编辑：

功能：系统管理员可通过医院管理员查询用例查询出想要编辑的医院管理员，然后可点击编辑操作，从而跳转到编辑页面，同时编辑页面也会加载相应的医院管理员信息。

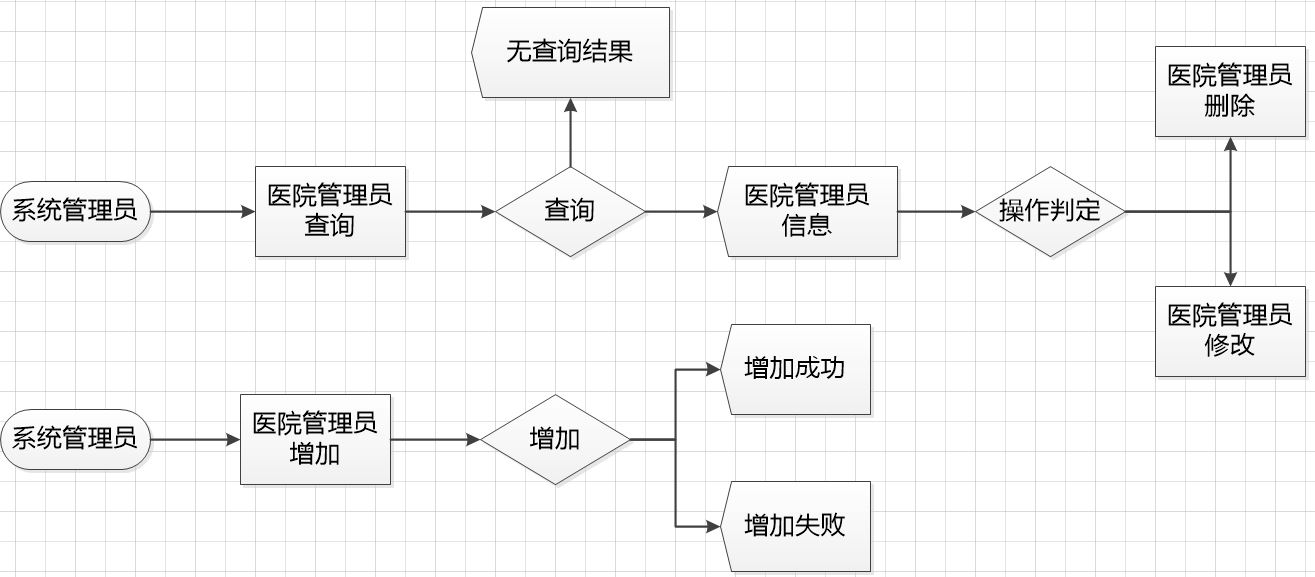
输入：管理员密码\管理员手机\管辖医院。

输出：医院编辑页面，医院编辑成功提示信息。

备注：一个医院只能设置一个医院管理员。

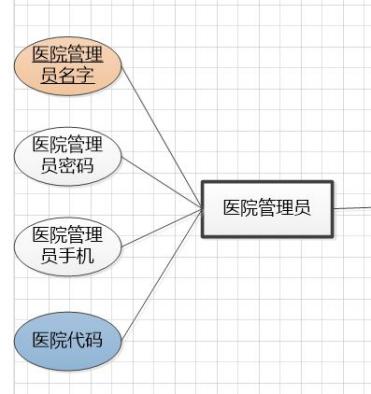
管理员名字不能修改。

1. 流程图：



5.数据库设计

ER图：



1. 程序界面：

医院管理员查询,删除：



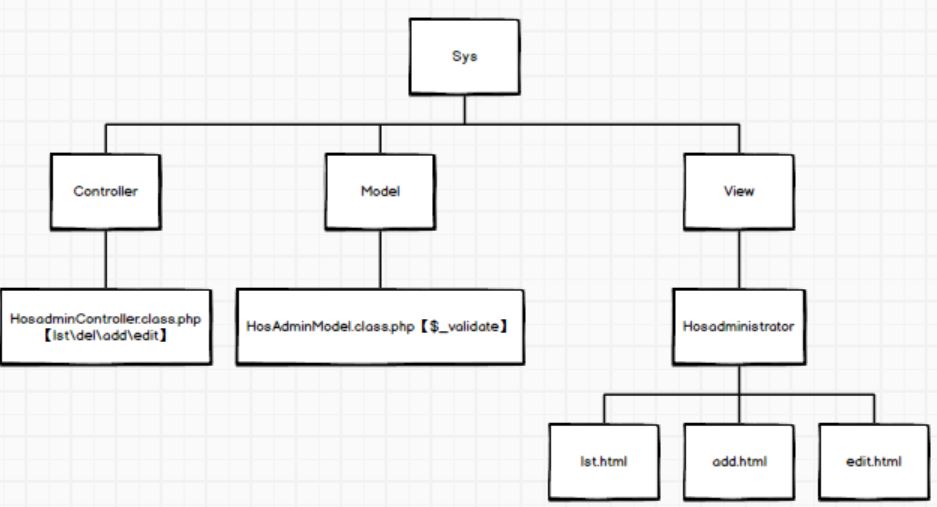
医院管理员增加：



医院管理员修改：



7.代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Sys\Controller | HosadminController.class.php  【lst(),del(),add(),edit()】 | 医院管理员控制器类 |
| Sys\View\Hosadministrator | lst.html | 医院管理员列表模板 |
| Sys\View\Hosadministrator | add.html | 医院管理员增加模板 |
| Sys\View\Hosadministrator | edit.html | 医院管理员修改模板 |

|  |  |
| --- | --- |
| Model类设计 | //hosAdmin表的模型类，用于自动验证  class HosAdminModel extends Model{} |
| 处理逻辑 | //用于自动验证  protected $\_validate=array(  array("hosadmin\_name","checkLength","医院管理员名字长度应该在20个字内!",0,"callback",3,array(20)),  array("hosadmin\_pwd","checkLength","医院管理员密码长度应该在10个字内!",0,"callback",3,array(40)),  array("hos\_code","","一个医院只能设置一个管理员!",0,"unique",3),  array("hosadmin\_tel","/^1[3578][0-9]{9}$/","手机号格式错误!",0,"regex",3),  );  function checkLength($str,$maxlength){  if(strlen($str)<=$maxlength){  return true;  }else{  return false;  }  } |

|  |  |
| --- | --- |
| 方法设计 | //医院管理员列表展示  //1.用于第一次刷出这个页面时的整表展示  //2.用于用户输入查询条件时的展示  public function lst(){} |
| 处理逻辑 | 梳理：用户可以传值的字段有2个：医院管理员名字，管辖医院  医院管理员名字有：  管辖医院有：  管辖医院无：  医院管理员名字无：  管辖医院有：  管辖医院无：  $sql="select \* from hosAdmin";  $selectModel=new Model();  $selectData=$selectModel->query($sql);  $this->assign("selectData",$selectData);  $this->display(); |

|  |  |
| --- | --- |
| 方法设计 | //删除某一条hosAdmin表中的记录  public function del(){} |
| 处理逻辑 | $hosadmin\_name=I("get.hosadmin\_name");  $deleteModel=M("hosAdmin");  $deleteModel->delete($hosadmin\_name); |

|  |  |
| --- | --- |
| 方法设计 | //向hosAdmin表中添加一条记录，并且会自动验证添加内容  public function add(){} |
| 处理逻辑 | if(IS\_POST){  $addData["xxx"]=I("post.xxx");  $addModel=D("hosAdmin");  //这里用了自动验证  if(!$addModel->create($addData)){  $this->error($addModel->getError());  }else{  $addModel->add();  }  }  $this->assign();  $this->display(); |

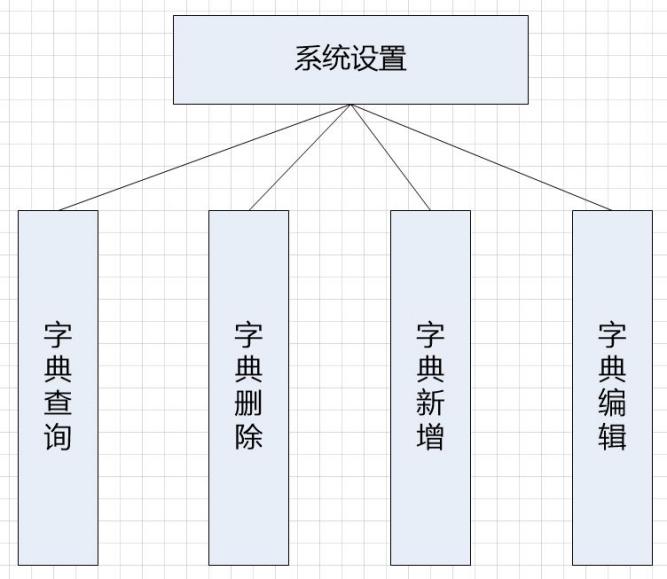
|  |  |
| --- | --- |
| 方法设计 | //医院管理员编辑  public function edit(){} |
| 处理逻辑 | //如果用户点击了编辑提交按钮，那么则将信息保存在数据库  if(IS\_POST){  $updateData["xxx"]=I("post.xxx");  $updateModel=D("hosAdmin");  //这里用了自动验证  if(!$updateModel->create($updateData)){  $this->error($updateModel->getError());  }else{  $updateModel->where("hosadmin\_name=".I("post.hosadmin\_name"))->save($updateData);  }  }  //获取  $hosadmin\_name=I("get.hosadmin\_name");  $selectModel=M("hosadmin");  $selectData=$selectModel->where("hosadmin\_name=".$hosadmin\_name)->find();  //展示  $this->assign("selectData",$selectData);  $this->display(); |

