**VUE**

# 一、VUE

## 1、MVC

Model view controller 数据与view 分离，一切关注数据 ,不用关注 dom的大量的操作，进行解耦

mvc 🡪mvp🡪 mvvm 🡪mv\*

## 2、数据绑定

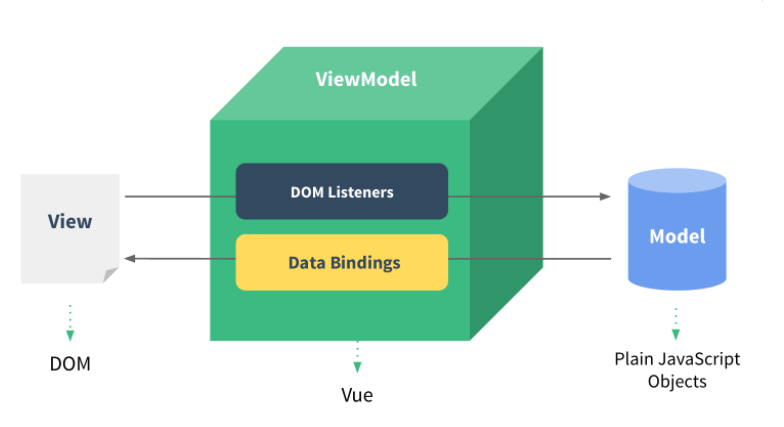
渲染dom上 ，

单向数据绑定 (一端操作): model 🡪view view 🡪mdel

双向数据绑定 : model 🡨controller🡪view ,原理即为监听 数据

## 3、vue框架





# (二)VUE的基础知识

## 1、模板绑定

{{}}:绑定多次，没有值为空，不解析html

{{\*}}:绑定一次，没有值为undefined 不解析html

{{{}}}:绑定多次，没有值为空，解析html

## 2、el

任意的选择器…. 元素 id 类选择器 ，表示VUE 使用范围

## 3、data

字符串 数字 布尔 数组 对象

# 指令

## 1、指令

### 1)、v-model

dom关联model的 ，常用于 input 文本 、textarea上

### 2)、v-text 、v-html

绑定数据到dom :

* 不会存在延时的闪烁问题
* 全部替换所有的内容， {{}}模板可以指定部分
* v-text：不会解析 html v-html:解析html

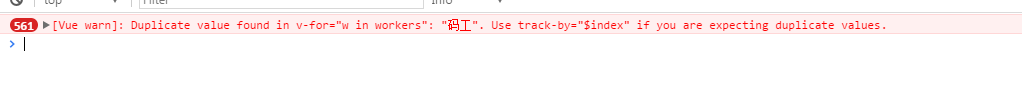
### 3)、v-if v-show v-else

v-if: 插入|删除元素 v-else

v-show:显示与隐藏元素 v-else

### 4)、v-for

* v-for 用于循环 数组|json对象的数据
* v-for=”t in 数组” v-for=”(k,v) in 对象”
* $index ${key}
* 不能存在重复元素 ，使用track-by=”$index”



### 5)、v-cloak

防止闪烁， 搭配 [v-cloak] 样式使用

|  |
| --- |
| <style>  [v-cloak]{  display: none !important;  }  </style> |
| <ul >  <li v-for="w in workers" track-by="$index" v-cloak>{{$index}} -->{{w}}</li>  </ul> |

## 2、属性

vue的指令扩展了属性 如 v-bind:src v-bind:width :height

class 与 style

### 1)、class

类样式，常用的形式

a)、非vue ： class=”类名1”;

b)、 :class=”’类名1’” 🡪注意 ‘’

c)、 :class=”数据名”

d)、:class=”[数据名]”

e)、:class=”[数据名1,数组名2]”

f)、 :class=”{类名1: true|false ,类名2:true|false}”

### 2)、style

与class 同理 ，需要json格式的数据,样式中: 复合样式，**需要使用驼峰命名法**

|  |
| --- |
| <link rel="stylesheet" href="../css/demo.css"/>  <script>  $(function(){  new Vue({  el:'#app',  data:{  total:10,  current:4,  pageClass:'pagination'  }  });  });  </script>  </head>  <body>  <div id="app">  <ul :class="[pageClass]">  <li v-for="p in total" track-by="$index">  <a href="javascript:void(0)"  :class="current==(p+1)?'active':''">{{p+1}}</a></li>  </ul>  </div>  </body> |

## 3、事件

与原生js事件不互通…..

1)、与原生的js不互通

2)、v-on:事件 省略为 @事件

3)、$event :事件对象

4)、阻止冒泡

a)、原生的 ： ev.cancelBubble=true;

b)、事件.stop: @click.stop

5)、默认行为(右键):

ev.preventDefault();

@contextmenu.prevent

6)、键盘事件: @keydown @keyup +keycode

event.keyCode

@keydown|@keyup.键码 :up down left right enter 、26个字母 或直接键码

键码别名: Vue.directive('on').keyCodes.f7=118

# 综合练习

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title>类型管理</title>  <link rel="stylesheet" href="../css/bootstrap.min.css"/>  <link rel="stylesheet" href="../css/bootstrap-theme.min.css"/>  <script src="../js/jquery-1.11.3.js"></script>  <script src="../js/bootstrap.min.js"></script>  <script src="../js/modal-center.js"></script>  <script src="../js/vue.js"></script>  <script>  $(function(){  new Vue({  el:'.container',  data:{  newsTypes:[],  types:{  title:'',  desc:''  },  currentIdx:-1  },  methods:{  addType:function(){  //一切操作数组  this.newsTypes.push(this.types);  this.types={  title:'',  desc:''  };  },  remove:function(){  if(this.currentIdx==-100){  this.newsTypes=[];  }else if(this.currentIdx>=0){  this.newsTypes.splice(this.currentIdx,1);  }  this.currentIdx =-1;  $("#show").modal('hide');  }  }  });  });  </script>  <style>  [v-cloak]{  display: none !important;  }  </style>  </head>  <body>  <!--  1、加入数组 遍历数组  2、隐藏 表格 和 全部删除 显示 暂无数据  3、添加数据 绑定数据 使用对象 + 添加事件  4、删除 记录位置  5、删除所有  -->  <div class="container">  <form role="form">  <div class="form-group">  <label for="tname">类型:</label>  <input type="text" id="tname" class="form-control"  placeholder="输入类型" v-model="types.title"/>  </div>  <div class="form-group">  <label for="tdes">说明</label>  <input type="text" id="tdes" class="form-control"  placeholder="输入说明" v-model="types.desc"/>  </div>  <div class="form-group">  <input type="button" value="添加" class="btn btn-primary" @click="addType();"/>  <input type="button" value="重置" class="btn btn-danger"/>  </div>  </form>  <hr/>  <table class="table table-bordered table-hover" v-cloak>  <caption class="h2 text-center" v-show="newsTypes.length!=0">类别管理</caption>  <tr class="text-center text-muted" v-if="newsTypes.length==0">  <td colspan="4" ><span>暂无数据</span></td>  </tr>  <tr v-else>  <th class="text-center">序号</th>  <th class="text-center">类别名</th>  <th class="text-center">说明</th>  <th class="text-center">操作</th>  </tr>  <tr class="text-center" v-for="nt in newsTypes" track-by="$index">  <td>{{$index+1}}</td>  <td>{{nt.title}}</td>  <td>{{nt.desc}}</td>  <td><button class="btn btn-sm btn-primary"  data-toggle="modal" data-target="#show"  @click="this.currentIdx=$index">删除</button></td>  </tr>  <tr class="text-right" v-show="newsTypes.length!=0">  <td colspan="4"><button class="btn btn-sm btn-danger" data-toggle="modal" data-target="#show"  @click="this.currentIdx=-100">全部删除</button></td>  </tr>  </table>  <!--模态框-->  <div role="dialog" class="modal fade" id="show">  <div class="modal-dialog">  <div class="modal-content">  <div class="modal-header">  <button class="close" data-dismiss="modal"><span>&times;</span></button>  <h3 class="modal-title">确认删除吗?</h3>  </div>  <div class="modal-body text-right">  <button class="btn btn-primary btn-sm" data-dismiss="modal">取消</button>  <button class="btn btn-danger btn-sm" @click="remove()">删除</button>  </div>  </div>  </div>  </div>  </div>  </body>  </html> |

# 三、过滤器

对数据的处理 |

## 1、内置的过滤器

|  |
| --- |
| {{'abc'|uppercase}}<br/>  {{'aBc'|lowercase}}<br/>  {{'aBc'|lowercase|capitalize}}  {{120|currency}}  {{120|currency '¥'}} |

