



北京动力节点教育科技有限公司

动力节点课程讲义

DONGLIJIEDIANKECHENGJIANGYI

www.bjpowernode.com

第1章 AJAX

1.1 全局刷新和局部刷新

B/S 结构项目中，浏览器（Browse）负责把用户的请求和参数通过网络发送给服务器（Server），服务端使用 Servlet（多种服务端技术的一种）接收请求，并将处理结果返回给浏览器。

浏览器在 html，jsp 上呈现数据，混合使用 css，js 帮助美化页面，或响应事件。

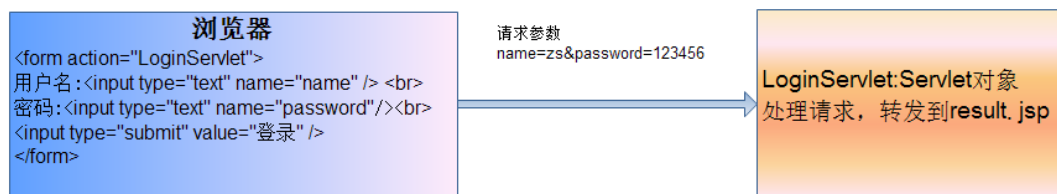
1.1.1 全局刷新

登录请求处理：

index.jsp 发起登录请求-----LoginServlet-----result.jsp

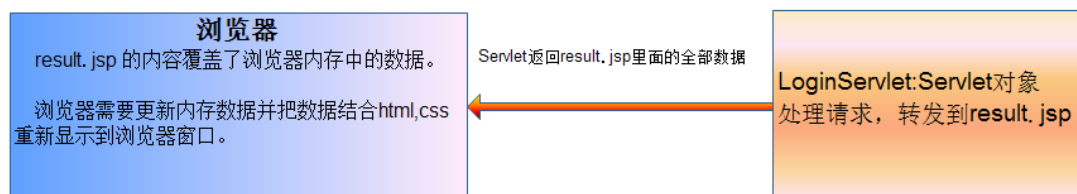
发起请求 request 阶段：

浏览器现在内存中是 index 页面的内容和数据：



服务器端应答结果阶段：

sevlet 返回后把数据全部覆盖掉原来 index 页面内容，result.jsp 覆盖了全部的浏览器内存数据。整个浏览器数据全部被刷新。重新在浏览器窗口显示数据，样式，标签等



全局刷新原理：

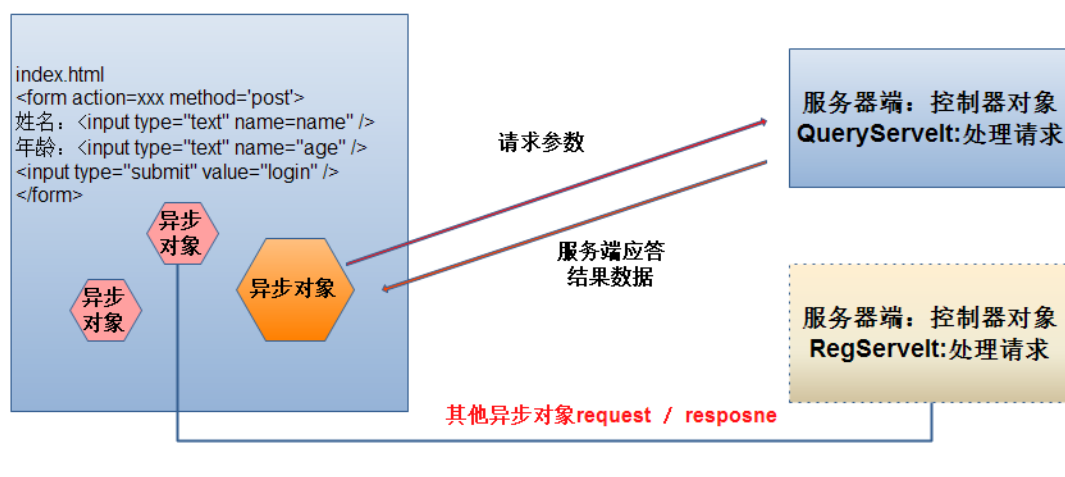
- 1) 必须由浏览器亲自向服务端发送请求协议包。
- 2) 这个行为导致服务端直接将【响应包】发送到浏览器内存中
- 3) 这个行为导致浏览器内存中原有内容被覆盖掉
- 4) 这个行为导致浏览器在展示数据时候，只有响应数据可以展示