### 2.1.3 **系统设置**

1. 需求概述：

此用例允许系统管理员登录到系统之后进行字典的配置。

1. 软件结构：



1. 具体描述：

（1）字典查询：

功能：医院管理员登录到系统之后可以查看字典。

输入：字点名。

输出：字典列表。

1. 字典删除：

功能：医院管理员登录到系统之后可以查看字典，在该列表项之后点击删除即可。

输入：无。

输出：删除成功提示信息。

1. 字典新增：

功能：医院管理员登录到系统之后可以新增字典。

输入：字典名，字典代码，字典值。

输出：新增成功提示信息。

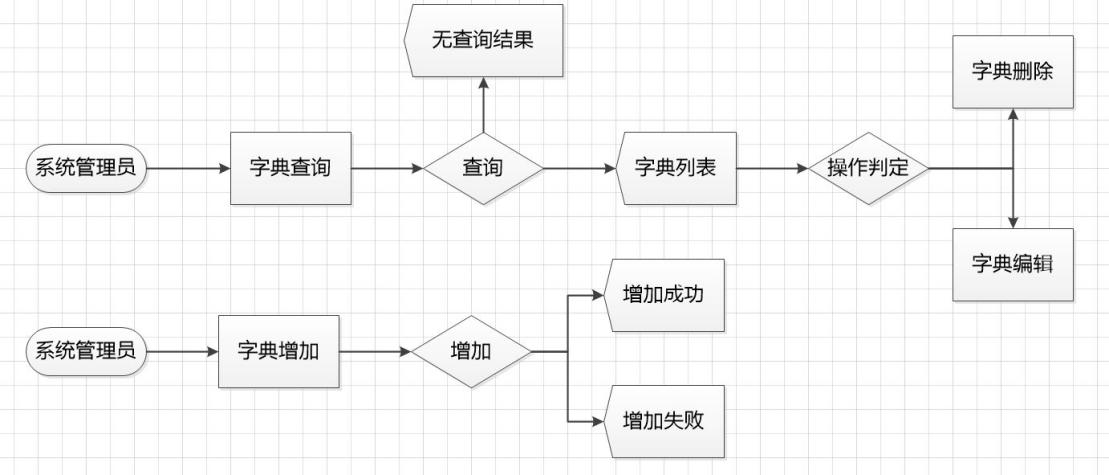
1. 字典编辑：

功能：医院管理员登录到系统之后可以查看字典，在该列表项之后点击编辑。

输入：字典代码，字典值。

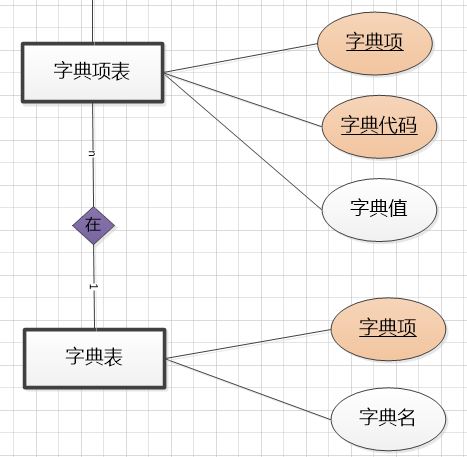
输出：编辑成功。

1. 流程图：



5.数据库设计

ER图：



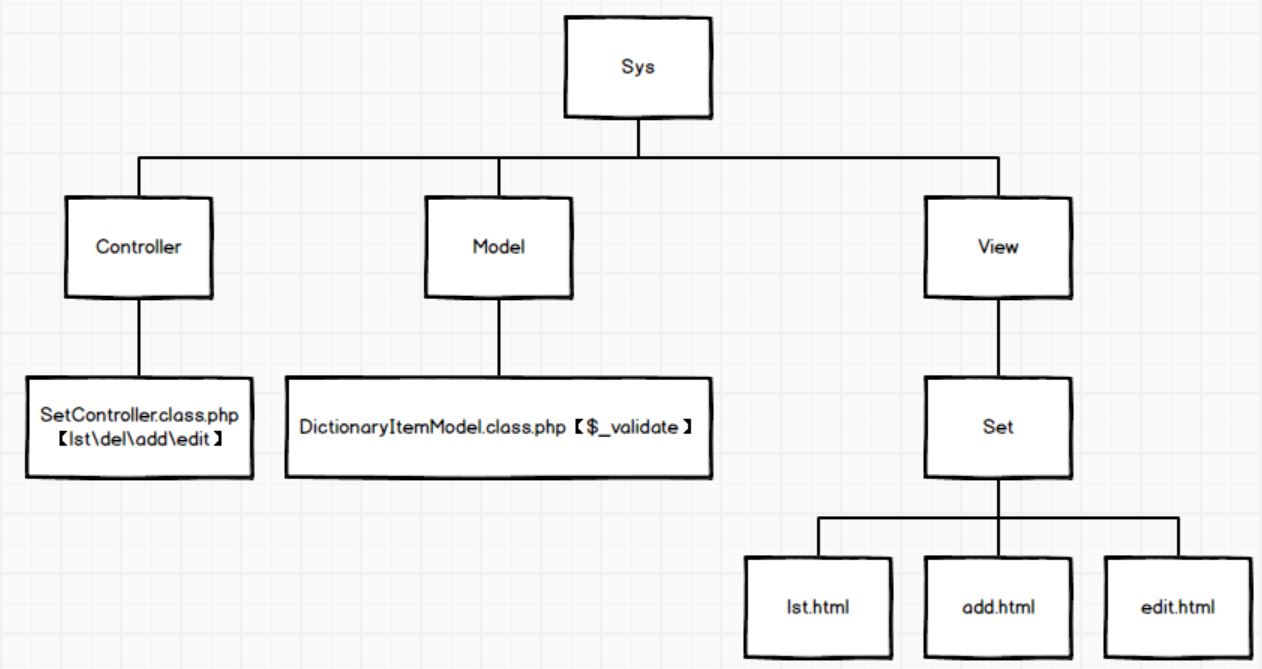
1. 程序界面：







1. 代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Sys\Controller | SetController.class.php  【lst(),del(),add(),edit()】 | 医院管理员控制器类 |
| Sys\Model | DictionaryItemModel.class.php【$\_validate】 | 模型类 |
| Sys\View\Set | lst.html | 列表模板 |
| Sys\View\Set | add.html | 新增模板 |
| Sys\View\Set | edit.html | 编辑模板 |

|  |  |
| --- | --- |
| Model类设计 | //dictionaryItem表的模型类，用于自动验证  class DictionaryItemModel extends Model{} |
| 处理逻辑 | //用于自动验证  protected $\_validate=array(  array("dic\_code","checkLength","字典代码应该在20个字内!",0,"callback",3,array(20)),  array("dic\_value","checkLength","字典值应该在20个字内!",0,"callback",3,array(20)),  );  function checkLength($str,$maxlength){  if(strlen($str)<=$maxlength){  return true;  }else{  return false;  }  } |

|  |  |
| --- | --- |
| 方法设计 | //字典列表展示  //1.用于第一次刷出这个页面时的整表展示  //2.用于用户输入查询条件时的展示  public function lst(){} |
| 处理逻辑 | 梳理：用户可以传值的字段有1个：字典名  字典名有：  字典名无：  $sql="xxx";  $selectModel=new Model();  $selectData=$selectModel->query($sql);  $this->assign("selectData",$selectData);  $this->display(); |

|  |  |
| --- | --- |
| 方法设计 | //删除某一条dictionaryItem表中的记录  public function del(){} |
| 处理逻辑 | $dic\_item=I("get.dic\_item");  $dic\_code=I("get.dic\_code");  $deleteModel=M("dictionaryItem");  $deleteModel->where("dic\_item=".$dic\_item." and dic\_code=".$dic\_code)->delete(); |

|  |  |
| --- | --- |
| 方法设计 | //向dictionaryItem表中添加一条记录，并且会自动验证添加内容  public function add(){} |
| 处理逻辑 | if(IS\_POST){  $addData["xxx"]=I("post.xxx");  $addModel=D("dictionaryItem");  //这里用了自动验证  if(!$addModel->create($addData)){  $this->error($addModel->getError());  }else{  $addModel->add();  }  }  $this->assign();  $this->display(); |

|  |  |
| --- | --- |
| 方法设计 | //字典项编辑  public function edit(){} |
| 处理逻辑 | //如果用户点击了编辑提交按钮，那么则将信息保存在数据库  if(IS\_POST){  $updateData["xxx"]=I("post.xxx");  $updateModel=D("dictionaryItem");  //这里用了自动验证  if(!$updateModel->create($updateData)){  $this->error($updateModel->getError());  }else{  $updateModel->where("dic\_item=".$dic\_item."anddic\_code=".$dic\_code)->save($updateData);  }  }  //获取  $dic\_item=I("get.dic\_item");  $dic\_code=I("get.dic\_code");  $selectModel=M("dictionaryItem");  $selectData=$selectModel->where("dic\_item=".$dic\_item."anddic\_code=".$dic\_code)->find();  //展示  $this->assign("selectData",$selectData);  $this->display(); |

### 2.1.4 **后台登录**

程序界面：

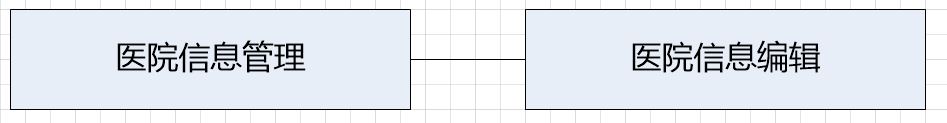
## 2.2医院管理员用例

### 2.2.1 **医院信息管理**

1. 需求概述：

本用例允许医院管理员管理自己管辖的医院，包括医院信息的修改

2.软件结构：



1. 具体描述：

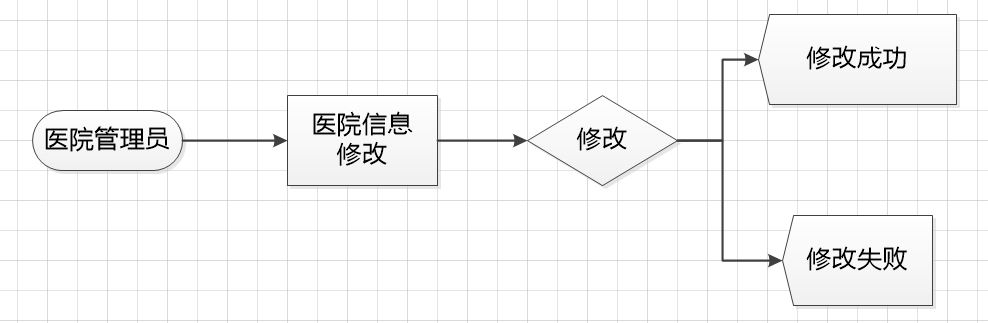
（1）医院信息编辑：

功能：医院管理员登录到系统之后可以查看自己管辖的医院的信息，并提交自己的修改。

输入：图片，地址，简介，电话。

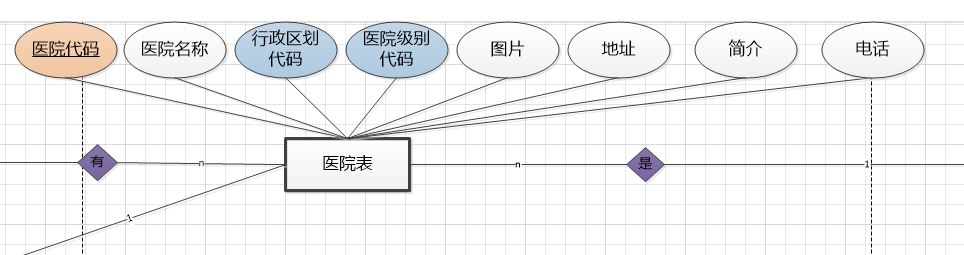
输出：医院信息修改成功提示信息。

1. 流程图：



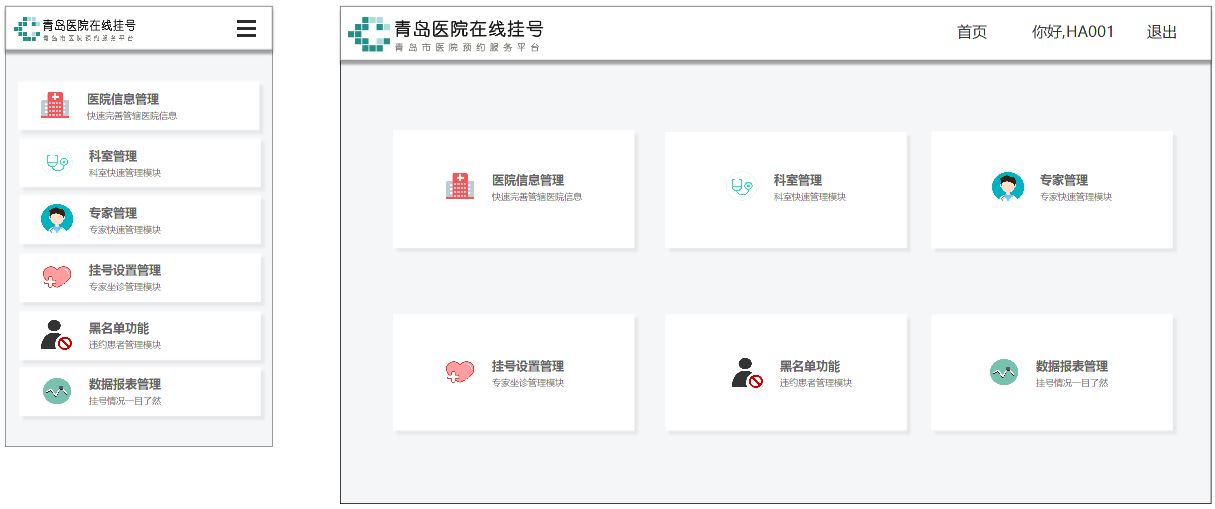
5.数据库设计

ER图：



1. 程序界面：

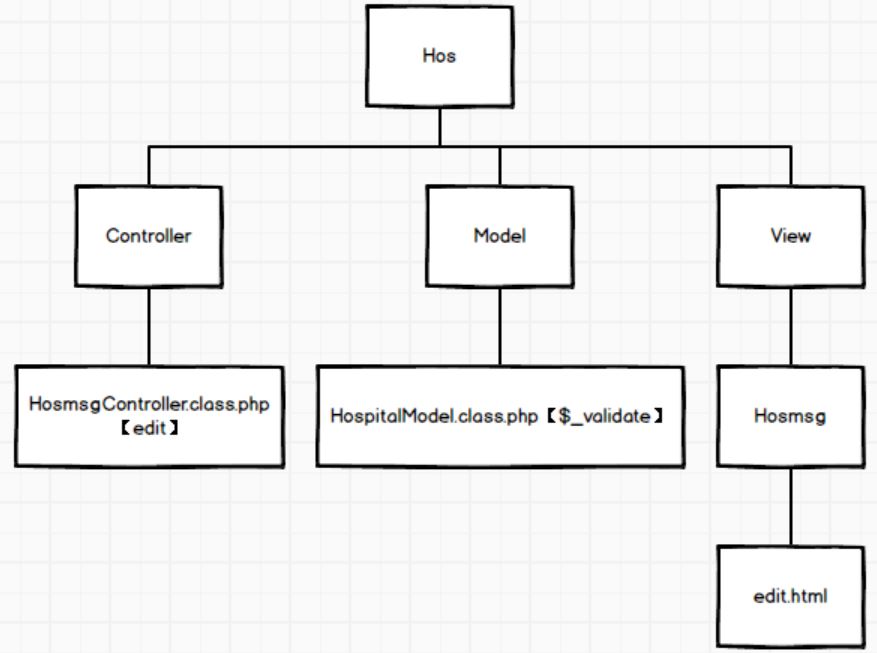
医院管理员权限：



医院信息管理（编辑）页面：



1. 代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Hos\Controller | HosmsgController.class.php  【edit()】 | 控制器类 |
| Hos\View\Hosmsg | edit.html | 视图模板 |

|  |  |
| --- | --- |
| Model类设计 | //hospital表的模型类，用于自动验证  class HospitalModel extends Model{} |
| 处理逻辑 | //用于自动验证  protected $\_validate=array(  array("hos\_code","checkLength","医院代码应该在20个字内!",0,"callback",3,array(20)),  array("hos\_name","checkLength","医院名称应该在40个字内!",0,"callback",3,array(40)),  array("addr","checkLength","医院地址应该在100个字内!",0,"callback",3,array(100)),  array("msg","checkLength","医院简介应该在200个字内!",0,"callback",3,array(200)),  );  function checkLength($str,$maxlength){  if(strlen($str)<=$maxlength){  return true;  }else{  return false;  }  } |