## 2、json

|  |
| --- |
| {{person|json}}<br/> |

## 3、debounce

配合事件使用的延时。。。

|  |
| --- |
| <button @click="show | debounce 2000"/> 延迟出现结果</button> |

## 4、limitBy

配合数组使用的，限制出现的个数。 limitBy 个数 起始索引

* v-for=val in arr|limitBy 2
* v-for=val in arr |limitBy 2 1
* v-for=val in arr|limitBy 2 arr.length-2

## 5、orderBy

* 正序 <li v-for="val in arr | orderBy 1">
* 倒序 <li v-for="val in arr | orderBy -1">
* 按属性 <li v-for="val in arr1 | orderBy 'age' -1">
* 按变量 <li v-for="val in arr1 | orderBy data -1">

## 6、filterBy

过滤数据 ，过滤条件如:

* 按内容 <li v-for="val in arr | filterBy 'a' ">
* 按变量 <li v-for="val in arr | filterBy 变量 ">

## 7、自定义

格式:

|  |
| --- |
| Vue.filter(‘名称’,function(input,参数列表){  //处理过程  return 结果;  }); |

## 8、双向过滤

一般情况下，普通的过滤都是 ，过滤model数据后传递给vue ,而，双向数据过滤，还会返回给model本身，修改model的值，即

* 一般数据过滤：数据(model) -> 视图(view)
* 双向数据过滤：model -> view view -> model

|  |
| --- |
| Vue.filter("fTwoWayHtml",{  read:function(input){  return input.replace(/<[^<]+>/g,'');  },  write:function(val){  return val;  }  }); |
| <input type="text" v-model="msg | fTwoWayHtml" /> |

# 四、动画

## 1、自定义动画

1)、transition

在目标元素上加入transition属性，transition="xxx"。

2)、编写css动画 ，一般有三个步骤

a)、定义动画效果 xxx-transition

b)、编写进入样式 xxx-enter

c)、编写离开样式 xxx-leave

## 2、推荐动画

animate.css

<https://daneden.github.io/animate.css/>

1)、使用bower下载即可

安装bower

|  |
| --- |
| npm install -g bower |

2)、bower 下载 animate.css

bower.json

|  |
| --- |
| {  "name": "sxt",  "description": "bower is so easy",  "main": "main.js",  "keywords": [  "sxt"  ],  "authors": [  "laopei"  ],  "license": "MIT",  "homepage": "index.html",  "private": true,  "ignore": [  "\*\*/.\*",  "node\_modules",  "bower\_components",  "js/lib",  "test",  "tests"  ]  } |

.bowerrc

|  |
| --- |
| {  "directory" : "js/lib"  } |

下载：bower install animate.css@3.3.0 -save

|  |
| --- |
|  |

使用

a)、class="animated"

在目标元素上加入class="animated" 样式

b)、transition="xxxx"

在目标元素上加入transition ="xxx" 选择动画，参考官网即可。

c)、transitions

在实例中，使用transitions属性

a)、enterClass: animate.css的class 🡪进入

b)、leaveClass: animate.css的class –>离开

d)、引入

<link rel="stylesheet" href="js/lib/animate.css/animate.min.css"/>

|  |
| --- |
| <script>  new Vue({  el:'#box',  data:{  flag:true  },  transitions:{  sxt:{  enterClass:"zoomInLeft",  leaveClass:"zoomOutRight"  }  },  methods:{  toggle:function(){  this.flag=!this.flag;  }  }  });  </script> |
| <body>  <div id="box">  <input type="button" value="按钮" @click="toggle">  <div id="div1" v-show="flag" class="animated" transition="sxt"></div>  </div>  </body> |
| <link rel="stylesheet" href="js/lib/animate.css/animate.min.css"/>  <style>  #div1{  width:60px;  height:80px;  background: #ccc;  }  </style> |

# 五、计算属性computed

之前vue实例的data中放置普通的属性，不能写业务逻辑代码，而computed中放置根据业务逻辑需要**计算的属性**，computed是内部调用的一个函数。(说白了就是es6的setter与getter的包装)。

如当data中的一个属性，想要随着另一个属性变化时，这种业务逻辑可以写在vue实例中的computed属性中，这种行为叫做计算属性。

|  |
| --- |
| computed:{  //根据需要计算的属性  b:function(){ //内部就是es6的getter访问器  return this.a + 1;  }  } |
| new Vue({  el:'.myclass',  data:{  a:10  // b:0 -->不在此处定义  },  methods:{},  computed:{  //根据需要计算的属性  b:function(){ //内部就是es6的getter访问器  return this.a + 1;  }  }  }); |
| computed:{  //根据需要计算的属性  b: {  get:function(){  return this.a+"sxt";  },  set:function(val){  this.a = val;  }  }  } |

# 六、PHP

解压PHPNow, setup.cmd

## 1、helloWorld

|  |
| --- |
| <?php  echo "hellworld..."."sxt";  ?> |

注意字符串的拼接为 .

## 2、变量

\* 1)、变量以 $(美元符号) 表示。如$box ,box 为变量名

\* 2)、php 中变量不需要事先声明就可以使用。

\* 3)、如果没有$符号，就是字符串， php 中不用 ""就可以表示字符串，当然会有警告信

|  |
| --- |
| <?php  $box=12;  echo $box;  //不要使用没有声明过的变量 没有声明直接使用 ，表示原样的字符串  //变量要确保值存在  echo $box2;  echo 'box';  echo "hellworld..."."sxt";  ?> |

## 3、地址

& 引用传递

|  |
| --- |
| <?php  $box=12;  //获取box的地址  $n=&$box;  $n='change';  echo $n ."-->".$box;  ?> |

## 4、数组

|  |
| --- |
| <?php  $cars=array("Volvo","BMW","BENZ");  //.拼接  echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".\n<br/>";  //使用 implode 拼接  echo implode(',', $cars);  //PHP 关联数组  $name = 'Hello world!';  $arr = array('name'=>$name,'b'=>'456');  echo $arr['name']."-->".$arr['b'];  //遍历  $age=array("Bill"=>"35","Steve"=>"37","Peter"=>"43");  foreach($age as $x=>$x\_value) {  echo "Key=" . $x . ", Value=" . $x\_value;  echo "<br>";  }  ?> |

## 5、控制语句

### 1)、if

|  |
| --- |
| <?php  //控制语句 if  $num =10;  if($num%2==0){  echo "偶数";  }else{  echo "奇数";  }  ?> |

header("Content-type:text/html;charset=GBK"); 🡪中文乱码

### 2)、循环

|  |
| --- |
| /\*  \* 循环:与 js 一样  \*/  for($i=0;$i<10;$i++){  echo $i;  }  echo '<br/>';  for($i=1;$i<=9;$i++){  for($j=1;$j<=$i;$j++){  echo "$i\*$j=".$i\*$j." ";  }  echo "<br/>";  } |

## 6、函数

|  |
| --- |
| <?php  /\*  \* 函数  \*/  function display($str){  echo $str;  }  //调用函数  display("shsxt");  ?> |

## 7、html 与 php 混编

|  |
| --- |
| <ul>  <?php for($i=0;$i<10;$i++){ ?>  <li><?php echo $i+1; ?></li>  <?php } ?>  </ul> |

当然越灵活，带来的问题也越大，不容易维护。前端人员一看，都是 php 代码，看不

懂。所以提出，”分离”思想，即 MVC。MVC 全名是 Model View Controller，是模型(model)

－视图(view)－控制器(controller)的缩写，即将业务逻辑、数据、界面显示分离。

当然这部分属于 php 深入部分，mvc 框架如 smarty，分离逻辑与显示。

## 8、表单提交

$\_GET["表单的文本框name"] $\_POST["表单的文本框name"] $\_REQUEST[文本框name];

|  |
| --- |
| <?php  //获取get参数  $uname=$\_GET["uname"];  $upwd=$\_GET["pwd"];  echo "success...".'-->'.$uname.'-->'.$upwd;  ?> |
| <form action="data.php" method="get">  用户：<input type="text" name="uname" />  密码：<input type="password" name="pwd" />  <input type="submit" value="提交"/>  </form> |

## 9、ajax提交

## 10、面向对象

|  |
| --- |
| <?php  class Person{  //构造器  function \_\_construct($name, $sex){  $this->name=$name;  $this->sex=$sex;  }  function getName() {  echo $this->name;  }  function getSex(){  echo $this->sex;  }  }  //继承  class Programmer extends Person{  function \_\_construct($name, $sex, $job){  //父类构造器  parent::\_\_construct($name, $sex);  $this->job=$job;  }  function getJob(){  echo $this->job;  }  }  $p=new Person('sxt', '男');  $p->getSex();  $w=new Programmer('sxt', '男', '码农');  $w->getJob();  ?> |