1.1.2 局部刷新

浏览器在展示数据时，此时在窗口既可以看到本次的响应数据， 同时又可以看到浏览器内存中原有数据

局部刷新原理:

- 1) 不能由浏览器发送请求给服务端
- 2) 浏览器委托浏览器内存中一个脚本对象代替浏览器发送请求.
- 3) 这个行为导致服务端直接将【响应包】发送脚本对象内存中
- 4) 这个行为导致脚本对象内容被覆盖掉，但是此时浏览器内存中绝大部分内容没有收到任何影响.
- 5) 这个行为导致浏览器在展示数据时候,同时展示原有数据和响应数据



AJAX 实现局部刷新的一种技术。

1.2 异步请求对象:

在局部刷新，需要创建一个对象，代替浏览器发起请求的行为，这个对象存在内存中。代替浏览器发起请求并接收响应数据。这个对象叫做异步请求对象。

全局刷新是同步行为， 局部刷新是异步行为[浏览器数据没有全部更新]

这个异步对象用于在后台与服务器交换数据。XMLHttpRequest 就是我们说的异步对象。

XMLHttpRequest 对象能够:

- 在不重新加载页面的情况下更新网页

- 在页面已加载后向服务器请求数据
- 在页面已加载后从服务器接收数据

所有现代浏览器 (IE7+、Firefox、Chrome、Safari 以及 Opera) 都内建了 XMLHttpRequest 对象。通过一行简单的 JavaScript 代码，我们就可以创建 XMLHttpRequest 对象

创建 XMLHttpRequest 对象的语法 (xhr):

```
var xmlhttp=new XMLHttpRequest();
```

AJAX 中的核心对象就是 XMLHttpRequest

1.3 AJAX

1.3.1 什么是 AJAX

AJAX = Asynchronous JavaScript and XML (异步的 JavaScript 和 XML)。

AJAX 是一种在无需重新加载整个网页的情况下，能够更新部分页面内容的新方法

AJAX 不是新的编程语言，而是使用现有技术混合使用的一种新方法。ajax 中使用的技术有 JavaScript, html, dom, xml, css 等。主要是 JavaScript, XML.

JavaScript: 使用脚本对象 XMLHttpRequest 发送请求，接收响应数据

XML: 发送和接收的数据格式，现在使用 json

AJAX 不单需要前端的技术，同时需要后端（服务器）的配合。服务器需要提供数据，数据是 AJAX 请求的响应结果。

1.3.2 AJAX 异步实现步骤

XMLHttpRequest 对象介绍

(1) 创建对象方式

```
var xmlhttp = new XMLHttpRequest();
```

(2) onreadystatechange 事件

当请求被发送到服务器时，我们需要执行一些基于响应的任务。每当 readyState 改变时，就会触发 onreadystatechange 事件。此事件可以指定一个处理函数 function。

通过判断 XMLHttpRequest 对象的状态，获取服务端返回的数据。

语法：

```
xmlHttp.onreadystatechange= function() {  
    if( xmlHttp.readyState == 4 && xmlHttp.status == 200){  
        处理服务器返回数据  
    }  
}
```

下面是 XMLHttpRequest 对象的三个重要的属性：

属性说明：

onreadystatechange 属性：一个 js 函数名 或 直接定义函数，每当 readyState 属性改变时，就会调用该函数

readyState 属性：

存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。

- 0：请求未初始化，创建异步请求对象 `var xmlHttp = new XMLHttpRequest()`
- 1：初始化异步请求对象， `xmlHttp.open(请求方式, 请求地址, true)`
- 2：异步对象发送请求， `xmlHttp.send()`
- 3：异步对象接收应答数据 从服务端返回数据。XMLHttpRequest 内部处理。
- 4：异步请求对象已经将数据解析完毕。 此时才可以读取数据。

status 属性：

200: "OK"

404: 未找到页面

（3） 初始化请求参数：

方法：

`open(method,url,async)` ： 初始化异步请求对象

参数说明：

- **method**: 请求的类型；GET 或 POST
- **url**: 服务器的 servlet 地址
- **async**: true（异步）或 false（同步）