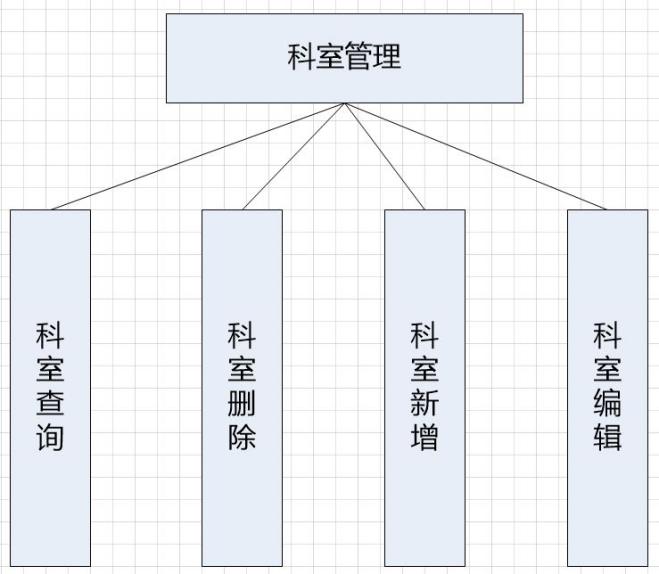
|  |  |
| --- | --- |
| 方法设计 | //医院信息管理编辑  public function edit(){} |
| 处理逻辑 | //如果用户点击了编辑提交按钮，那么则将信息保存在数据库  if(IS\_POST){  $updateData["xxx"]=I("post.xxx");  $updateModel=D("hospital");  //这里用了自动验证  if(!$updateModel->create($updateData)){  $this->error($updateModel->getError());  }else{  $updateModel->where("hos\_code=".I("post.hos\_code"))->save($updateData);  }  }  //获取  $hos\_code=I("get.hos\_code");  $selectModel=M("hosadmin");  $selectData=$selectModel->where("hos\_code=".$hos\_code)->find();  //展示  $this->assign("selectData",$selectData);  $this->display(); |

### 2.2.2 **科室管理**

1. 需求概述：

本用例允许医院管理员管理管辖医院里的科室信息。

1. 软件结构：



3.具体描述：

（1）科室查询：

功能：根据医院管理员输入的查询条件，查询出符合要求的科室信息，如果没有查询到相关科室的信息，则提示医院管理员没有相关搜索结果。

输入：科室代码，科室名称，科室类别

输出：科室代码，科室名称，科室类别，科室简介

（2）医院删除：

功能：医院管理员可通过科室查询用例查询出想要删除的科室，然后可点击删除操作。

输入：无。

输出：科室删除成功提示信息。

（3）医院新增：

功能：医院管理员填写科室新增表单，从而完成医院增加用例。

输入：科室代码，科室名称，科室类别，科室简介

输出：医院增加成功提示信息。

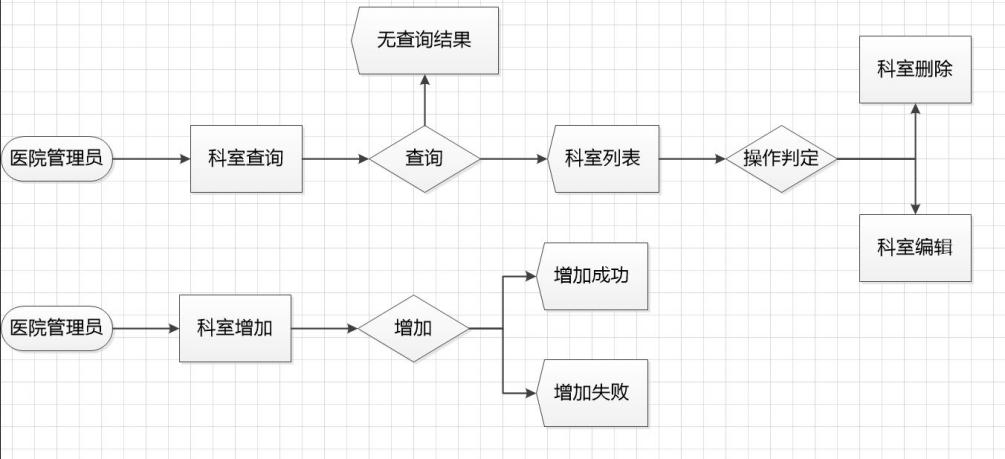
（4）科室编辑：

功能：医院管理员可通过医院查询用例查询出想要编辑的医院，然后可点击编辑操作，从而跳转到编辑页面，同时编辑页面也会加载相应的科室信息。

输入：科室简介。

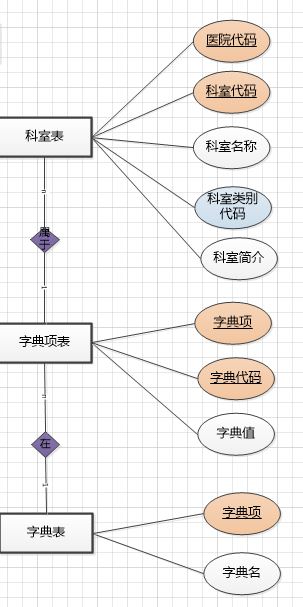
输出：科室编辑页面，科室编辑成功提示信息。

1. 流程图：



5.数据库设计

ER图：



1. 程序界面：

科室查询，删除界面：



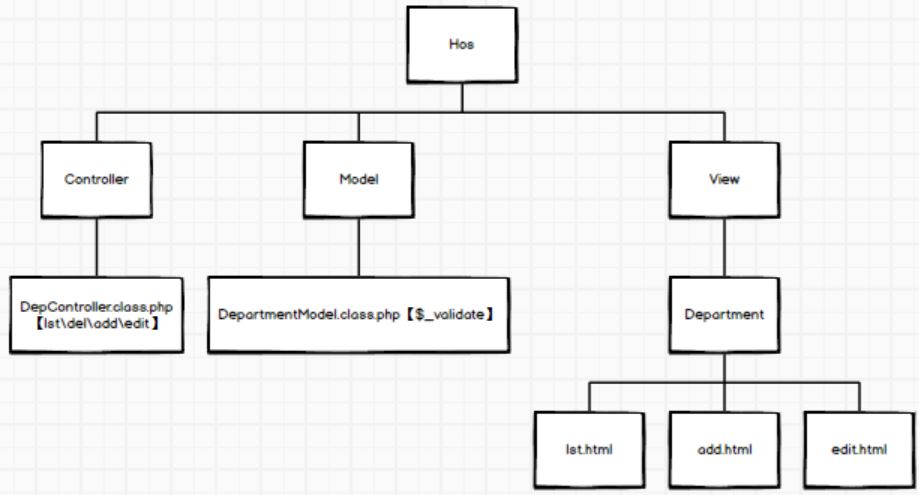
科室新增界面：



科室编辑界面：



7.代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Hos\Controller | DepController.class.php  【lst(),del(),add(),edit() 】 | 控制器类 |
| Hos\Model | DepartmentModel.class.php | 模型类 |
| Hos\View\Department | lst.html | 列表模板 |
| Hos\View\Department | add.html | 增加模板 |
| Hos\View\Department | edit.html | 修改模板 |

|  |  |
| --- | --- |
| Model类设计 | //department表的模型类，用于自动验证  class DepartmentModel extends Model{} |
| 处理逻辑 | //用于自动验证  protected $\_validate=array(  array("dep\_code","checkLength","科室代码应该在20个字内!",0,"callback",3,array(20)),  array("dep\_name","checkLength","科室名称应该在20个字内!",0,"callback",3,array(20)),  array("dep\_description","checkLength","科室简介应该在200个字内!",0,"callback",3,array(40)),  );  function checkLength($str,$maxlength){  if(strlen($str)<=$maxlength){  return true;  }else{  return false;  }  } |

|  |  |
| --- | --- |
| 方法设计 | //科室列表展示  //1.用于第一次刷出这个页面时的整表展示  //2.用于用户输入查询条件时的展示  public function lst(){} |
| 处理逻辑 | 梳理：用户可以传值的字段有3个：科室代码，科室名称，科室类别  科室代码有：  科室名称有：  科室类别有：  科室类别无：  科室名称无：  科室类别有：  科室类别无：  科室代码无：  科室名称有：  科室类别有：  科室类别无：  科室名称无：  科室类别有：  科室类别无：  $sql="select \* from department";  $selectModel=new Model();  $selectData=$selectModel->query($sql);  $this->assign("selectData",$selectData);  $this->display(); |

|  |  |
| --- | --- |
| 方法设计 | //删除某一条department表中的记录  public function del(){} |
| 处理逻辑 | $dep\_code=I("get.dep\_code");  $deleteModel=M("department");  $deleteModel->delete($dep\_code); |

|  |  |
| --- | --- |
| 方法设计 | //向department表中添加一条记录，并且会自动验证添加内容  public function add(){} |
| 处理逻辑 | if(IS\_POST){  $addData["xxx"]=I("post.xxx");  $addModel=D("department");  //这里用了自动验证  if(!$addModel->create($addData)){  $this->error($addModel->getError());  }else{  $addModel->add();  }  }  $this->assign();  $this->display(); |

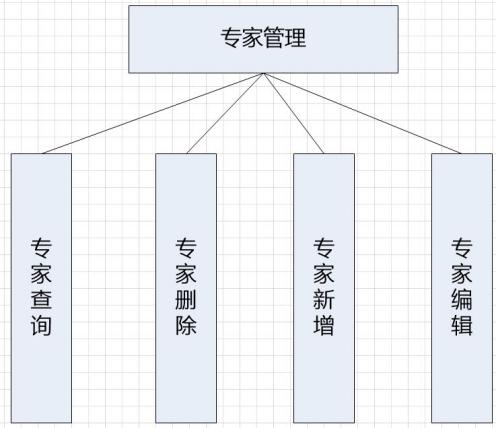
|  |  |
| --- | --- |
| 方法设计 | //科室编辑  public function edit(){} |
| 处理逻辑 | //如果用户点击了编辑提交按钮，那么则将信息保存在数据库  if(IS\_POST){  $updateData["xxx"]=I("post.xxx");  $updateModel=D("department");  //这里用了自动验证  if(!$updateModel->create($updateData)){  $this->error($updateModel->getError());  }else{  $updateModel->where("dep\_code=".I("get.dep\_code"))->save($updateData);  }  }  //获取  $dep\_code=I("get.dep\_code");  $selectModel=M("department");  $selectData=$selectModel->where("dep\_code=".$dep\_code)->find();  //展示  $this->assign("selectData",$selectData);  $this->display(); |