：：双冒号操作符即作用域限定操作符 Scope Resolution Operator 可以访问静态、const 和类中重写的属性与方法。

|  |
| --- |
| class Fruit {  const CONST\_VALUE = 'Fruit Color';  protected function showColor() {  echo "Fruit::showColor()\n";  }  }  //访问静态属性  echo Fruit::CONST\_VALUE.'<br/>';  //parent self :: -->常量、静态信息、父类构造器、重写方法  class Apple extends Fruit{  public static $color = 'Red';  public static function doubleColor() {  echo parent::CONST\_VALUE . "\n";  echo self::$color . "\n";  }  public function showColor(){  //调用父类的方法  parent::showColor();  echo "Apple::showColor()\n";  }  }  //调用静态方法  Apple::doubleColor();  $apple = new Apple();  $apple->showColor(); |

$ 🡪变量

. 🡪拼接

=>键值对

&🡪取地址

->从属

:: 限定符

\_\_:构造器

# 七、资源交互

## 1、引入正确的js

vue-resource.js

支持请求方式: GET POST JSONP ，注入 $http

## 2、GET请求

|  |
| --- |
| <script src="../js/jquery-1.11.3.js"></script>  <script src="../js/vue.js"></script>  <script src="../js/vue-resource.js"></script>  <script>  $(function(){  new Vue({  el:'body',  data:{  msg:''  },  methods:{  getDatas:function(){  //提交请求 $http  var url="test.txt";  this.$http.get(url).then(function(succ){  this.msg =succ.data;  },function(){});  }  }  });  });  </script>  </head>  <body>  <button @click="getDatas">获取数据</button>  {{msg}}  </body> |
| 请求资源: test.txt  abc success |

请求参数

|  |
| --- |
| var url="data.php";  var datas={uname:'sxt',pwd:'123456'};  this.$http.get(url,datas).then(function(succ){  this.msg =succ.data;  }); |
| 请求资源: data.php  <?php  header("Content-type:text/html;charset=UTF-8");  //获取get参数  $uname=$\_GET["uname"];  $upwd=$\_GET["pwd"];  echo "success...".'-->'.$uname.'-->'.$upwd;  ?> |

## 3、POST请求

post需要加入第三个参数**emulateJSON:true**

|  |
| --- |
| //提交请求 $http  var url="data2.php";  var datas={uname:'上海',pwd:'123456'};  var post={emulateJSON:true};  this.$http.post(url,datas,post).then(function(succ){  this.msg =succ.data;  }); |
| 服务器data2.php  <?php  //获取get参数  $uname=$\_POST["uname"];  //$upwd=$\_POST["pwd"];  //不建议使用，效率低  $upwd=$\_REQUEST["pwd"];  echo "success...".'-->'.$uname.'-->'.$upwd;  ?> |

## 4、JSONP

可以解决跨域问题

https://tcc.taobao.com/cc/json/mobile\_tel\_segment.htm?tel=15201840784&callback=vue

|  |
| --- |
| var url="https://tcc.taobao.com/cc/json/mobile\_tel\_segment.htm";  var datas={tel:this.tel};  this.$http.jsonp(url,datas).then(function(succ){  this.msg =succ.data.carrier;  },function(fail){  console.log(fail);  }); |

## 5、合并

可以将前三个参数，合三为一，根据需要填写。 加入

* <url:地址>
* method：get post jsonp 默认get
* data:后台需要的数据
* emulateJSON:post方式需要该参数为true
* jsonp:jsonp需要该参数填写 回调名称

|  |
| --- |
| this.$http({  url:url,  method:'jsonp',  data:datas  }).then(function (resovled) {  this.msg=resovled.data;  }, function (rejected) {  }); |

练习

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title>获取百度的数据</title>  <script src="../js/jquery-1.11.3.js"></script>  <script src="../js/vue.js"></script>  <script src="../js/vue-resource.js"></script>  <style>  .sel{  background:#ccc;  }  </style>  </head>  <body>  <input type="text" v-model="keywords" @keyup="queryData" @keydown.up.prevent="moveup"  @keydown.down="movedown"  />  <ul>  <li v-for="d in datas" :class="{sel:currentIdx==$index}">{{d}}</li>  </ul>  <span v-if="datas.length==0">没有获取数据</span>  </body>  </html>  <script>  $(function(){  new Vue({  el:'body',  data:{  keywords:'',  datas:[],  currentIdx:0  },  methods:{  queryData:function(ev){  //阻止上下移动 不发送请求  if(ev.keyCode==38 ||ev.keyCode==40){  return ;  }  //回城  if(ev.keyCode==13){  window.open("https://www.baidu.com/s?wd="+this.keywords);  this.keywords="";  this.currentIdx=0  }  //发送 jsonp的请求  var url ="https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su";  var data={wd:this.keywords};  this.$http({  url:url,  data:data,  method:'jsonp',  jsonp:'cb'  }).then(function(succ){  this.datas=succ.data.s;  });  },  moveup:function(){  this.currentIdx--;  if(this.currentIdx==-1){  this.currentIdx=this.datas.length-1;  }  this.keywords =this.datas[this.currentIdx]; //文本框内容  },  movedown:function(){  this.currentIdx++;  if(this.currentIdx==this.datas.length){  this.currentIdx=0;  }  this.keywords =this.datas[this.currentIdx]; //文本框内容  }  }  });  });  </script> |

# 八、数据库

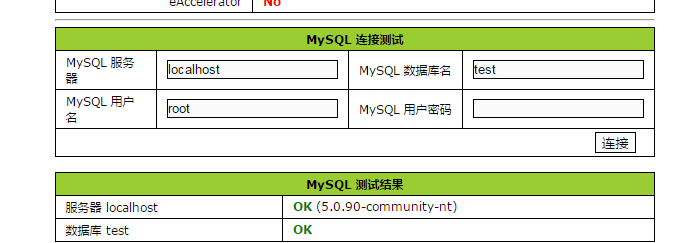
## 1、概念

关系型: 类似数组 mysql 非关系型:类似JSON对象

## 2、SQL语言

SQL: structed query language 结构化查询语言 ，关系型数据库都支持的语言。

数据库 + 表 (字段、类型、长度、记录 约束)



## 1)、创建数据库

|  |
| --- |
| --创建数据库  CREATE DATABASE sxt\_db DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci; |

## 2)、创建表

|  |
| --- |
| use sxt\_db;  create table tb\_comment(  id int primary key auto\_increment,  content varchar(200),  pubtime timestamp,  agree int default '0',  disagree int default '0'  ) ENGINE=MyISAM AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8; |

## 3)、操作数据

a)、添加数据 🡪评论添加

|  |
| --- |
| --添加  insert into tb\_comment(content,pubtime,agree,disagree) values('成功了吗',now(),9999,0);  insert into tb\_comment(content,pubtime,agree,disagree) values('vue学的如何',now(),888,0);  INSERT INTO tb\_comment(content,pubtime,agree,disagree) VALUES('坚持',NOW(),99799,0);  INSERT INTO tb\_comment(content,pubtime,agree,disagree) VALUES('test',FROM\_UNIXTIME(1497447921),0,0)  COMMIT; |

b)、删除数据 🡪评论删除

|  |
| --- |
| --删除  DELETE FROM tb\_comment WHERE id=1;  DELETE FROM tb\_comment WHERE id<10; |

c)、更新数据 🡪评论内容修改

|  |
| --- |
| --更新记录  UPDATE tb\_comment SET disagree =disagree+1 WHERE id=23; |

d)、查询数据 🡪获取评论

e)、查询部分数据 🡪获取某几条评论 (分页)

|  |
| --- |
| --查询记录  SELECT content,pubtime,disagree,id,agree FROM tb\_comment ORDER BY id DESC;  --分页5  SELECT content,pubtime,disagree,id,agree FROM tb\_comment ORDER BY id DESC LIMIT 0,5;  SELECT content,pubtime,disagree,id,agree FROM tb\_comment ORDER BY id DESC LIMIT 5,5;  SELECT content,pubtime,disagree,id,agree FROM tb\_comment ORDER BY id DESC LIMIT 10,5;  SELECT content,pubtime,disagree,id,agree FROM tb\_comment ORDER BY id DESC LIMIT 15,5;  SELECT content,pubtime,disagree,id,agree FROM tb\_comment ORDER BY id DESC LIMIT 20,5; |