例如：

```
xmlHttp.open("get","http:192.168.1.20:8080/myweb/query",true)
```

（4） 发送请求

`xmlHttp.send()`

(5) 接收服务器响应的数据

如需获得来自服务器的响应，请使用 `XMLHttpRequest` 对象的 `responseText` 或 `responseXML` 属性。

`responseText`: 获得字符串形式的响应数据

`responseXML`: 获得 XML 形式的响应数据

1.4 AJAX 实例

1.4.1 全局刷新计算 bmi

需求：计算某个用户的 BMI。用户在 `jsp` 输入自己的身高，体重；`servlet` 中计算 BMI，并显示 BMI 的计算结果和建议。

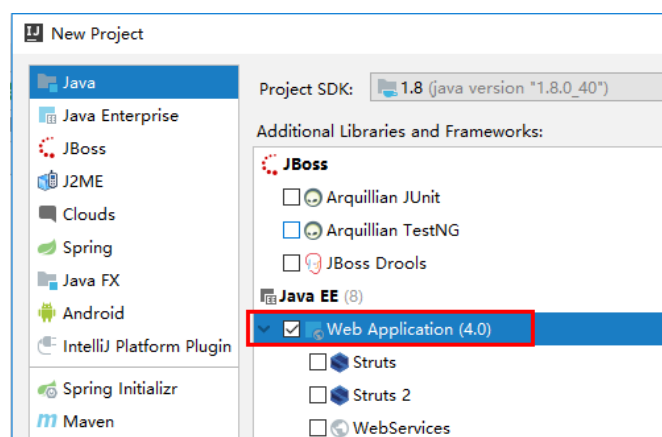
BMI 指数（即身体质量指数，英文为 `BodyMassIndex`，简称 BMI），是用体重公斤数除以身高米数平方得出的数字，是目前国际上常用的衡量人体胖瘦程度以及是否健康的一个标准

成人的 BMI 数值：

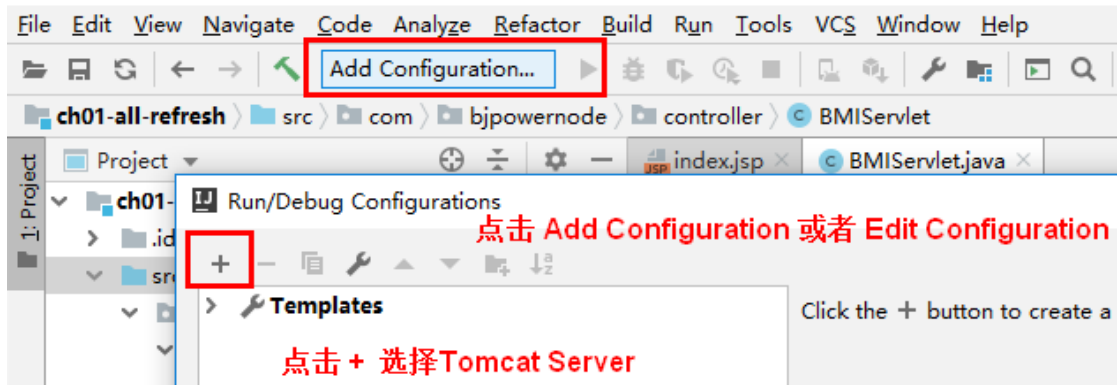
- 1) 过轻：低于 18.5
- 2) 正常：18.5-23.9
- 3) 过重：24-27
- 4) 肥胖：28-32
- 5) 非常肥胖,高于 32