### 2.2.3 **专家管理**

1. 需求概述：

此用例允许医院管理员管理医院的相应科室下的专家信息。

1. 软件结构：



3.具体描述：

（1）专家查询：

功能：根据医院管理员输入的查询条件，查询出符合要求的专家信息，如果没有查询到相关专家的信息，则提示没有相关搜索结果。

输入：专家代码，专家名字，科室名称，专家级别

输出：专家代码，专家名字，科室名称，专家级别，专家手机

（2）专家删除：

功能：医院管理员可通过专家查询用例查询出想要删除的专家，然后可点击删除操作。

输入：无。

输出：专家删除成功提示信息。

（3）专家新增：

功能：医院管理员填写专家增加表单，从而完成专家增加用例。

输入：专家代码，专家名字，科室名称，专家级别，专家密码，专家手机

输出：专家增加成功提示信息。

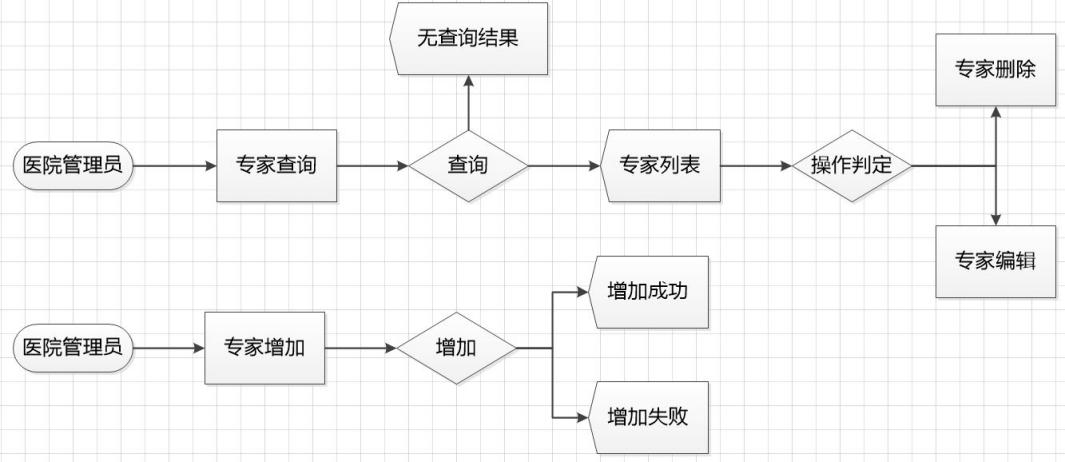
（4）专家编辑：

功能：医院管理员可通过专家查询用例查询出想要编辑的专家，然后可点击编辑操作，从而跳转到编辑页面，同时编辑页面也会加载相应的专家信息。

输入：科室名称，专家级别，专家密码，专家手机

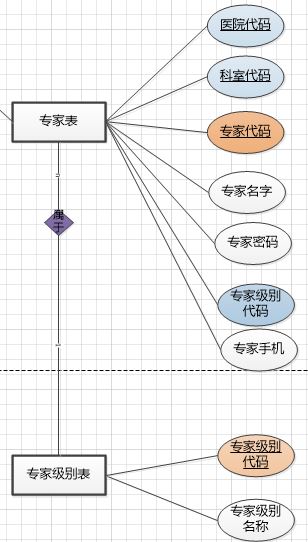
输出：专家编辑成功提示信息。

1. 流程图：



5.数据库设计

ER图：



1. 程序界面：

专家查询，删除页面：



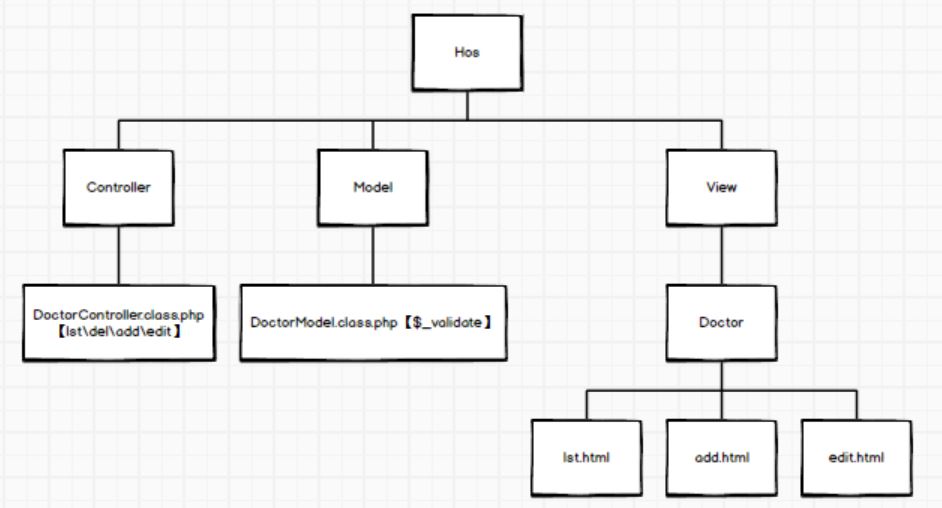
专家新增页面：



专家编辑页面：



1. 代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Hos\Controller | DoctorController.class.php  【lst(),add(),edit(),del() 】 | 控制器类 |
| Hos\View\Doctor | lst.html | 列表模板 |
| Hos\View\Doctor | add.html | 增加模板 |
| Hos\View\Doctor | edit.html | 修改模板 |

|  |  |
| --- | --- |
| Model类设计 | //表的模型类，用于自动验证  class DoctorModel extends Model{} |
| 处理逻辑 | //用于自动验证  protected $\_validate=array(  array("doc\_code","checkLength","专家代码应该在20个字内!",0,"callback",3,array(20)),  array("doc\_name","checkLength","专家名字应该在20个字内!",0,"callback",3,array(20)),  array("doc\_pwd","checkLength","专家密码应该在10个字内!",0,"callback",3,array(10)),  array("doc\_tel","/^1[3578][0-9]{9}$/","手机号格式错误!",0,"regex",3),  );  function checkLength($str,$maxlength){  if(strlen($str)<=$maxlength){  return true;  }else{  return false;  }  } |

|  |  |
| --- | --- |
| 方法设计 | //专家列表展示  //1.用于第一次刷出这个页面时的整表展示  //2.用于用户输入查询条件时的展示  public function lst(){} |
| 处理逻辑 | 梳理：用户可以传值的字段有4个:专家代码\专家名字，科室名称，专家级别  专家代码\专家名字有：  科室名称有：  专家级别有：  专家级别无：  科室名称无：  专家级别有：  专家级别无：  专家代码\专家名字无：  科室名称有：  专家级别有：  专家级别无：  科室名称无：  专家级别有：  专家级别无：  $sql="select \* from doctor";  $selectModel=new Model();  $selectData=$selectModel->query($sql);  $this->assign("selectData",$selectData);  $this->display(); |

|  |  |
| --- | --- |
| 方法设计 | //删除某一条doctor表中的记录  public function del(){} |
| 处理逻辑 | $doc\_code=I("get.doc\_code");  $deleteModel=M("doctor");  $deleteModel->delete($doc\_code); |

|  |  |
| --- | --- |
| 方法设计 | //向doctor表中添加一条记录，并且会自动验证添加内容  public function add(){} |
| 处理逻辑 | if(IS\_POST){  $addData["xxx"]=I("post.xxx");  $addModel=D("doctor");  //这里用了自动验证  if(!$addModel->create($addData)){  $this->error($addModel->getError());  }else{  $addModel->add();  }  }  $this->assign();  $this->display(); |

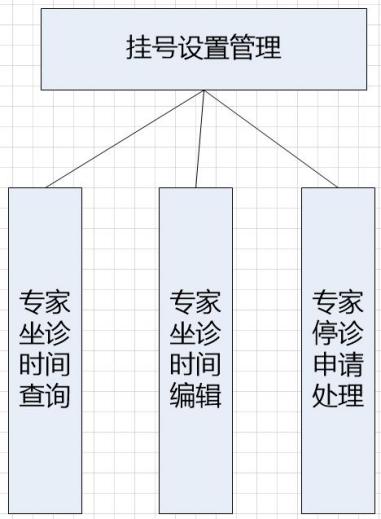
|  |  |
| --- | --- |
| 方法设计 | //专家编辑  public function edit(){} |
| 处理逻辑 | //如果用户点击了编辑提交按钮，那么则将信息保存在数据库  if(IS\_POST){  $updateData["xxx"]=I("post.xxx");  $updateModel=D("doctor");  //这里用了自动验证  if(!$updateModel->create($updateData)){  $this->error($updateModel->getError());  }else{  $updateModel->where("doc\_code=".I("get.doc\_code"))->save($updateData);  }  }  //获取  $doc\_code=I("get.doc\_code");  $selectModel=M("doctor");  $selectData=$selectModel->where("doc\_code=".$doc\_code)->find();  //展示  $this->assign("selectData",$selectData);  $this->display(); |

### 2.2.4 **挂号设置管理**

1.需求概述：

此用例允许医院管理员查看并编辑专家的坐诊时间，并且可以处理专家提交过来的停诊申请。

1. 软件结构：



3.具体描述：

（1）专家坐诊时间查询：

功能：根据医院管理员输入的查询条件，查询出符合要求的专家信息，并显示出专家未来7天的坐诊信息。

输入：专家代码\专家名字，科室名称

输出：专家信息以及未来7天坐诊信息。

（2）专家坐诊时间编辑：

功能：医院管理员在相应专家的未来7天的坐诊信息上点击即可改变专家坐诊状态。

输入：点击专家相应时间的坐诊状态。

输出：坐诊状态更改成功。

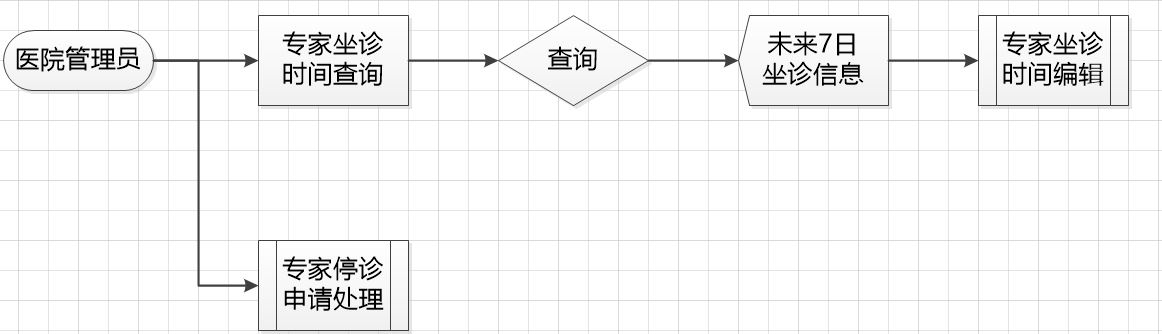
（3）专家停诊申请处理：

功能：医院管理员可以通过该用例查看待审核专家的申请停诊记录，并且可以进行同意\拒绝。

输入：同意\拒绝。

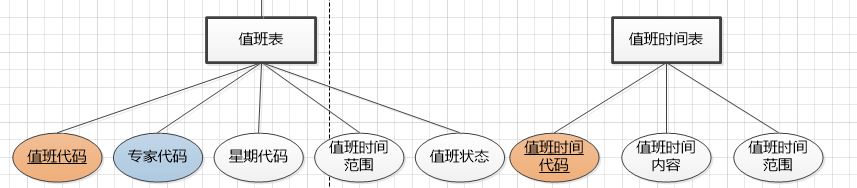
输出：状态更改成功。

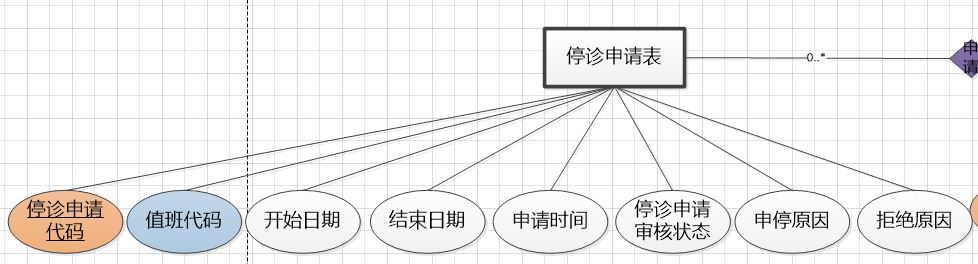
1. 流程图：



5.数据库设计

ER图：



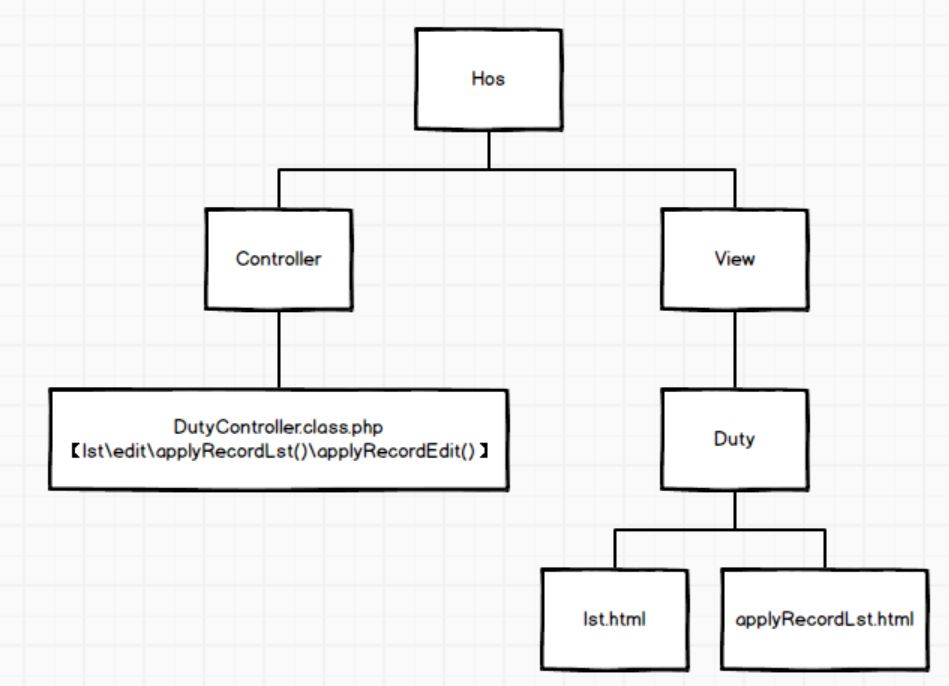


6.程序界面：





7.代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Hos\Controller | DutyController.class.php  【lst(),edit(),applyRecordLst(),applyRecordEdit()】 | 控制器类 |
| Hos\View\Duty | lst.html | 列表模板 |
| Hos\View\Duty | applyRecordLst.html | 列表模板 |