f)、统计总记录数

|  |
| --- |
| --统计总记录数  Select count(\*) from tb\_ comment |

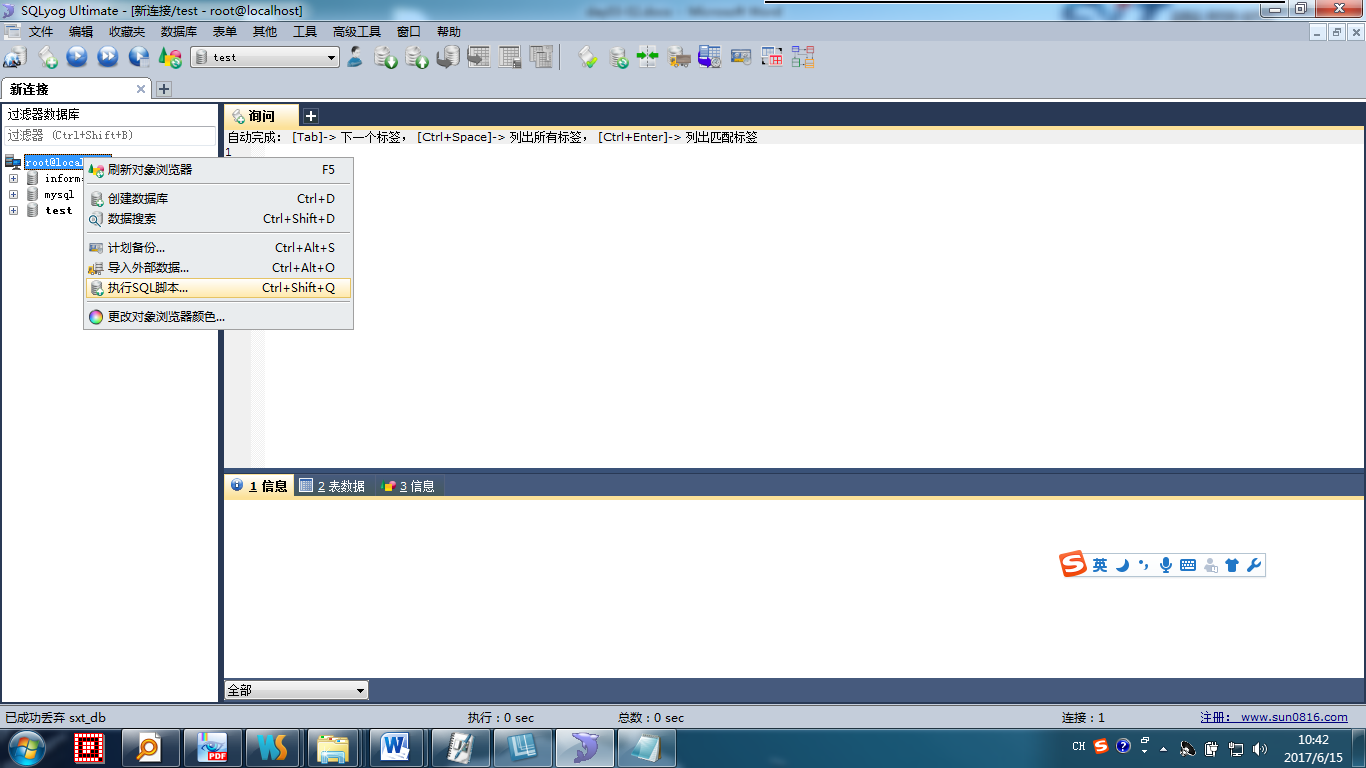
# 九、PHP连接数据库

## 1、准备工作

1)、安装好数据库

2)、使用SQLYog连接

3)、导入SQL文件



Comment.sql文件为:

|  |
| --- |
| /\*  SQLyog Ultimate v11.11 (64 bit)  MySQL - 5.0.90-community-nt : Database - sxt\_db  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*/  /\*!40101 SET NAMES utf8 \*/;  /\*!40101 SET SQL\_MODE=''\*/;  /\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;  /\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;  /\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;  /\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;  CREATE DATABASE /\*!32312 IF NOT EXISTS\*/`sxt\_db` /\*!40100 DEFAULT CHARACTER SET utf8 \*/;  USE `sxt\_db`;  /\*Table structure for table `tb\_comment` \*/  DROP TABLE IF EXISTS `tb\_comment`;  CREATE TABLE `tb\_comment` (  `id` int(11) NOT NULL auto\_increment,  `content` varchar(200) default NULL,  `pubtime` timestamp default NULL,  `agree` int(11) default '0',  `disagree` int(11) default '0',  PRIMARY KEY (`id`)  ) ENGINE=MyISAM AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;  /\*Data for the table `tb\_comment` \*/  insert into `tb\_comment`(`id`,`content`,`pubtime`,`agree`,`disagree`) values (1,'成功了吗','10:39:00',9999,0),(2,'vue学的如何','10:39:00',888,0),(3,'坚持','10:39:00',99799,0);  /\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;  /\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;  /\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;  /\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/; |

## 2、PHP操作数据库

### 1)、连接数据库

|  |
| --- |
| //字符集  header("Content-Type:text/html;charset=UTF-8");  //连接数据库  $link=mysql\_connect("localhost","root","shsxt");  if(!$link){  echo "没有连接成功!";  } else{  } |

### 2)、操作数据

|  |
| --- |
| //1)、选择数据库  mysql\_select\_db("sxt\_db", $link);  mysql\_query("set names 'utf8'",$link);  //获取对应的处理  $action=$\_GET['act'];  switch($action){  case 'add': //添加  case 'del': //删除  case 'agr': //点赞  case 'dis'://踩  case 'per': //获取每页记录  case 'page'://统计页数 break;  } |

### 3)、接口

目标url 传递参数 与获取结果

|  |  |  |  |
| --- | --- | --- | --- |
| **操作** | **url** | **参数** | **结果** |
| **添加** | http://localhost/php/005\_db/comment.php | act=add &conent=xxx&cb=xxx | 成功: xxx({"msg": 1, "id": 4, "time": 1497495559})  失败:没有对应的id |
| **点赞** | http://localhost/php/005\_db/comment.php | act=agr&id=xx&cb=xxx | 成功: xxx({"msg":1})  失败: xxx({"msg":0}) |
| **踩** | http://localhost/php/005\_db/comment.php | act=dis&id=xx&cb=xxx | 成功: xxx({"msg":1})  失败: xxx( {"msg":0}) |
| **删除** | http://localhost/php/005\_db/comment.php | act=del&id=xx&cb=xxx | 成功:{"msg":1}  失败: {"msg":0} |
| **分页** | http://localhost/php/005\_db/comment.php | act=per&page=2&cb=xxx | xxx([{"id":5,"content":"测试中文","pubtime":0,"agree":0,"disagree":0},{"id":4,"content":"abc","pubtime":0,"agree":2,"disagree":1},{"id":3,"content":"坚持","pubtime":0,"agree":99799,"disagree":0}]) |
| **页数** | http://localhost/php/005\_db/comment.php | act=page&cb=xxx | xxx({"page": 2}) |

# 十、访问接口BSD搭桥

## 1、确保服务能够访问

http://localhost/php/005\_db/comment.php?act=page

|  |
| --- |
| {"page": 2} |

## 2、编写静态页

|  |
| --- |
| <!doctype html>  <html>  <head>  <meta charset="utf-8">  <title>vue评论</title>  <link href="css/comment.css" rel="stylesheet" type="text/css" />  </head>  <body>  <div class="sxt">尚学堂真心不错哦</div>  <div class="leave">  <!--留言列表-->  <div class="viewLeave">  <div class="leaveNo">暂无留言</div>  <div class="leaveList">  <div class="leaveHis">  <p class="leaveContent">不错加油</p>  <p class="dealwith">  <span class="leaveTime">2016-12-12 12:12:12</span>  <span class="dohaha">  <a href="javascript:;" class="agree">100</a>  <a href="javascript:;" class="disagree">20</a>  <a href="javascript:;" class="remove">删除</a>  </span>  </p>  </div>  <div class="leaveHis">  <p class="leaveContent">不错加油</p>  <p class="dealwith">  <span class="leaveTime">2016-12-12 12:12:12</span>  <span class="dohaha">  <a href="javascript:;" class="agree">100</a>  <a href="javascript:;" class="disagree">20</a>  <a href="javascript:;" class="remove">删除</a>  </span>  </p>  </div>  </div>  <div class="pageList"><a href="javascript:;" class="active">5</a></div>  </div>  <!--留言发表-->  <div class="comment">  <textarea class="commentNow"></textarea>  <div class="commentSub">  <input type="button" class="sub" value="" />  </div>  </div>  </div>  </body>  </html> |

### 1)、获取第一页

|  |
| --- |
| queryComments:function(n){  //设置页码  this.now=n||1;  //获取第一页的数据  this.$http({  url:'http://localhost/php/005\_db/comment.php',  data:{act:'per',page:this.now},  method:'jsonp',  jsonp:'cb'  }).then(function(success){  this.comments=success.data;  });  } |
| created:function(){  //查询第一页  this.queryComments(1);  //获取页码  this.queryPages();  } |

### 2)、获取页码

|  |
| --- |
| queryPages:function(){  //获取页码  this.$http({  url:'http://localhost/php/005\_db/comment.php',  data:{act:'page'},  method:'jsonp',  jsonp:'cb'  }).then(function(success){  this.pages=success.data.page;  });  } |