开发步骤：

1.在 idea 中创建新的工程，名称：ch01-bmi-ajax



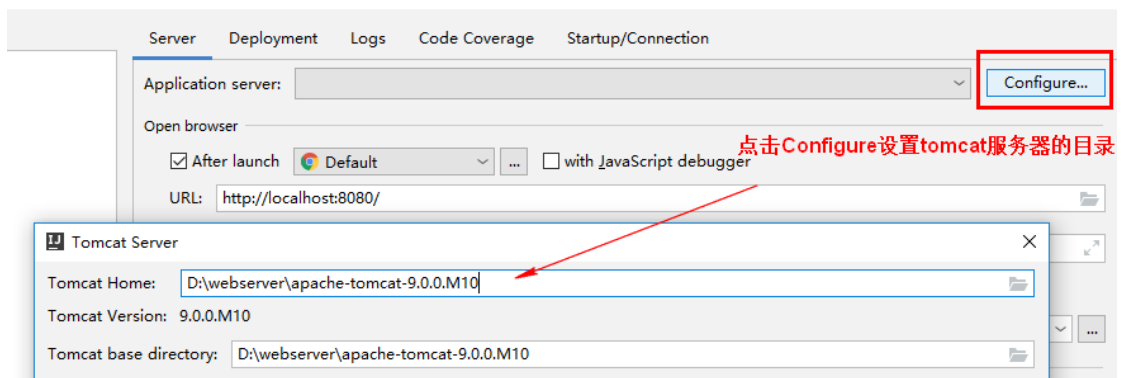
2.配置 tomcat 服务器，如果已经配置，省略此步骤。



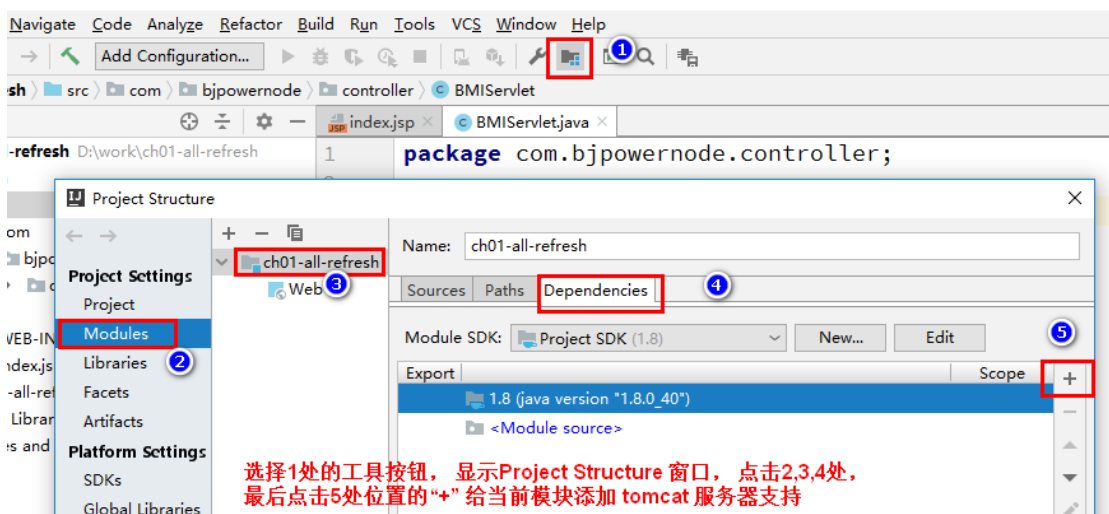
选择 Local



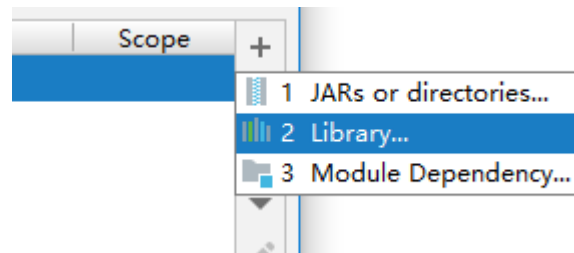
配置 tomcat 服务器的位置



Module 添加 tomcat 支持



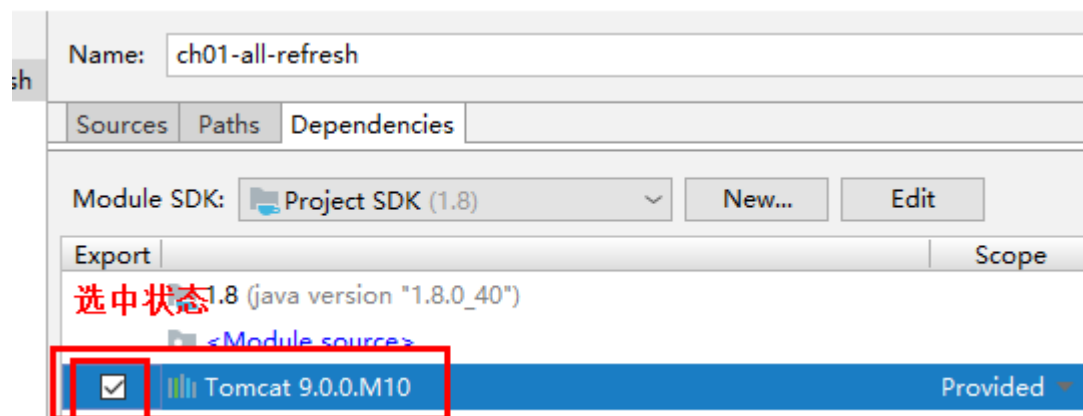
出现窗口



选择 2 Library



确定使用 tomcat



3.创建 jsp, 定义 form, 有参数 name, weight, height

```
<body>
<div style="...">
  <form action="bmiServlet" method="get">
    姓名: <input type="text" name="name"> <br>
    身高(米): <input type="text" name="height"> <br>
    体重(公斤): <input type="text" name="weight"> <br>
    <input type="submit" value="计算 bmi ">
  </form>
</div>
</body>
```