|  |  |
| --- | --- |
| Model类设计 | //hospital表的模型类，用于自动验证  class StopScheduleModel extends Model{} |
| 处理逻辑 | //用于自动验证  protected $\_validate=array(  array("refuse\_reason","checkLength","医院代码应该在50个字内!",0,"callback",3,array(50)),  );  function checkLength($str,$maxlength){  if(strlen($str)<=$maxlength){  return true;  }else{  return false;  }  } |

|  |  |
| --- | --- |
| 方法设计 | //值班表展示  //1.用于第一次刷出这个页面时的整表展示  //2.用于用户输入查询条件时的展示  public function lst(){} |
| 处理逻辑 | 梳理：用户可以传值的字段有2个：专家代码\专家名字，科室名称  专家代码\专家名字有：  科室名称有：  科室名称无：  专家代码\专家名字无：  科室名称有：  科室名称无：  $sql="select \* from dutySchedule";  $selectModel=new Model();  $selectData=$selectModel->query($sql);  $this->assign("selectData",$selectData);  $this->display(); |

|  |  |
| --- | --- |
| 方法设计 | //值班表编辑  public function edit(){} |
| 处理逻辑 | //获取该值班代码对应的值班状态  $dut\_code=I("get.dut\_code");  $selectModel=M("dutySchedule");  $dut\_status=$selectModel->where("dut\_code=".$dut\_code)->getField("dut\_status");  //如果值班状态原先为0，那么现在要改为1；如果值班状态原先为1，那么现在要改为0  $updateModel=M("dutySchedule");  if($dut\_code==0){  $updateData["dut\_status"]=1;  }else{  $updateData["dut\_status"]=0;  }  $updateModel->where("dut\_code=".$dut\_code)->field("dut\_status")->save($updateData); |

|  |  |
| --- | --- |
| 方法设计 | //停诊申请表展示  public function applyRecordLst(){} |
| 处理逻辑 | //展示出停诊申请表中待审核的记录  $sql="select \* from stopSchedule where stop\_state=1";  $selectModel=new Model();  $selectData=$selectModel->query($sql);  $this->assign("selectData",$selectData);  $this->display(); |

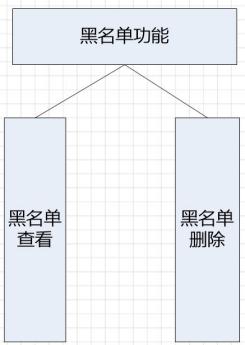
|  |  |
| --- | --- |
| 方法设计 | //停诊申请处理  public function applyRecordEdit(){} |
| 处理逻辑 | //获取停诊申请代码  $action=I("get.action");  //获取操作提示，如果action=no则将stop\_state状态改为2,如果action=yes则将stop\_state状态改为3  if($action=="no"){  $updateData["stop\_status"]=2;  }else{  $updateData["stop\_status"]=3;  }  $updateModel=M("stopSchedule");  $stop\_code=I("get.dut\_code");  $updateModel->where("stop\_code=".$stop\_code)->field("stop\_status")->save($updateData); |

### 2.2.5 **黑名单功能**

1. 需求概述：

医院管理员登录到系统之后，可以查看黑名单中的患者；对于表现良好的患者，医院管理员可以将其从黑名单中删除。

1. 软件结构：



3.具体描述：

（1）黑名单查看：

功能：医院管理员进入黑名单功能模块，即可查看黑名单。

输入：无。

输出：黑名单。

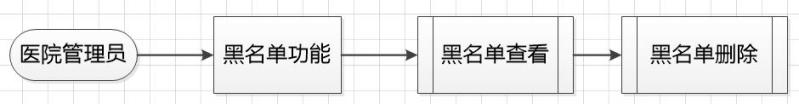
（2）黑名单删除：

功能：医院管理员可通过点击黑名单列表中相应患者删除按钮，即可将该患者从黑名单中删除。

输入：点击删除按钮。

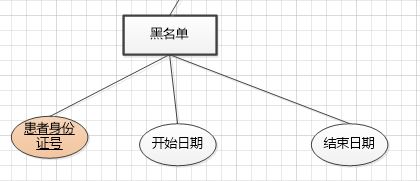
输出：移除成功提示信息。

1. 流程图：



5.数据库设计

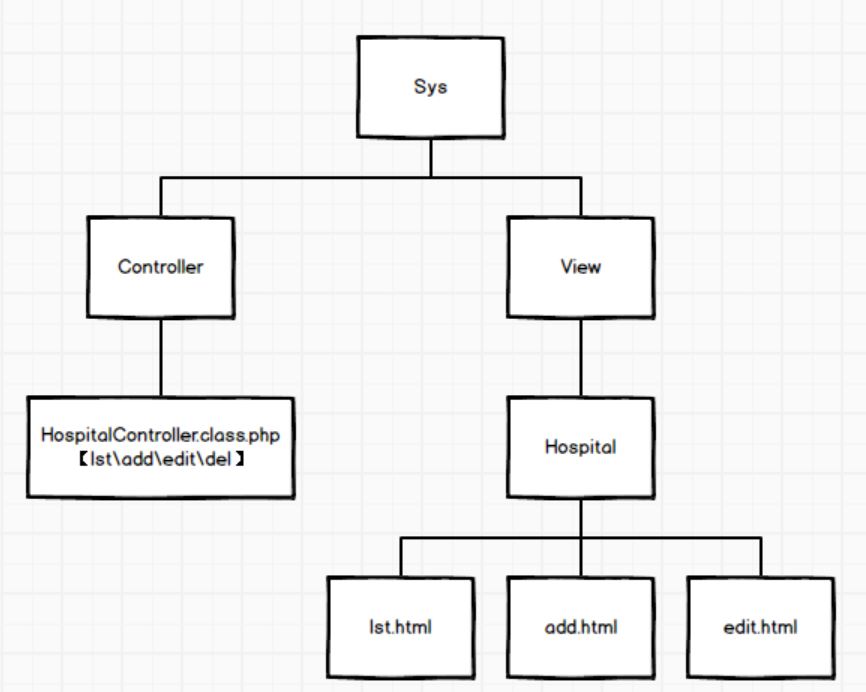
ER图：



1. 程序界面：



1. 代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Hos\Controller | BlacklistController.class.php  【lst(),del()】 | 控制器类 |
| Hos\View\Blacklist | lst.html | 列表模板 |

|  |  |
| --- | --- |
| 方法设计 | //黑名单列表展示  public function lst(){} |
| 处理逻辑 | $selectModel=M("blacklist");  $selectData=$selectModel->select();  $this->assign("selectData",$selectData);  $this->display(); |

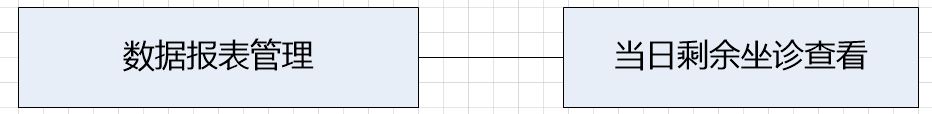
|  |  |
| --- | --- |
| 方法设计 | //删除某一条blacklist表中的记录  public function del(){} |
| 处理逻辑 | $p\_id=I("get.p\_id");  $deleteModel=M("blacklist");  $deleteModel->delete($p\_id); |

### 2.2.5 **数据报表管理**

1. 需求概述：

此用例允许医院管理员查看当日的剩余坐诊列表。

1. 软件结构：



3.具体描述：

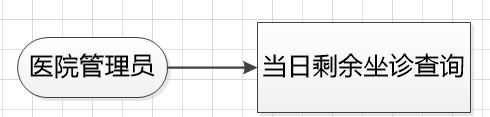
（1）当日剩余坐诊查询：

功能：医院管理员进入数据报表管理模块即可查看当日剩余坐诊列表。

输入：无。

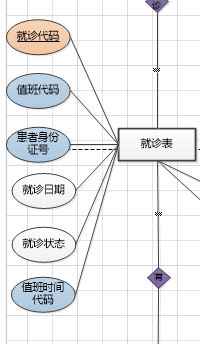
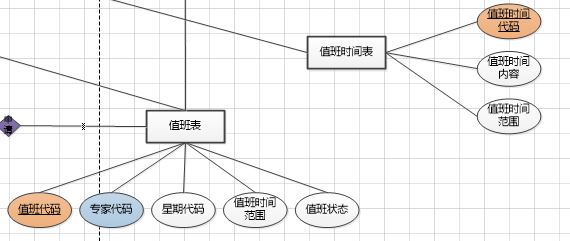
输出：当日剩余坐诊列表。

1. 流程图：



5.数据库设计

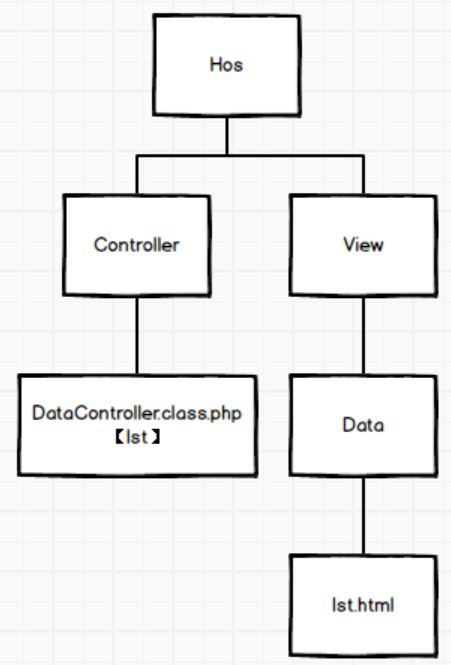
ER图：

6.程序界面：



1. 代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Hos\Controller | DataController.class.php  【lst()】 | 控制器类 |
| Hos\View\Data | lst.html | 列表模板 |

|  |  |
| --- | --- |
| 方法设计 | //当时剩余挂号列表展示  public function lst(){} |
| 处理逻辑 | $sql="xxx";  $selectModel=new Model();  $selectData=$selectModel->query($sql);  $this->assign("selectData",$selectData);  $this->display(); |