### 3)、获取其他页面的数据

绑定事件

|  |
| --- |
| @click="queryComments(p+1)" |

### 4)、添加评论

|  |
| --- |
| <textarea class="commentNow" v-model="content"></textarea> |
| <div class="commentSub">  <input type="button" class="sub" @click="addComment"/>  </div> |
| addComment:function(){  //添加评论  this.$http({  url:'http://localhost/php/005\_db/comment.php',  data:{act:'add',content:this.content},  method:'jsonp',  jsonp:'cb'  }).then(function(success){  if(success.data.msg>0){ //成功  this.content ="";  this.queryComments(1);  }  });  } |

### 5)、点赞

|  |
| --- |
| agree:function(id){  //点赞  this.$http({  url:'http://localhost/php/005\_db/comment.php',  data:{act:'agr',id:id},  method:'jsonp',  jsonp:'cb'  }).then(function(success){  if(success.data.msg>0) { //成功  this.queryComments(this.now);  }  });  } |

### 6)、删除

删除数据 成功后

a)、更新页码

b)、最后一页的处理

加入了一个变量

|  |
| --- |
| remove:function(id){  //删除  this.$http({  url:'http://localhost/php/005\_db/comment.php',  data:{act:'del',id:id},  method:'jsonp',  jsonp:'cb'  }).then(function(success){  if(success.data.msg>0) { //成功  this.queryPages();  this.num--;  if(this.num==0){  this.now =this.now-1;  }  //查询当前页  this.queryComments(this.now);  }  });  } |

# 十一、VUE 实例

## 1、创建

### 1)、默认构造器

用vue必须先 通过构造函数Vue 创建一个 Vue 的根实例。

|  |
| --- |
| /vm是 MVVM 模式中所描述的 ViewModel 缩写  var vm = new Vue({  // 选项（必选，可以包含数据、模板、挂载元素、方法、生命周期钩子等选项。）  }); |

### 2)、组件构造器

组件构造器（所有的 Vue.js 组件其实都是被扩展的 Vue 实例）

|  |
| --- |
| var MyComponent = Vue.extend({  // 扩展选项  });  // 所有的 `MyComponent` 实例都将以预定义的扩展选项被创建  var instance = new MyComponent(); |

使用 :注意 所有的属性 必须 以函数的形式返回

|  |
| --- |
| //组件构造器 创建实例  var myComponent =Vue.extend({  /\*  错误的写法  data:{  }  \*/  data:function(){  return {  msg:"vue is so easy..."  }  }  });  $(function(){  var vm =new myComponent();  //挂载一定在实例上  vm.$mount("body"); //手动挂载  }); |
| <body>  {{msg}}  </body> |

## 2、属性与方法

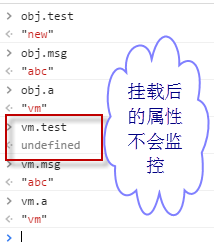
### 1)、代理属性

每个 Vue 实例都会代理其 data 对象里所有的属性（意思是：data内的属性就是vm的属性）

注意！只有这些被代理的属性是响应的。如果在实例创建之后添加新的属性，它不会触发视图更新。

|  |
| --- |
| <script>  var obj ={a:'vm',msg:"abc"};  var Component=Vue.extend({  data:function(){  return obj;  }  });  var vm ;  $(function(){  //创建实例  vm =new Component();  //挂载  vm.$mount('body');  obj["test"] ="new"; //新增属性  });  </script> |

此时: 数据对象 被 vm 监控 obj与vm都可以操作 (已有的属性) ，新增属性不会监控



### 2)、实例属性

* vm.$el: 元素
* vm.$data: data
* vm.$mount： 手动挂在vue程序
* vm.$options 获取自定义属性
* vm.$destroy 销毁对象
* vm.$log() 查看现在数据的状态
* vm.$watch() 监听数据变化

|  |
| --- |
| <script>  //组件构造器  var Component=Vue.extend({  data:function(){  return {  msg:'component构造器'  };  }  });  var vm =new Component();  $(function(){  //手动挂载vue程序  vm.$mount('#test');  //任意操作 dom $el  vm.$el.style.background="#ccc";  //$data  console.log(vm.$data.msg);  });  </script>  </head>  <body>  <div id="test">  {{msg}}  </div>  </body> |

延迟挂载

|  |
| --- |
| <script>  //延迟挂载  var Component=Vue.extend({  data:function(){  return {  msg:'component构造器'  };  }  });  var vm =new Component();  function lazyMount(){  console.log("aaaaa");  vm.$mount('#test');  }  </script>  </head>  <body>  <div id="test">  {{msg}}  </div>  <button onclick="lazyMount()">延迟挂载</button>  </body> |

**$options**

|  |
| --- |
| <script>  $(function(){  new Vue({  el:'body',  data:{msg:'options'},  self:'自定义属性1',  methods:{  test:function(){  console.log(this.$options.self);  }  }  });  })  </script>  </head>  <body>  <div id="test">  {{msg}}  <button @click="test"> 查看其它的新追加的属性</button>  </div>  </body> |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## 3、生命周期

vue中提供了回调函数(**钩子函数**)，说明了生存周期，这里介绍常用的(1.0版本，2.0以后有了变化)：**常用ready….**

|  |  |
| --- | --- |
| **状态** | **说明** |
| **init** | 组件实例刚被创建，属性还未绑定 |
| **created** | 实例已经创建，属性已绑定，但DOM未生成  this便指向vue实例,**这点很重要** |
| **beforeCompile** | 模板被编译之前 |
| **compiled** | 模板被编译完成之后 |
| **ready** | 模板被插入到文档中 |
| **beforeDestroy** | 销毁之前 |
| **destroyed** | 销毁之后 |

说明：

* vue的created事件以后,this便指向vue实例,这点很重要。
* 编译，是将HTML中的模板在内存中编译成真正的数据。
* 插入到文档中，才会在页面中显示数据。
* 实例销毁之后，插入到DOM中的内容并不会消失，只是这个实例销毁以后不能再使用了。

|  |
| --- |
|  |

# 十二（一）、组件

## 1、全局组件

### 第一种: 先定义后注册 再使用

#### 1)、定义组件

Vue.extend({

template:’’

});

#### 2)、注册组件

Vue.component(“标签名”,组件名);

#### 3)、使用组件

与之前html标签 一致

|  |
| --- |
| <body>  <hello></hello>  </body>  <script>  //定义组件  var Hello =Vue.extend({  template:'<h1>hello vue</h1>'  });  //注册组件  Vue.component("hello",Hello);  //使用组件 必须在 范围内使用  //入口点 在 注册组件之后  new Vue({  el:'body'  });  </script>  </html> |

### 第二种: 匿名(定义+注册) 再使用

|  |
| --- |
| <body>  <hello></hello>  </body>  <script>  //匿名组件  Vue.component("hello",{  template:'<b>第二种方式</b>'  });  //使用组件 必须在 范围内使用  //入口点 在 注册组件之后  new Vue({  el:'body'  });  </script> |

### 数据与方法

#### 1、数据

必须为函数的形式 与 返回json对象

|  |
| --- |
| <body>  <hello></hello>  </body>  <script>  //组件的数据 必须 在 组件的模板中使用 +函数的形式  Vue.component("hello",{  template:'<b>第二种方式</b>{{msg}}',  data:function(){  return {  msg:'组件中使用的数据'  };  }  });  //使用组件 必须在 范围内使用  //入口点 在 注册组件之后  new Vue({  el:'body'  });  </script> |

#### 2、方法

方法 编写与 之前一致，使用 必须在组件中使用

|  |
| --- |
| <body>  <hello></hello>  </body>  <script>  //组件的数据 必须 在 组件的模板中使用 +函数的形式  Vue.component("hello",{  template:'<b>第二种方式</b>{{msg}} <button @click="show">点击改变</button>',  data:function(){  return {  msg:'组件中使用的数据'  };  },  methods:{  show:function(){  this.msg="abc";  }  }  });  //使用组件 必须在 范围内使用  //入口点 在 注册组件之后  new Vue({  el:'body'  });  </script> |

## 2、局部组件

包含在 实例 或组件中的组件 +注册的位置改变了。。。

可以先定义后注册+也可以匿名

|  |
| --- |
| <body>  <sxt></sxt>  </body>  <script>  //定义组件  var Sxt=Vue.extend({  template:'<h1>定义组件</h1> <inner></inner>',  components:{  "inner":{  template:'<h2>内部组件2</h2>'  }  }  });  //Vue.component("sxt",Sxt);  //入口点 在 注册组件之后  new Vue({  el:'body',  components:{  "sxt":Sxt  }  });  </script> |
|  |