4.创建 Servlet, 名称 BMIServlet


```

public class BMIServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse
resp)
        throws ServletException, IOException {
        String strName = req.getParameter("name");
        String strWeight = req.getParameter("weight");
        String strHeight = req.getParameter("height");
        //计算 bmi
        float weight = Float.parseFloat(strWeight);
        float height = Float.parseFloat(strHeight);
        float bmi = weight / (height * height);
        System.out.println(String.format("%s 的 bmi%s",strName,bmi));

        String msg = "";
        if( bmi < 18.5 ){
            msg = "过瘦";
        } else if( bmi >= 18.5 && bmi < 23.9 ){
            msg = "正常";
        } else if( bmi >=23.9 && bmi <= 27){
            msg = "过重";
        } else if(bmi > 27 && bmi < 32 ){
            msg = "肥胖";
        } else {
            msg="非常肥胖";
        }

        req.setAttribute("msg", strName + "你的 bmi 是"+bmi+", "+msg);
        req.getRequestDispatcher("/result.jsp").forward(req,resp);
    }
}

```

5.注册 servlet

```

<servlet>
    <servlet-name>bmiServlet</servlet-name>
    <servlet-class>com.bjpowernode.controller.BMIServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>bmiServlet</servlet-name>
    <url-pattern>/bmiServlet</url-pattern>
</servlet-mapping>

```

6.创建 result.jsp

web 目录下创建 result.jsp 文件

```
<body>
    查看bmi: ${msg}
</body>
```

6.配置运行程序，输入参数。显示 bmi

1.4.2 使用 HttpServletResponse 响应输出

1.新建 jsp: indexPrint.jsp

```
<div style="margin-left: 500px">
    <form action="bmiServletPrint" method="get">
        姓名: <input type="text" name="name"> <br>
        身高: <input type="text" name="height"> (米)<br>
        体重: <input type="text" name="weight"> (公斤)<br>
        <br>
        <input type="submit" value="计算 bmi ">
    </form>
</div>
```

2.新建 Servlet, 名称 BMIServletPrint

```
public class BMIServletPrint extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
        String strName = req.getParameter("name");
        String strWeight = req.getParameter("weight");
        String strHeight = req.getParameter("height");

        //计算 bmi
        float weight = Float.parseFloat(strWeight);
        float height = Float.parseFloat(strHeight);
        float bmi = weight / (height * height);
        System.out.println(String.format("%s 的 bmi%s",strName,bmi));

        String msg = "";
        if( bmi < 18.5 ){
            msg = "过瘦";
        } else if( bmi >= 18.5 && bmi < 23.9 ){
            msg = "正常";
        } else if( bmi >=23.9 && bmi <= 27){
            msg = "过重";
        } else if(bmi > 27 && bmi < 32 ){
```

```

        msg = "肥胖";
    } else {
        msg="非常肥胖";
    }
    String res = strName + "你的 bmi 是"+bmi+", "+msg;
    resp.setContentType("text/html;charset=utf-8");
    PrintWriter pw = resp.getWriter();
    pw.println(res);
    pw.flush();
    pw.close();
}
}

```

3.注册 Servlet

```

<servlet>
    <servlet-name>bmiServletPrint</servlet-name>
    <servlet-class>com.bjpowernode.controller.BMIServletPrint</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>bmiServletPrint</servlet-name>
    <url-pattern>/bmiServletPrint</url-pattern>
</servlet-mapping>

```

1.4.3 使用 ajax 请求，计算 bmi

1.新建 ajax.jsp

```

<body>
    <div style="margin-left: 500px">
        <!-- 没有form -->
        姓名: <input type="text" id="name" > <br>
        身高: <input type="text" id="height" > (米)<br>
        体重: <input type="text" id="weight" > (公斤)<br>
        <br>
        <input type="button" value="计算 bmi " onclick="doAjax()">
        <br>
        <div id="dataDiv">等待更新数据.....</div>
    </div>
</body>