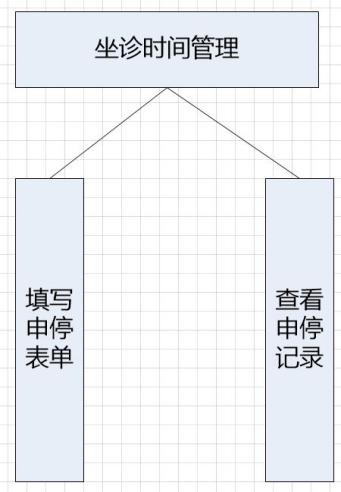
## 2.3医院专家用例

### 2.3.1 **坐诊时间管理**

1. 需求概述：

该医院专家可以通过此用例填写申停表单，并且可以查看申停记录。

1. 软件结构：



3.具体描述：

（1）填写申停表单：

功能：医院专家可以通过该用例提交自己的申停表单。

输入：填写申停表单。

输出：申停已发送。

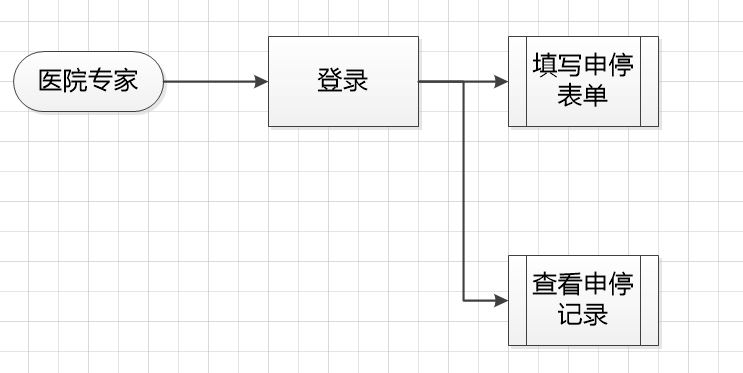
（2）查看申停记录：

功能：医院专家可以通过该用例查看自己的申停记录。

输入：无。

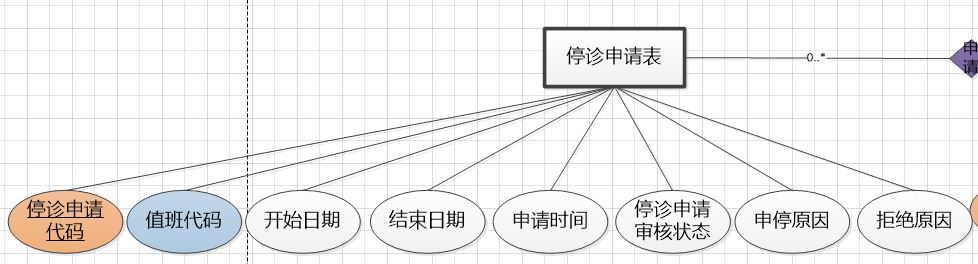
输出：申停记录。

1. 流程图：



5.数据库设计

ER图：



1. 程序界面：

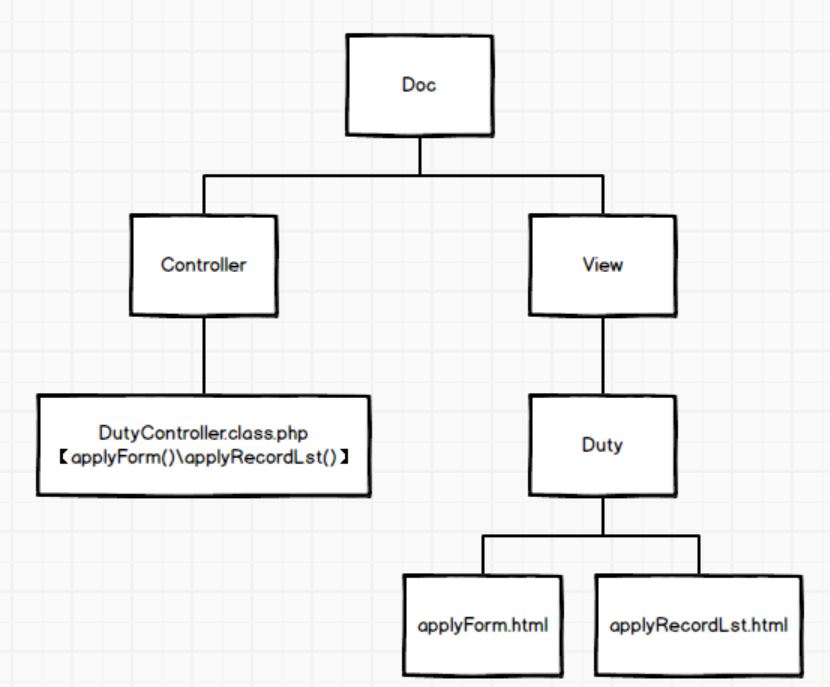
申请停诊表单：



申停记录：



7.代码结构：



|  |  |  |  |
| --- | --- | --- | --- |
| 文件路径 | | 文件名称 | 文件说明 |
| Doc\Controller | | DutyController.class.php  【applyForm(),applyRecordLst()】 | 控制器类 |
| Doc\View\Duty | | applyForm.html | 列表模板 |
| Doc\View\Duty | | applyRecordLst.html | 列表模板 |
| Model类设计 | //hospital表的模型类，用于自动验证  class StopScheduleModel extends Model{} | | | |
| 处理逻辑 | //用于自动验证  protected $\_validate=array(  array("apply\_reason","checkLength","申停原因应该在50个字内!",0,"callback",3,array(50)),  );  function checkLength($str,$maxlength){  if(strlen($str)<=$maxlength){  return true;  }else{  return false;  }  } | | | |

|  |  |
| --- | --- |
| 方法设计 | //展示申停表单，并在提交时向stopSchedule表中添加一条记录。  public function applyForm(){} |
| 处理逻辑 | if(IS\_POST){  $addData["xxx"]=I("post.xxx"); //申请时间是当前时间  $addModel=M("hospital");  $addModel->add($addData);  }  $this->assign();  $this->display(); |

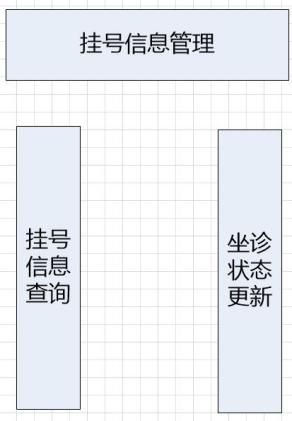
|  |  |
| --- | --- |
| 方法设计 | //申停记录展示  public function applyRecordLst(){} |
| 处理逻辑 | $sql="select \* from hospital"; //注意要筛选出自己的记录  $selectModel=new Model();  $selectData=$selectModel->query($sql);  $this->assign("selectData",$selectData);  $this->display(); |

### 2.3.2 **挂号信息管理**

1. 需求概述：

此用例允许专家登录到系统之后查看当日患者的挂号情况并且进行坐诊状态的更新。

1. 软件结构：



3.具体描述：

（1）挂号信息查询：

功能：专家登录到系统之后在挂号信息功能模块即可查看当日自己患者的挂号信息。

输入：无。

输出：当日自己患者的挂号信息。

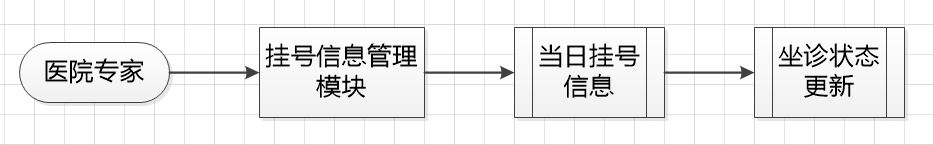
（2）坐诊状态更新：

功能：在当日自己患者的挂号信息列表中点击相应患者的已就诊/未就诊按钮即可更新。

输入：点击已就诊/未就诊按钮。

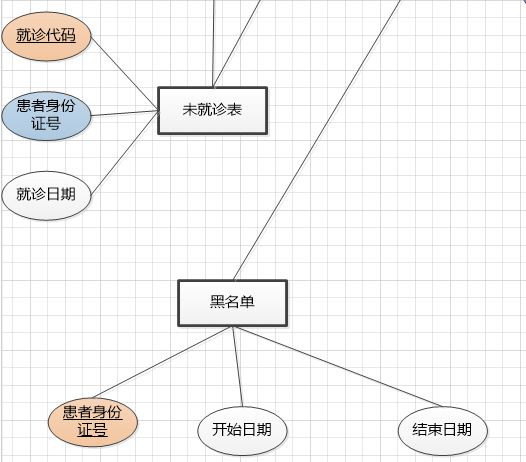
输出：状态更新成功提示信息。

1. 流程图：



5.数据库设计

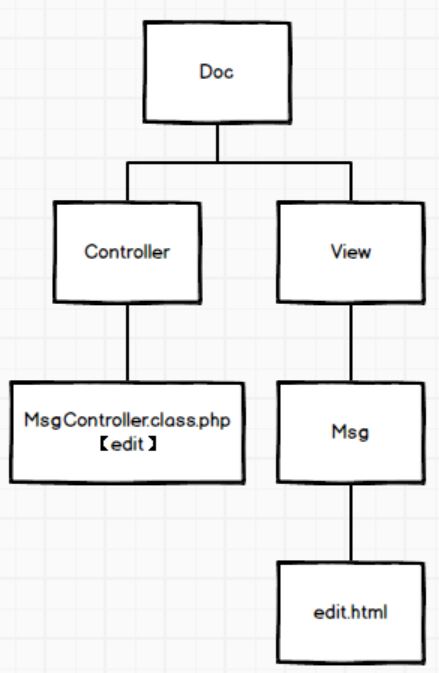
ER图：



6.程序界面：



7.代码结构：



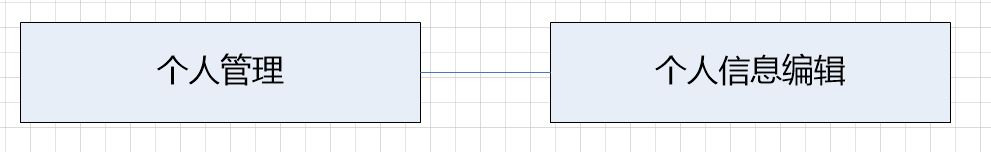
|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Doc\Controller | MsgController.class.php  【edit()】 | 控制器类 |
| Doc\View\Msg | edit.html | 列表模板 |

|  |  |
| --- | --- |
| 方法设计 | //挂号信息展示以及状态更新逻辑  public function edit(){} |
| 处理逻辑 | //查出该专家当日的就诊表中的信息（已就诊和未就诊的就不能再有状态更新了）  //专家点击已就诊：将该就诊列表项的就诊状态改为已就诊  //专家点击未就诊：1.将该就诊列表项的就诊状态改为未就诊  2 查询untreatment表中是否有该患者，  如果untreatment表中有该患者的0/1条记录，那么就将该患者插入untreatment表中；  如果Untreatment表中有该患者的2条记录，那么将该患者插入blacklist表中，并且删除untreatment表中患者的2条记录。 |

### 2.3.3 **个人管理**

1. 需求概述：此用例允许医院专家登录系统后编辑自己的个人信息。

2.软件结构：



3.具体描述：

（1）个人信息编辑：

功能：医院专家登录系统之后进入个人管理模块即可编辑。

输入：修改后的个人信息。

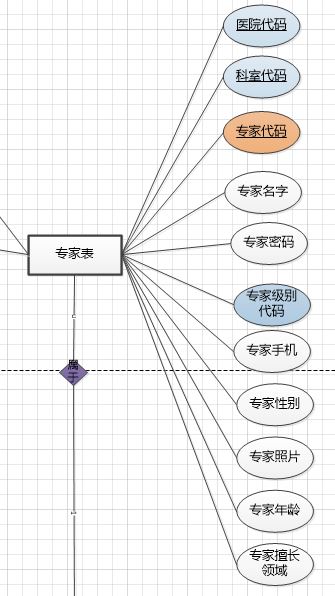
输出：修改成功提示信息。

1. 流程图：



1. 数据库设计

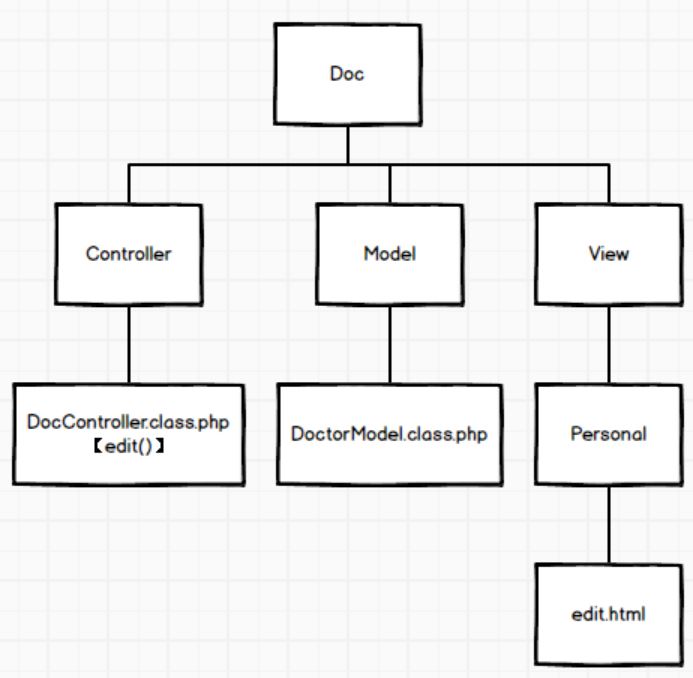
ER图：



6.程序界面：



7.代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Hos\Controller | DocController.class.php  【edit()】 | 控制器类 |
| Hos\View\Personal | edit.html | 编辑页面模板 |

|  |  |
| --- | --- |
| Model类设计 | //doctor表的模型类，用于自动验证  class DoctorModel extends Model{} |
| 处理逻辑 | //用于自动验证  protected $\_validate=array(  array("doc\_code","checkLength","专家代码应该在20个字内!",0,"callback",3,array(20)),  array("doc\_name","checkLength","专家名字应该在20个字内!",0,"callback",3,array(20)),  array("doc\_pwd","checkLength","专家密码应该在10个字内!",0,"callback",3,array(10)),  array("doc\_tel","/^1[3578][0-9]{9}$/","手机号格式错误!",0,"regex",3),  array("doc\_area","checkLength","擅长领域应该在200个字内!",0,"callback",3,array(200)),  );  function checkLength($str,$maxlength){  if(strlen($str)<=$maxlength){  return true;  }else{  return false;  }  } |

|  |  |
| --- | --- |
| 方法设计 | //专家编辑  public function edit(){} |
| 处理逻辑 | //如果用户点击了编辑提交按钮，那么则将信息保存在数据库  if(IS\_POST){  $updateData["xxx"]=I("post.xxx");  $updateModel=D("doctor");  //这里用了自动验证  if(!$updateModel->create($updateData)){  $this->error($updateModel->getError());  }else{  $updateModel->where("doc\_code=".I("get.doc\_code"))->save($updateData);  }  }  //获取  $doc\_code=I("get.doc\_code");  $selectModel=M("doctor");  $selectData=$selectModel->where("doc\_code=".$doc\_code)->find();  //展示  $this->assign("selectData",$selectData);  $this->display(); |