# （二）组件

## 1、模板

### 1)、创建三种方式

内置 template <script type=”x-template” id=””> <template id=”xxx”>

|  |
| --- |
| <body>  <sxt></sxt>  </body>  <script type="x-template" id="subtemp">  <h5>使用script标签制作的模板</h5>  <ul>  <li><child></child></li>  <li>testsub</li>  <li>testsub</li>  </ul>  </script>  <template id="childtemp">  <hr/>  <h1>使用template标签制作的模板</h1>  </template>  <script>  var SXT =Vue.extend({  template:`<b>内置在组件体中</b><br/><sub></sub><sub></sub><sub></sub>`,  data:function(){  return {  msg:'模板'  }  },  components:{  'sub':{  template:'#subtemp',  components:{  'child':{  template:'#childtemp'  }  }  }  }  });  Vue.component("sxt",SXT);  new Vue({  el:'body'  });  </script>  </html> |

### 2)、slot

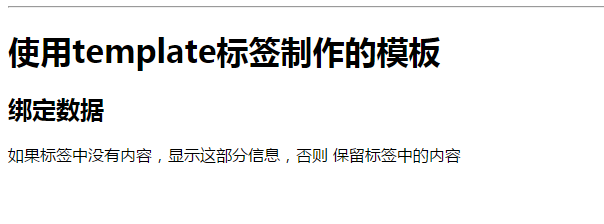
默认情况下，标签的内容会被模板的值覆盖。 slot:占位。

a)、作用: 保留标签中的内容

b)、注意:

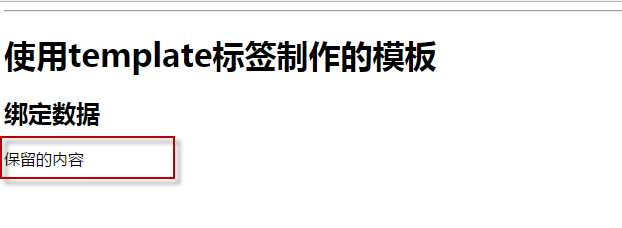
* 如果标签中没有内容，显示 slot里面的 文本

|  |
| --- |
| <body>  <sxt></sxt>  </body>  <template id="temp">  <hr/>  <h1>使用template标签制作的模板</h1>  <h2>{{msg}}</h2>  <slot>如果标签中没有内容，显示这部分信息，否则 保留标签中的内容</slot>  </template> |



* 如果标签中存在内容， 保留标签中的内容

|  |
| --- |
| <body>  <sxt>保留的内容</sxt>  </body>  <template id="temp">  <hr/>  <h1>使用template标签制作的模板</h1>  <h2>{{msg}}</h2>  <slot>如果标签中没有内容，显示这部分信息，否则 保留标签中的内容</slot>  </template> |



* slot 可以存在多个 ，默认情况下一个slot 表示整个标签的内容

|  |
| --- |
| <body>  <sxt><b>保留的内容1</b><b>保留的内容2</b></sxt>  </body>  <template id="temp">  <slot>1</slot>  <hr/>  <slot>2</slot>  <h1>使用template标签制作的模板</h1>  <h2>{{msg}}</h2>  <slot>3</slot>  </template> |



* 保留内容 分离 。标签中使用slot=”xx” ，模板中 slot name=xxxS

|  |
| --- |
| <body>  <sxt><b slot="c1">保留的内容1</b><b slot="c2">保留的内容2</b></sxt>  </body>  <template id="temp">  <slot name="c1">1</slot>  <hr/>  <slot name="c2">2</slot>  <h1>使用template标签制作的模板</h1>  <h2>{{msg}}</h2>  <slot>3</slot>  </template> |



## 2、动态组件

默认使用的是标签名的形式使用组件， 动态切换组件 ， is:组件

|  |
| --- |
| <!--  <sh></sh>  <bj></bj>  -->  <component :is="'bj'" ></component>  <component :is="name" ></component> |
| <template id="sh">  <h1>上海东方明珠</h1>  </template>  <template id="bj">  <h1>北京尚学堂</h1>  </template>  <script>  new Vue({  el:'body',  components:{  'bj':{  template:'#bj'  },  'sh':{  template:'#sh'  }  },  data:{  comname:'bj'  }  });  </script> |

注意: 组件名 为字符串 加入’’ , 建议使用属性 来包裹。

动态切换组件的实例:

|  |
| --- |
| <body>  <!--  <sh></sh>  <bj></bj>  -->  <component :is="'bj'" ></component>  <component :is="comname" ></component>  <input type="text" v-model="comname"/>  </body>  <template id="sh">  <h1>上海东方明珠</h1>  </template>  <template id="bj">  <h1>北京尚学堂</h1>  </template>  <script>  new Vue({  el:'body',  components:{  'bj':{  template:'#bj'  },  'sh':{  template:'#sh'  }  },  data:{  comname:'bj'  }  });  </script>  </html> |

## 3、父子组件数据传递

* $root: 根组件 🡪入口点
* $parent 父组件
* $children[索引] :子组件
* $refs: 子组件别名
  + 定义别名: v-ref:别名
  + 使用别名:$refs.别名

|  |
| --- |
| <body>  <sh v-ref:oop></sh>  <sh2 v-ref:oop2></sh2>  {{$refs.oop.c}} {{$root.name}} {{$root.$children[0].$parent.type}}  </body>  <template id="sh">  {{$parent.type}} {{$parent==$root}}  </template>  <script>  new Vue({  el:'body',  data:{  name:'parent.',  type:'vue'  },  components:{  'sh':{  template:'#sh',  data:function(){  return {  c:'child'  }  }  }  }  });  </script> |

## 4、获取父组件数据

1)、在子组件中加入属性props即可，通过两种方式定义props:

* + 数组方式:props:['属性1', ‘属性2’]
  + json方式: props:{

'属性1':Js类型,

'属性2':js类型

}

注意: 驼峰命名 如 sxtName 绑定时 v-bind:sxt-Name

在子组件中定义接受变量 props

|  |
| --- |
| //props:['n','t']  props:{  'sxtName':String,  't':String  } |

在父模板中 使用时 绑定v-bind 父变量

|  |
| --- |
| <sh v-bind:t="type" :sxt-Name="name"></sh> |

在子模板中使用

|  |
| --- |
| <template id="sh">  <h1>获取父组件的数据</h1>  {{sxtName}} {{t}}  </template> |

数据绑定的问题(**了解**)

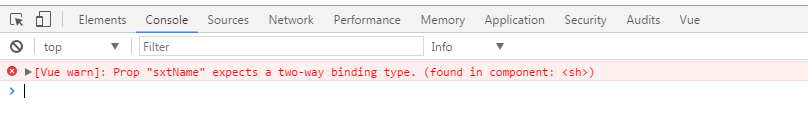
默认情况下，父亲的数据改变 ，prop跟着改变 ，反之 props改变，父亲的数据不改变。需要双向绑定 加入.sync

|  |
| --- |
| <sh v-bind:t="type" :sxt-Name.sync="name"></sh><br/> |

在编写props是，加入一个验证，强制提醒需要双向绑定。如果没有.sync编译错误。

|  |
| --- |
| props:{  'sxtName':{  twoWay: true  },  't':String  } |
| <sh v-bind:t="type" :sxt-Name.sync="name"></sh><br/> |

如果没有.sync 编译错误



## 5、获取子组件的数据

在子组件中通过事件形式，通过methods中定义的方法来发射数据，在此方法中编写:

格式:vm.$emit(事件名称,数据);

* 事件名称：自定义的名称
* 数据：要传递给父组件的数据

1)、通过事件 发射出 子组件的数据

|  |
| --- |
| methods:{  shooting:function(){  console.log("aaaaaaa");  this.$emit("child-data1",this.name);  this.$emit("child-data2",this.type);  }  },  ready:function(){  this.shooting();  } |

2)、父组件中 准备方法 接受数据

|  |
| --- |
| data:{  p1:"",  p2:""  },  methods:{  getData1:function(data){  this.p1=data;  },  getData2:function(data){  this.p2=data;  }  }, |

3)、父模板中关联事件 v-on:

|  |
| --- |
| <sh @child-data1="getData1" @child-data2="getData2"></sh> |

## 6、作用域

组件和组件之间，即使有同名属性，值也不共享

|  |
| --- |
| <div id="app">  <h3>组件和组件之间，即使有同名属性，值也不共享</h3><br/>  <add></add>  <del></del>  </div>  </body>  <script>  var vm = new Vue({  el: '#app',  components: {  "add": {  template: "<button>btn：{{btn}}</button>",  data: function () {  return {btn: "123"};  }  },  del: {  template: "<button>btn：{{btn}}</button>",  data: function () {  return {btn: "456"};  }  }  }  });  </script> |

# 十三、路由

路由即根据不同url地址，分发出现不同效果，用来制作SPA应用即单页面应用。这种实现，简称单页面应用（A single-page application (SPA)），SPA是仅包含单个网页的应用，目的是为了提供类似于本地应用的流畅用户体验。