```

2.在 ajax.jsp 的 head 部分指定 doAjax()函数

```

<head>
  <title>全局刷新</title>
  <script type="text/javascript">
    function doAjax() {
      //创建异步对象
      var xmlhttp = new XMLHttpRequest();
      //绑定事件
      xmlhttp.onreadystatechange = function () {
        alert( "处理请求的状态: " + xmlhttp.readyState)
      }
      //初始化参数
      xmlhttp.open("get","bmiAjax",true);
      //发送ajax异步请求
      xmlhttp.send();
    }
  </script>
</head>

```

3.复制 BMIServletPrint，重新命名 BMIServletAjax

代码不需要改动

4.注册 Servlet

```

<servlet>
  <servlet-name>bmiServletAjax</servlet-name>
  <servlet-class>com.bjpowernode.controller.BMIServletAjax</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>bmiServletAjax</servlet-name>
  <url-pattern>/bmiAjax</url-pattern>
</servlet-mapping>

```

5.在浏览器访问 ajax.jsp

在 BMIServletAjax 的第一行设置断点，然后在 jsp 中点击按钮，发起请求，观察浏览器中的弹出的内容变化

6.修改 ajax.jsp 中的 doAjax()函数

```

<script type="text/javascript">
  function doAjax() {
    //创建异步对象
    var xmlhttp = new XMLHttpRequest();
    //绑定事件
    xmlhttp.onreadystatechange = function () {
      alert( "处理请求的状态: " + xmlhttp.readyState
        + "|服务器端返回数据: "+xmlhttp.responseText);
    }
    //初始化参数
    xmlhttp.open("get",
      "bmiAjax?name=张三&height=1.8&weight=75",true);
    //发送 ajax 异步请求
  }

```

```
        xmlhttp.send();
    }
</script>
```

7.访问 ajax.jsp 请求

在 jsp 中点击按钮，发起请求，观察浏览器中的弹出的内容变化

8.获取 dom 对象 value 值

```
<script type="text/javascript">
    function doAjax() {
        //创建异步对象
        var xmlhttp = new XMLHttpRequest();
        //绑定事件
        xmlhttp.onreadystatechange = function () {
            alert( "处理请求的状态: " + xmlhttp.readyState
                + " | 服务器端返回数据: "+xmlhttp.responseText);
        }
        //初始化参数
        //获取页面 dom 中的数据
        var name = document.getElementById("name").value;
        var height = document.getElementById("height").value;
        var weight = document.getElementById("weight").value;
        var param = "name="+name+"&height="+height+"&weight="+weight;
        xmlhttp.open("get","bmiAjax?"+param,true);
        //发送 ajax 异步请求
        xmlhttp.send();
    }
</script>
```

9. 在浏览器测试发送 ajax 请求

10.修改 doAjax 函数

```
<script type="text/javascript">
    function doAjax() {
        //创建异步对象
        var xmlhttp = new XMLHttpRequest();
        //绑定事件
        xmlhttp.onreadystatechange = function () {
            if( xmlhttp.readyState == 4 && xmlhttp.status == 200){
                var data = xmlhttp.responseText
                document.getElementById("dataDiv").innerText = data;
            }
        }
    }
</script>
```

```

    }
    //初始化参数
    //获取页面 dom 中的数据
    var name = document.getElementById("name").value;
    var height = document.getElementById("height").value;
    var weight = document.getElementById("weight").value;
    var param = "name="+name+"&height="+height+"&weight="+weight;
    xmlhttp.open("get","bmiAjax?" + param, true);
    //发送 ajax 异步请求
    xmlhttp.send();
}
</script>

```

1.4.4 根据省份 id 查询省份名称

需求：用户在文本框输入省份的编号 id，在其他文本框显示省份名称

项目环境准备：

1) 数据库：springdb

2) 数据表：

省份信息表：

```

CREATE TABLE `province` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL COMMENT '省份名称',
  `jiancheng` varchar(255) DEFAULT NULL COMMENT '简称',
  `shenghui` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8;