## 2.4 患者用例

### 2.4.1 **登录**

1. 需求概述：此用例允许患者登录到该系统。

2.软件结构：

3.具体描述：

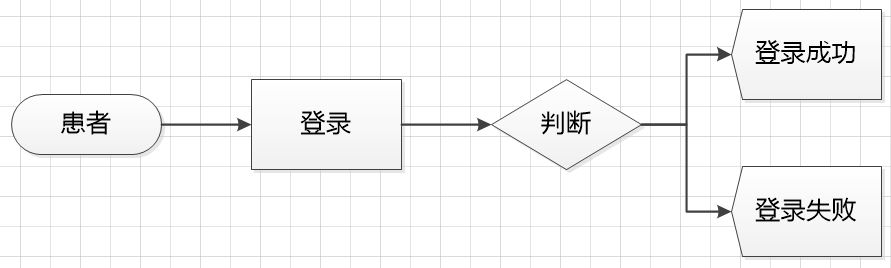
（1）登录：

功能：此用例允许患者登录到该系统。

输入：姓名，身份证号，手机号

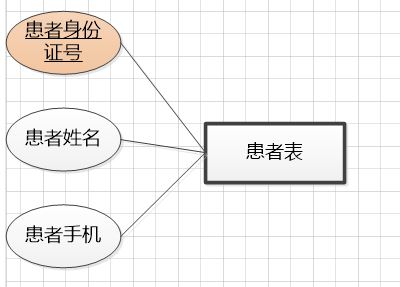
输出：登录成功

1. 流程图：



5.数据库设计

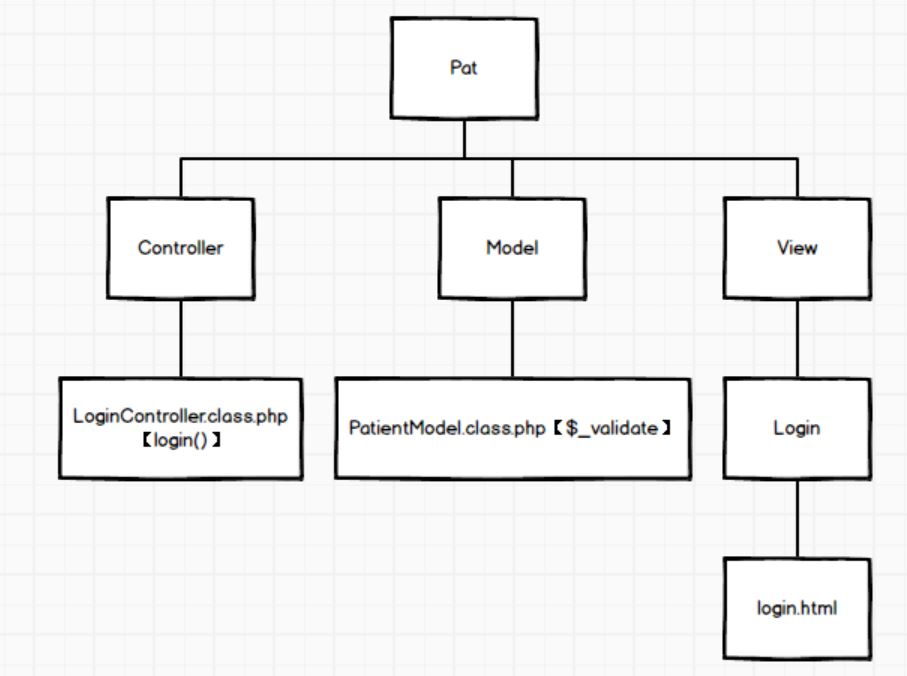
ER图：



6.程序界面：



7.代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Pat\Controller | LoginController.class.php  【login()】 | 控制器类 |
| Pat\Model | PatientModel.class.php | 模型类 |
| Pat\View\Login | login.html | 登录模板 |

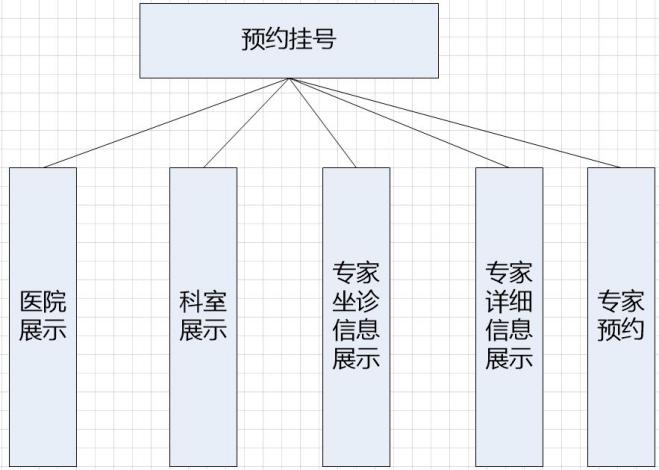
|  |  |
| --- | --- |
| Model类设计 | //patient表的模型类，用于自动验证  class PatientModel extends Model{} |
| 处理逻辑 | //用于自动验证  protected $\_validate=array(  array("p\_name","checkLength","名字应该在20个字内!",0,"callback",3,array(20)),  array("p\_tel","/^1[3578][0-9]{9}$/","手机号格式错误!",0,"regex",3),  );  function checkLength($str,$maxlength){  if(strlen($str)==0){  return false;  }else{  if(strlen($str)<=$maxlength){  return true;  }else{  return false;  }  }  }  function checkFixedLength($str,$fixedlength){  if(strlen($str)!=$fixedlength){  return false;  }else{  return true;  }  } |

|  |  |
| --- | --- |
| 方法设计 | //展示登录页面，如果点击登录进行登录逻辑并会进行自动验证  public function login(){} |
| 处理逻辑 | if(IS\_POST){  //获取患者身份证号  $p\_id=I("post.p\_id");  $selectModel=M("patient");  $selectData=$selectModel->where("p\_id=".$p\_id)->getField("p\_id");  if($selectData==null){  //说明patient表中没有该患者  //\*\*\*\*\*\*\*  $addData["xxx"]=I("post.xxx");  $addModel=D("patient");  //这里用了自动验证  if(!$addModel->create($addData)){  $this->error($addModel->getError());  }else{  $addModel->add();  }  //\*\*\*\*\*\*\*  }else{  //说明patient表中已经有该患者了，那么就什么也不做  }  //跳转到首页  header();  return;  }  //展示登录页面  $this->assign();  $this->display(); |

### 2.4.2 **预约挂号**

1. 需求概述：该用例允许登录后的患者可通过医院筛选，科室筛选，专家筛选从而预约专家。

2.软件结构：



3.具体描述：

（1）医院展示：

功能：患者进入该系统的首页即为医院展示。

输入：无。

输出：各个医院的信息。（轮播图，列表）

（2）科室展示：

功能：患者可通过该用例了解选择的医院的科室情况。

输入：已经选择了医院。

输出：科室列表。

（3）专家坐诊信息展示：

功能：患者在选择了医院以及科室后会进入专家坐诊信息页面。

输入：已经选择了科室。

输出：专家坐诊信息列表。

（4）专家详细信息展示：

功能：患者可以通过点击专家坐诊信息列表中的专家连接即可进入专家详情页面。

输入：点击专家连接。

输出：专家详细信息。

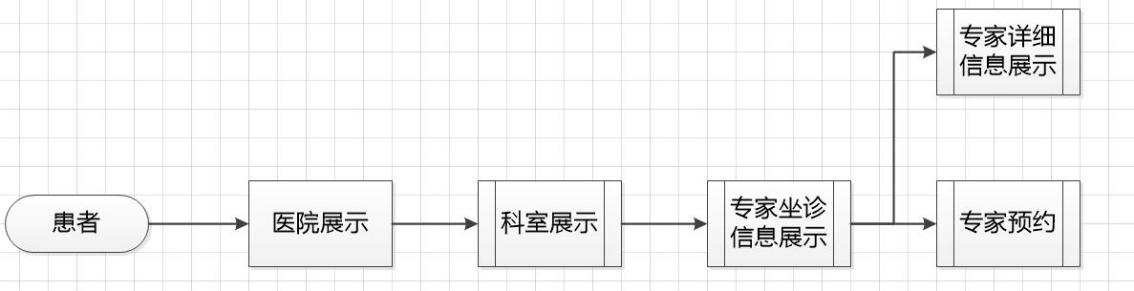
1. 专家预约：

功能：患者可通过该用例实现预约某一个专家。

输入：在专家坐诊信息展示页面可以点击某一个专家的某一天的上午/下午里的预约按钮。

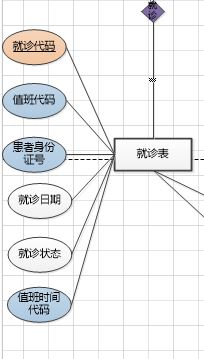
输出：预约结果。

1. 流程图：



5.数据库设计

ER图：



1. 程序界面：

医院展示：



科室展示：



专家坐诊信息展示：



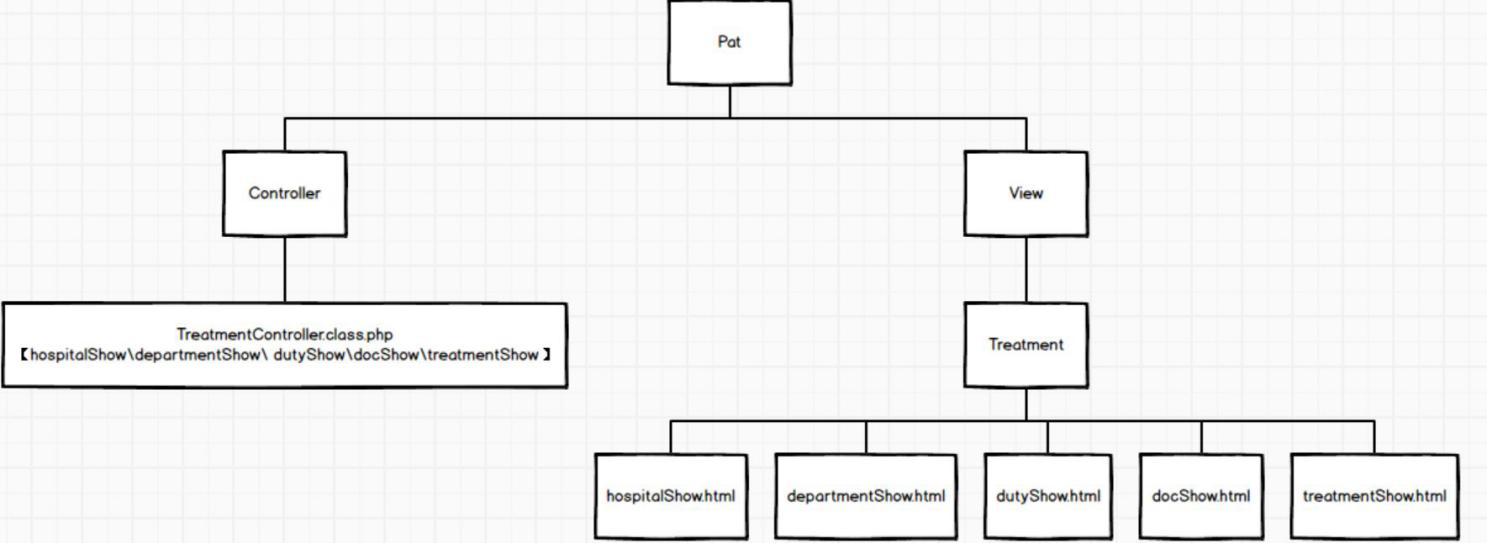
专家详细信息：



专家预约：



1. 代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Pat\Controller | TreatmentController.class.php  【hospitalShow\departmentShow\dutyShow\  docShow\treatmentShow】 | 控制器类 |
| Pat\View\Treatment | hospitalShow.html | 列表模板 |
| Pat\View\Treatment | departmentShow.html | 列表模板 |
| Pat\View\Treatment | dutyShow.html | 列表模板 |
| PatAdmin\View\Treatment | docShow.html | 模板 |
| PatAdmin\View\Treatment | treatmentShow.html | 模板 |