1、hash 2、h5 history

## 1、原理

|  |
| --- |
| <script>  //http://localhost:63342/class/013\_route/001\_start.html#abc  //必须带一个锚点才有hash值  function show(){  console.log("--->"+window.location.hash+"-->" +window.location.hash.slice(1));  }  </script>  <title></title>  </head>  <body>  <input type="button" value="t2" onclick="show()"/>  </body> |

使用hash值来跳转….

|  |
| --- |
| <script>  function Router() {  this.routes = {};  this.currentUrl = '';  }  //关联 路径 与对应的函数  Router.prototype.route = function(path, callback) {  this.routes[path] = callback || function(){};  };  //刷新 ：调用 回调函数  Router.prototype.refresh = function() {  this.currentUrl = location.hash.slice(1) || '/';  this.routes[this.currentUrl]();  };  //绑定事件  Router.prototype.init = function() {  //load事件在当前页面加载时触发  window.addEventListener('load', this.refresh.bind(this), false);  //hashchange事件在当前页面URL中的hash值发生改变时触发  window.addEventListener('hashchange', this.refresh.bind(this), false);  }  window.Router = new Router();  window.Router.init();  </script> |
| <body>  <ul>  <li><a href="#/">切换1</a></li>  <li><a href="#/gray">切换2</a></li>  <li><a href="#/green">切换3</a></li>  </ul>  <script>  var body = document.querySelector('body');  //切换hash的事件  function changeBgColor(color) {  body.style.backgroundColor = color;  }  //切换  Router.route('/', function() {  changeBgColor('#fff');  });  Router.route('/gray', function() {  changeBgColor('#f0f0f0');  });  Router.route('/green', function() {  changeBgColor('green');  });  </script>  </body> |

## 2、路由

### 1)、新增路由支持

使用bower [下载vue-router@0.7.13](mailto:下载vue-router@0.7.13)

#### 新建 .bowerrc

|  |
| --- |
| {  "directory" : "js/lib"  } |

#### 新建bower.json

|  |
| --- |
| {  "name": "sxt",  "description": "bower is so easy",  "main": "main.js",  "keywords": [  "sxt"  ],  "authors": [  "laopei"  ],  "license": "MIT",  "homepage": "index.html",  "private": true,  "ignore": [  "\*\*/.\*",  "node\_modules",  "bower\_components",  "js/lib",  "test",  "tests"  ],  "dependencies": {  "vue-router": "0.7.13"  }  } |

#### 安装

|  |
| --- |
| bower install vue-router@0.7.13 |

### 2)、编写路由

#### a)、跳转与渲染

跳转: v-link

|  |
| --- |
| <div>  <ul>  <li><a v-link="{path:'/home'}">主页</a></li>  <li><a v-link="{path:'/about'}">关于</a></li>  </ul>  </div> |

渲染: 显示dom 的地方

|  |
| --- |
| <router-view></router-view> |

#### b)、编写路由的步骤

* 创建组件
  + 根组件
  + 子组件
* 创建路由实例
* 配置映射
* 启动路由

##### I)、创建组件

|  |
| --- |
| //1、创建根组件  var App =Vue.extend({});  //2、创建2个子组件  var Home=Vue.extend({  template:'#home'  });  var About=Vue.extend({  template:'#about'  }); |

##### II)、创建路由实例

|  |
| --- |
| //创建路由实例  var myRouter=new VueRouter(); |

##### III)、配置映射

配置 路由的路径与 组件的关系

|  |
| --- |
| myRouter.map({  '/home':{  component:Home  },  '/about':{  component:About  }  }); |

##### V)、启动路由

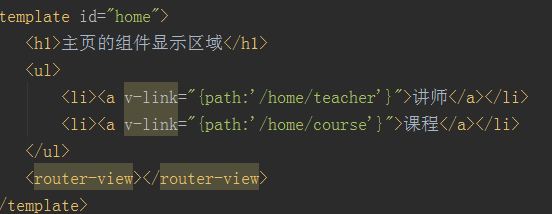
|  |
| --- |
| myRouter.start(App, '#box'); |

##### VI)、设置主页

|  |
| --- |
| //配置主页  myRouter.redirect({  '/': '/home'  }); |

### 3)、路由的嵌套

子路由对应的 链接与渲染的位置 在 父路由的模板中



在映射时存在

|  |
| --- |
| subRoutes:{  } |

完整的代码

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title>路由</title>  <script src="../js/jquery-1.11.3.js"></script>  <script src="../js/vue.js"></script>  <script src="../js/vue-router.min.js"></script>  </head>  <body>  <div id="box">  <ul>  <li><a v-link="{path:'/home'}">主页</a></li>  <li><a v-link="{path:'/about'}">关于</a></li>  </ul>  <router-view></router-view>  </div>  </body>  <template id="home">  <h1>主页的组件显示区域</h1>  <ul>  <li><a v-link="{path:'/home/teacher'}">讲师</a></li>  <li><a v-link="{path:'/home/course'}">课程</a></li>  </ul>  <router-view></router-view>  </template>  <template id="course">  <ul>  <li>前端</li>  <li>JAVA</li>  <li>大数据</li>  </ul>  </template>  <template id="teacher">  <ul>  <li>老相</li>  <li>你懂得</li>  </ul>  </template>  <template id="about">  <h1>关于我们</h1>  </template>  <script>  //1、创建根组件  var App =Vue.extend({});  //2、创建2个子组件  var Home=Vue.extend({  template:'#home'  });  var About=Vue.extend({  template:'#about'  });  var Teacher=Vue.extend({  template:'#teacher'  });  var Course=Vue.extend({  template:'#course'  });  //创建路由实例  var myRouter=new VueRouter();  //配置路由的路径  myRouter.map({  '/home':{  component:Home,  subRoutes:{  '/teacher':{  component:Teacher  },  '/course':{  component:Course  }  }  },  '/about':{  component:About  }  });  //启动路由 -->不需要创建vue实例  myRouter.start(App, '#box');  //配置主页  myRouter.redirect({  '/': '/home'  });  </script>  </html> |

## 3、具名路径

如访问/ home/courses/detail/01 和/ home/courses/02 存在两种方式

* 相对路径:name + params:{id:01}
* 绝对路径/ home/courses/detail/01

|  |
| --- |
| <ul>  <li><a v-link="{path:'/home/teacher/info/001'}">老相</a></li>  <li><a v-link="{name: 'info', params: {teacherno: '002'} }">你懂得</a></li>  </ul> |

name 必须搭配,map配置



路径与路由的参数

|  |
| --- |
| subRoutes: {  'info/:teacherno/:pwd': {  name: 'info',  component: Info  }  } |
| <ul>  <li><a v-link="{path:'/home/teacher/info/001/aaa'}">老相</a></li>  <li><a v-link="{name: 'info', params: {teacherno: '002',pwd:'sss'} }">你懂得</a></li>  </ul> |

## 4、选中样式

选中的路由默认样式的为v-link-active，可以设置activeClass: 'active'

|  |
| --- |
| <style>  .active{  font-size: 20px;  color: blue;  }  </style> |
| <ul>  <li><a v-link="{path:'/home',activeClass: 'active'}">主页</a></li>  <li><a v-link="{path:'/about',activeClass: 'active'}">关于</a></li>  </ul> |

## 5、其他属性

|  |  |
| --- | --- |
| **属性对象** | **说明** |
| **$route.path** | 字符串，等于当前路由对象的路径，会被解析为绝对路径 |
| **$route.params** | 对象，包含路由中的动态片段和全匹配片段的键值对 |
| **$route.query** | 对象，包含路由中查询参数的键值对 |
| **$route.router** | 路由规则所属的路由器（以及其所属的组件）。 |
| **$route.matched** | 数组，包含当前匹配的路径中所包含的所有片段所对应的配置参数对象。 |
| **$route.name** | 当前路径的名字，如果没有使用具名路径，则名字为空。 |

# 十四、v-loader加载器

现在开始做vue最奇妙的部分，使用vue-loader构建组件.vue。

vue-loader是基于webpack下loader(加载器)插件可以把.vue文件输出成组件，即翻译过程。(建议使用这种模式开发。。。)

## 1、loader加载器

模块加载器就是翻译过程，将nodjs编写的代码|模块化进行翻译工具，如

* + 前端: css-loader、url-loader、html-loader browserify.....

require(“xxx.css”)

* + 后端: webpack vue-loader

## 2、.vue文件

vue文件，官方推荐命名法，首字母大写(Hello.vue)，.vue 放置vue组件的代码，有三块组成:

* + <template>html代码</template>
  + <style>css代码</style>
  + <script>es6代码<script>