```

城市信息表：

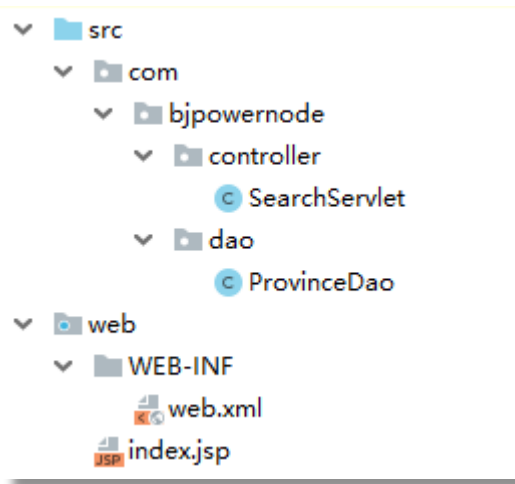
```

CREATE TABLE `city` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `provinceid` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=17 DEFAULT CHARSET=utf8;

```

表数据在课件资源目录\数据库文件\xxx.sql 文件

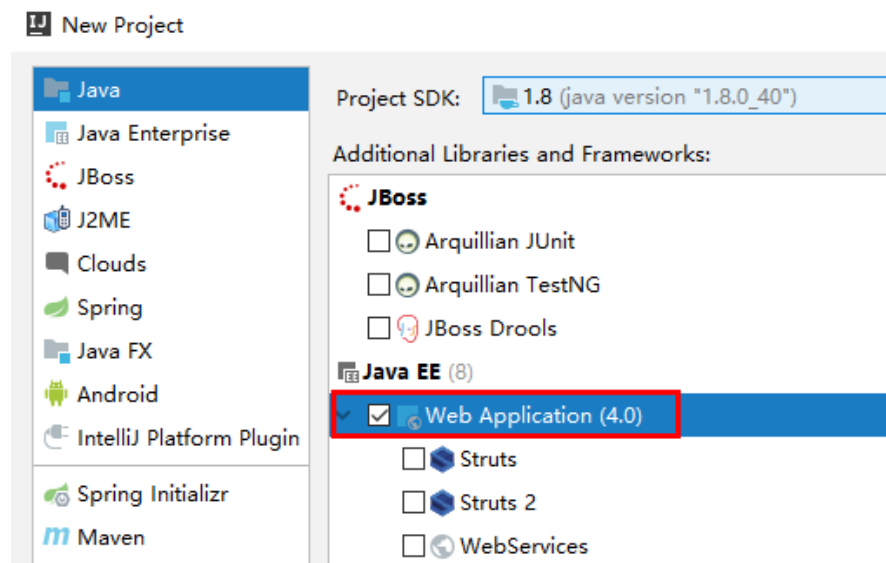
项目结构：



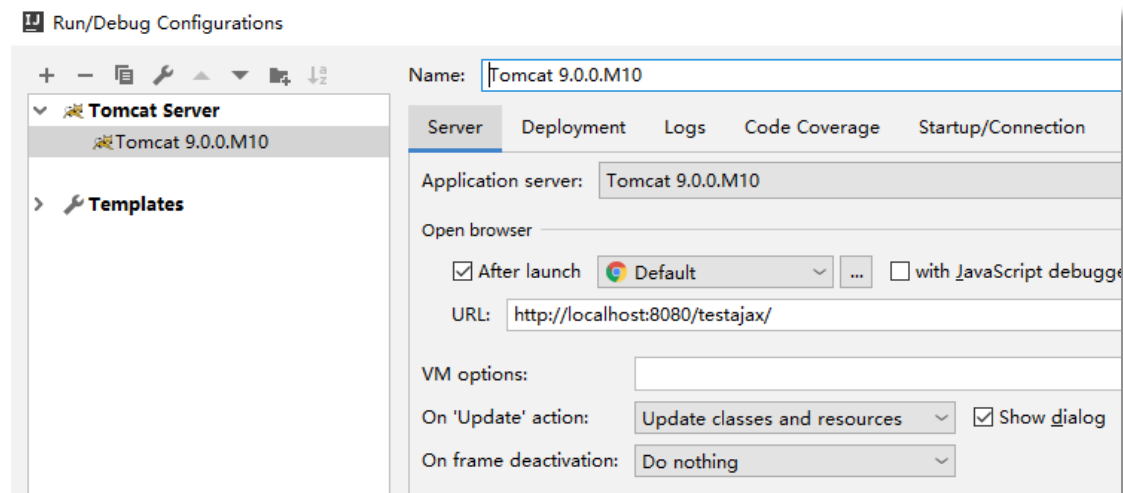
项目是一个 web 应用， index.jsp 发送请求， SearchServlet 接收请求， 调用 ProvinceDao 从数据库获取指定 id 的省份名称