|  |  |
| --- | --- |
| 方法设计 | //医院展示  //1.用于第一次刷出这个页面时的整表展示  //2.用于用户输入查询条件时的展示  public function hospitalShow(){} |
| 处理逻辑 | 梳理：用户可以传值的字段有3个：医院名称，行政区划，医院级别  医院代码\名称有：  行政区划有：  医院级别有：  医院级别无：  行政区划无：  医院级别有：  医院级别无：  医院代码\名称无：  行政区划有：  医院级别有：  医院级别无：  行政区划无：  医院级别有：  医院级别无：  $sql="select \* from hospital";  $selectModel=new Model();  $selectData=$selectModel->query($sql);  $this->assign("selectData",$selectData);  $this->display(); |

|  |  |
| --- | --- |
| 方法设计 | //科室展示  public function departmentShow(){} |
| 处理逻辑 | //获取传入的医院代码  $hos\_code=I("get.hos\_code");  //获取相应的医院信息  $selectModel1=M("hospital");  $selectData1=$selectModel1->where("hos\_code=".$hos\_code)->find();  //获取相应医院的所有科室信息  $selectModel2=M("department");  $selectData2=$selectModel2->where("hos\_code=".$hos\_code)->select();  $this->assign("selectData1",$selectData1);  $this->assign("selectData2",$selectData2);  $this->display(); |

|  |  |
| --- | --- |
| 方法设计 | //专家坐诊信息展示  public function dutyShow(){} |
| 处理逻辑 | //获取传入的医院代码  $hos\_code=I("get.hos\_code");  //获取传入的科室代码  $dep\_code=I("get.dep\_code");  //获取相应的科室信息  $selectModel1=M("department");  $selectData1=$selectModel1->where("hos\_code=".$hos\_code." and dep\_code=".$dep\_code)->find();  //获取相应科室下的所有专家信息  $selectModel2=M("doctor");  $selectData2=$selectModel2->where("hos\_code=".$hos\_code." and dep\_code=".$dep\_code)->select();  $this->assign("selectData1",$selectData1);  $this->assign("selectData2",$selectData2);  $this->display(); |

|  |  |
| --- | --- |
| 方法设计 | //专家详细信息展示  public function docShow(){} |
| 处理逻辑 | //获取传入的专家代码  $doc\_code=I("get.doc\_code");  //获取相应的专家详细信息  $selectModel=M("doctor");  $selectData=$selectModel->where("doc\_code=".$doc\_code)->find();  $this->assign("selectData",$selectData);  $this->display(); |

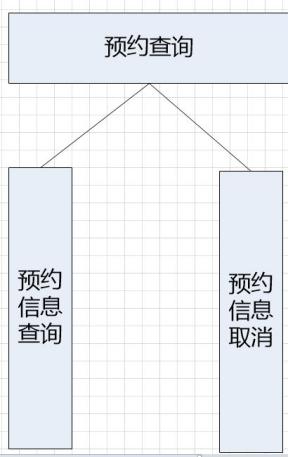
|  |  |
| --- | --- |
| 方法设计 | //专家预约  public function treatmentShow(){} |
| 处理逻辑 | //获取传入的值班代码  $dut\_code=I("get.dut\_code");  //获取相应的值班信息  $selectModel=M("dutySchedule");  $selectData=$selectModel->where("dut\_code=".$dut\_code)->find();  $this->assign("selectData",$selectData);  $this->display(); |

### 2.4.3 **预约查询**

1. 需求概述：

此用例允许患者登录到系统之后查看自己的专家预约情况，也可以取消预约。

1. 软件结构：



3.具体描述：

（1）预约信息查询：

功能：患者登录到系统之后就可以查看自己的预约信息。

输入：无。

输出：预约信息列表。

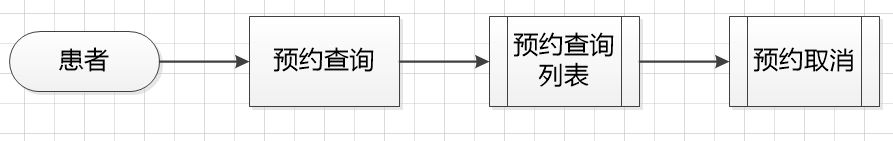
（2）预约信息取消：

功能：查看自己的预约信息后可以点击取消按钮从而取消预约。

输入：点击取消按钮。

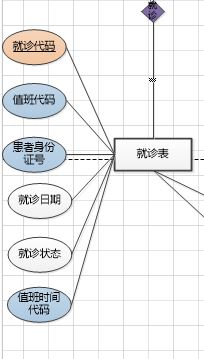
输出：取消成功提示信息。

1. 流程图：



5.数据库设计

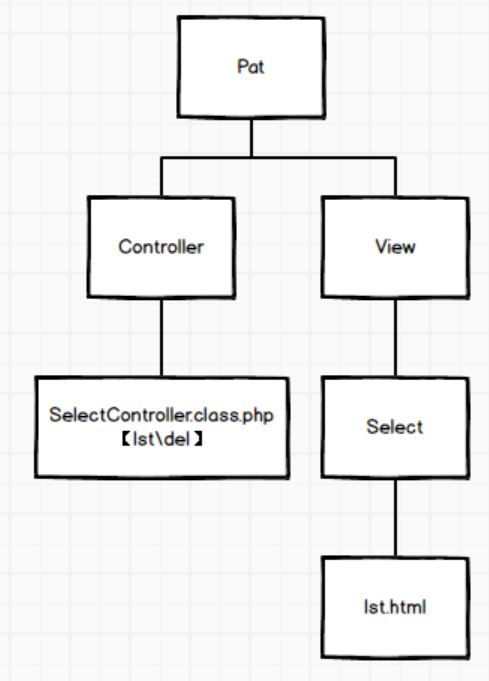
ER图：



6.程序界面：



7.代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Pat\Controller | SelectController.class.php  【lst(),del()】 | 控制器类 |
| Pat\View\Select | lst.html | 列表模板 |

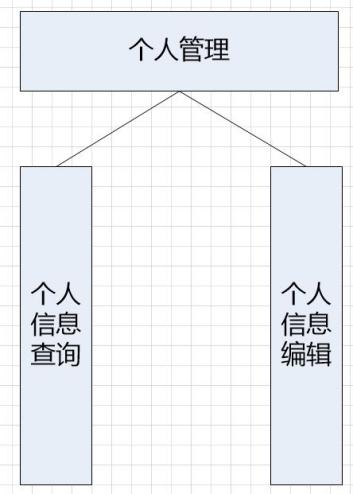
|  |  |
| --- | --- |
| 方法设计 | //预约查询列表展示  public function treatmentShow(){} |
| 处理逻辑 | //获取传入的患者身份证号  $p\_id=I("get.p\_id");  //获取该患者的明天及以后的预约信息  $selectModel=M("treatment");  $selectData=$selectModel->where("p\_ide=".$p\_id)->select();  $this->assign("selectData",$selectData);  $this->display(); |

|  |  |
| --- | --- |
| 方法设计 | //删除某一条treatment表中的记录  public function del(){} |
| 处理逻辑 | $t\_code=I("t.hos\_code");  $deleteModel=M("treatment");  $deleteModel->delete($t\_code); |

### 2.4.4 **个人管理**

1. 需求概述：此用例允许患者登录系统后编辑自己的个人信息。

2.软件结构：



3.具体描述：

（1）个人信息查询：

功能：患者登录系统之后进入个人管理模块即可查看个人信息。

输入：无。

输出：个人信息。

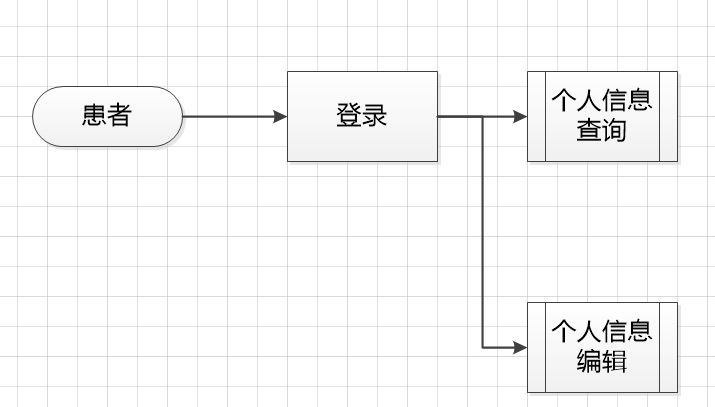
（2）个人信息编辑：

功能：患者登录系统之后进入个人管理模块即可修改。（只能修改手机号）

输入：修改后的个人信息。

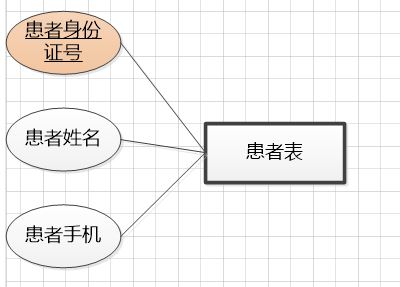
输出：修改成功提示信息。

4.流程图：



5.数据库设计

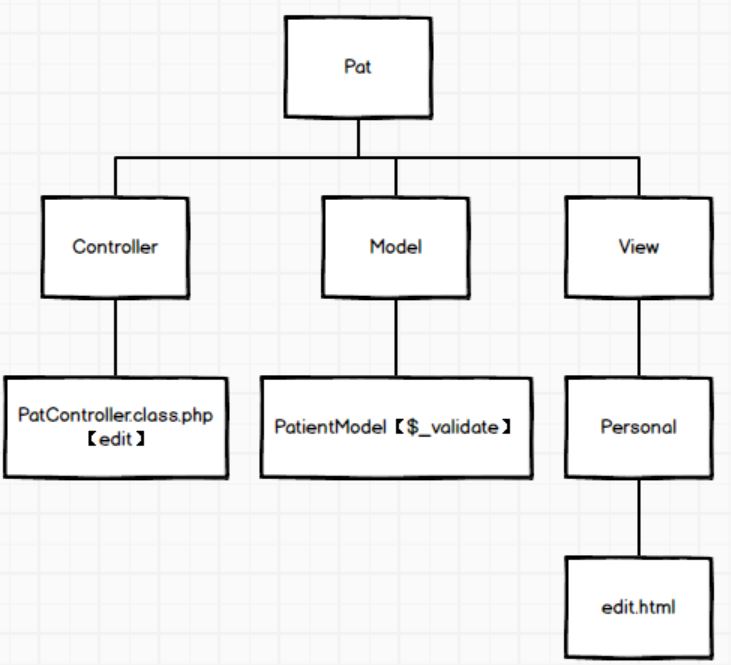
ER图：



6.程序界面：



7.代码结构：



|  |  |  |
| --- | --- | --- |
| 文件路径 | 文件名称 | 文件说明 |
| Pat\Controller | PatController.class.php  【edit()】 | 控制器类 |
| Pat\Model | PatientModel.class.php | 模型类 |
| Pat\View\Personal | edit.html | 列表模板 |

|  |  |
| --- | --- |
| Model类设计 | //patient表的模型类，用于自动验证  class PatientModel extends Model{} |
| 处理逻辑 | //用于自动验证  protected $\_validate=array(  array("p\_tel","/^1[3578][0-9]{9}$/","手机号格式错误!",0,"regex",3),  ); |

|  |  |
| --- | --- |
| 方法设计 | //患者编辑  public function edit(){} |
| 处理逻辑 | //如果用户点击了编辑提交按钮，那么则将信息保存在数据库  if(IS\_POST){  $updateData["xxx"]=I("post.xxx");  $updateModel=D("patient");  //这里用了自动验证  if(!$updateModel->create($updateData)){  $this->error($updateModel->getError());  }else{  $updateModel->where("p\_id=".I("get.p\_id"))->save($updateData);  }  }  //获取  $p\_id=I("get.p\_id");  $selectModel=M("patient");  $selectData=$selectModel->where("p\_id=".$p\_id)->find();  //展示  $this->assign("selectData",$selectData);  $this->display(); |

### 2.4.5 **系统公告**