**.vue称之为组件**

## 3、简单目录结构

|-index.html 根文件

|-main.js 入口文件

|-App.vue vue文件，官方推荐命名法，首字母大写。

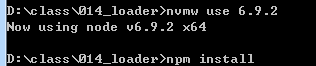
|-package.json 团队开发的工程文件(项目依赖、名称、配置)

|-webpack.config.js webpack配置文件

## 4、配置npm的文件 package.json

|  |
| --- |
| {  "name": "005\_loader",  "version": "1.0.0",  "description": "",  "main": "main.js",  "scripts": {  "dev": "webpack-dev-server --inline --hot --port 8082",  "build":"webpack -p"  },  "keywords": [],  "author": "",  "license": "ISC",  "devDependencies": {  "babel-core": "^6.21.0",  "babel-loader": "^6.2.10",  "babel-plugin-transform-runtime": "^6.15.0",  "babel-preset-es2015": "^6.18.0",  "babel-runtime": "^6.20.0",  "css-loader": "^0.26.1",  "style-loader": "^0.13.1",  "vue-hot-reload-api": "^1.3.2",  "vue-html-loader": "^1.2.3",  "vue-loader": "^7.5.3",  "webpack": "^1.14.0",  "webpack-dev-server": "^1.16.2"  },  "dependencies": {  "vue": "^1.0.28"  }  } |

安装



## 5、WebPack的配置文件 webpack.config.js

|  |
| --- |
| module.exports={  //入口文件  entry:'./main.js',  //出口文件  output:{  path:\_\_dirname,//当前目录  filename:'result.js'  },  module:{  loaders:[//所有的loader，都配置到这里  //vue-loader 的-loader不写，多个loader 以!连接  {test:/\.vue$/,loader:'vue'},  {test:/\.js$/,loader:'babel',exclude:/node\_modules/}  ]  },  babel:{  //es2015  presets:['es2015'],  //还需要一个插件:实时编译  plugins:['transform-runtime']  }  } |

## 6、App.vue

|  |
| --- |
| <template>  vue-loader is powerfull  </template>  <style></style>  <script><script> |

## 7、index.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title>vue使用</title>  </head>  <body>  <App></App>  <script src="result.js"></script>  </body>  </html> |

## 8、main.js

|  |
| --- |
| import Vue from 'vue'  import App from './App.vue'  new Vue({  el:'body',  components:{  'app':App  }  }); |

## 9、运行

npm run dev

## 10、复杂的Vue

|  |
| --- |
| <template>  <h3 @click="haha();">vue-loader is powerfull</h3>  {{msg}}  </template>  <style>  body{  background:'#ccc'  }  </style>  <script>  //导出模块 类似于component组件  export default{  data(){  return {  msg:'success....'  }  },methods:{  haha(){  this.msg ="----end..."  }  }  }  </script> |

# 十五、v-router

## 1、下载

npm install vue-router@0.7.13 -sav-dev

## 2、编写两个组件

components /Hello.vue

|  |
| --- |
| <template><h1>上海尚学堂</h1></template> |

components /About.vue

|  |
| --- |
| <template><h1>www.shsxt.com</h1></template> |

## 3、main.js 导入路由

|  |
| --- |
| import Vue from 'vue'  import App from './App.vue'  import VueRouter from 'vue-router'  import Home from './components/Home.vue'  import About from './components/About.vue'  //使用router  Vue.use(VueRouter); |

## 4、配置路由

|  |
| --- |
| //创建Router  let router =new VueRouter();  //配置  router.map({  'home':{  component:Home  },  'about':{  component:About  }  });  //默认主页  router.redirect({  "/":'home'  });  //开启  router.start(App,"#sxt"); |

## 5、修改 index.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title>vue使用</title>  </head>  <body>  <div id="sxt">  </div>  <script src="result.js"></script>  </body>  </html> |

## 6、修改App.vue

|  |
| --- |
| ﻿<template>  <div id="sxt">  <ul>  <li><a v-link="{path:'/home'}">主页</a></li>  <li><a v-link="{path:'/about'}">关于</a></li>  </ul>  <router-view></router-view>  </div>  </template>  <style>  body{  background:'#ccc'  }  </style> |

## 7、启动

|  |
| --- |
| npm run dev |

## 8、加入一个选中样式

|  |
| --- |
| .active{  font-size: 20px;  color: blue;  } |

App.vue 加入

|  |
| --- |
| <ul>  <li><a v-link="{path:'/home',activeClass: 'active'}">主页</a></li>  <li><a v-link="{path:'/about',activeClass: 'active'}">关于</a></li>  </ul> |

## 9、加入配置文件

route.config.js

|  |
| --- |
| //配置路由规则  import Home from './components/Home.vue'  import About from './components/About.vue'  //配置  export default {  'home':{  component:Home  },  'about':{  component:About  }  } |

## 10、修改main.js

|  |
| --- |
| import config from './route.config.js' |
| //读取配置  router.map(config); |

完整代码为:

|  |
| --- |
| import Vue from 'vue'  import App from './App.vue'  import VueRouter from 'vue-router'  import config from './route.config.js'  //使用router  Vue.use(VueRouter);  //创建Router  let router =new VueRouter();  //读取配置  router.map(config);  //默认主页  router.redirect({  "/":'home'  });  //开启  router.start(App,"#sxt"); |

## 11、启动

|  |
| --- |
| npm run dev |

## 12、子组件

只关注 如何 编写 子组件 及 配置 路由即可

### 1)、加入两个子路由

加入Course.vue 和Teacher.vue

|  |
| --- |
| <template>  <h1>课程列表</h1>  </template>  <script></script> |
| <template>  <h1>上海松江</h1>  </template>  <script></script> |

### 2)、配置路由

|  |
| --- |
| //配置路由规则  import Home from './components/Home.vue'  import About from './components/About.vue'  import Course from './components/Course.vue'  import Teacher from './components/Teacher.vue'  //配置  export default {  'home':{  component:Home,  subRoutes:{  'teacher':{  component:Teacher  },  'course':{  component:Course  }  }  },  'about':{  component:About  }  } |

### 3)、使用子组件

|  |
| --- |
| <template>  <h1>上海尚学堂-->very good</h1>  <ul>  <li><a v-link="{path:'/home/teacher',activeClass: 'active'}">教师团队</a></li>  <li><a v-link="{path:'/home/course',activeClass: 'active'}">金牌课程</a></li>  </ul>  <div id="box"><router-view></router-view></div>  </template>  <script></script>  <style>  .box{  border:1px dashed #000;  }  </style> |

### 4)、传入参数

#### 1)、Teacher.vue中加入详情

|  |
| --- |
| <template>  <h1>金牌讲师...</h1>  <ul>  <li><a v-link="{path:'/home/teacher/detail/001',activeClass: 'active'}">老裴</a></li>  <li><a v-link="{path:'/home/course/detail/002',activeClass: 'active'}">...</a></li>  </ul>  <router-view></router-view>  </template>  <script></script> |

#### 2)、定义Detail.vue

|  |
| --- |
| <template>  <h1>尚学堂老师 -->{{$route.params.id}}</h1> </template> <script></script> |

#### 3)、编写路由规则

|  |
| --- |
| //配置路由规则  import Home from './components/Home.vue'  import About from './components/About.vue'  import Course from './components/Course.vue'  import Teacher from './components/Teacher.vue'  import Detail from './components/Detail.vue'  //配置  export default {  'home':{  component:Home,  subRoutes:{  'teacher':{  component:Teacher,  subRoutes:{  'detail/:id':{  component:Detail  }  }  },  'course':{  component:Course  }  }  },  'about':{  component:About  }  } |

# 十六、脚手架vue-cli

**使用脚手架不用进行配置。。。。**

vue-cli 是 vue.js 的脚手架(即项目模板)，用于自动生成 vue.js 模板工程的,在公司玩的时

候不会自己配置，都是会用脚手架，即提供好了项目的一些基本结构，避免自己配置麻烦，

容易错误。常用的几个脚手架

*  simple
*  webpack 严格的代码检查(恶心 分号 空格 缩进)
*  webpack-simple 推荐
*  browserify
*  browserify-simple

## 1、安装

npm install vue-cli –g

## 2、下载simple版本

vue init simple#1.0 simple-demo

## 3、下载webpack-simple

vue init webpack-simple#1.0 webpack-simple-demo

a)、进入 webpack-simple-demo

cd webpack- simple -demo

b)、安装

npm install

漫长的等待后…. 安装好几次才成功。。。

c)、运行

npm run dv

如果端口冲突 ，需要更改 package.json 中加入 --port 8086

d)、结果

http://localhost:8086/

e)、尝试修改一下里面的内容 App.vue

|  |
| --- |
| <template>  <div id="app">  <h1>{{ msg }}</h1>  <button @click="reverse">反转</button>  </div>  </template>    <script>  export default {  data () {  return {  msg: 'Hello Vue!'  }  },methods:{  reverse(){  var str =this.msg;  str =str.split('').reverse();  this.msg =str;  }  }  }  </script>    <style>  body {  font-family: Helvetica, sans-serif;  }  </style> |