实现步骤：

1. 在 idea 新建 web application：项目名称 ajaxweb



2. 配置 tomcat 服务器



3. 在 index.jsp 中创建 XMLHttpRequest 对象

定义表单：

```
<div align="center">
  <table border="1">
    <tr>
      <td>省份编号: </td>
      <td><input type="text" id="proid">
        <input type="button" value="搜索" onclick="search()">
      </td>
    </tr>
    <tr>
      <td>省份名称: </td>
      <td><input type="text" id="proname" /> </td>
    </tr>
    <tr>
      <td>省份简称: </td>
      <td><input type="text" id="projiancheng" /> </td>
    </tr>
  </table>
</div>
```

创建 XMLHttpRequest 对象


```

<script type="text/javascript">
    function search(){
        var proid = document.getElementById("proid").value;
        //创建异步请求对象
        var xmlHttp = new XMLHttpRequest();
        //绑定事件
        xmlHttp.onreadystatechange= function () {
            if(xmlHttp.readyState==4 && xmlHttp.status==200){
                var data = xmlHttp.responseText;
                alert(data);
            }
        }
        //初始请求参数
        xmlHttp.open("get","searchProvince?proid="+proid,true);
        //发送请求
        xmlHttp.send();
    }
</script>

```

5. 创建 Servlet 处理 Ajax 请求。

```

public class SearchServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String provinceName = "无数据";
        //获取参数 proid
        String param = request.getParameter("proid");
        if("1".equals(param) ){
            provinceName="河北";
        }
        response.setContentType("text/html;charset=utf-8");
        PrintWriter pw = response.getWriter();
        pw.println(provinceName);
        pw.flush();
        pw.close();
    }
}

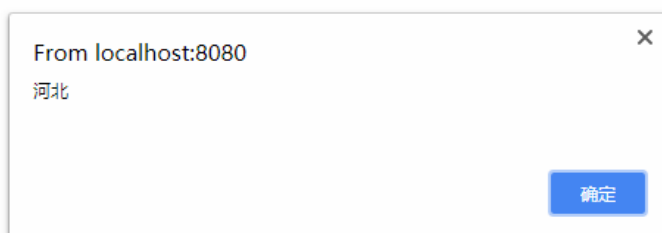
```

6. web.xml 文件，注册 servlet

```
<servlet>
    <servlet-name>SearchServlet</servlet-name>
    <servlet-class>com.bjpowernode.controller.SearchServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>SearchServlet</servlet-name>
    <url-pattern>/searchProvince</url-pattern>
</servlet-mapping>
```

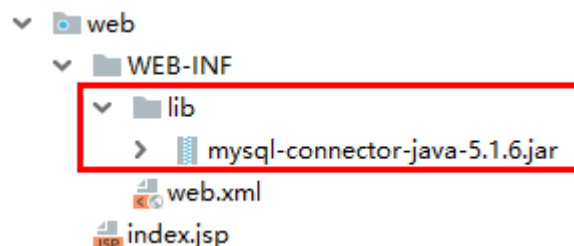
7. 发布应用到 tomcat 服务器，在浏览器访问 index.jsp，得到省份名称

省份编号：	<input type="text" value="1"/>	<input type="button" value="搜索"/>
省份名称：	<input type="text"/>	
省份简称：	<input type="text"/>	



8. 添加 mysql 驱动

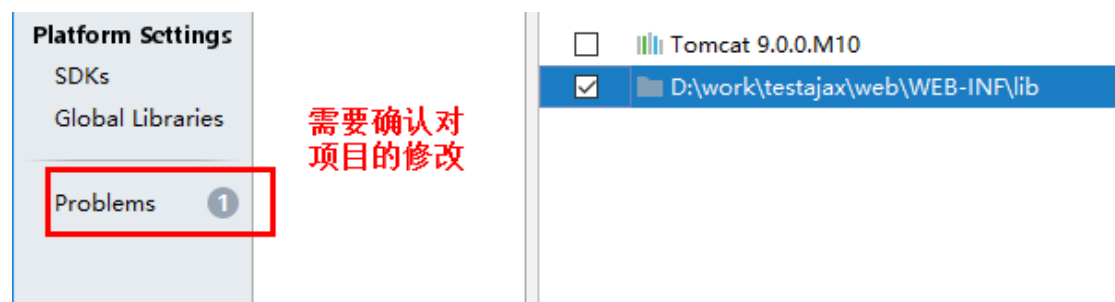
可以在 WEB-INF 目录下创建 lib 文件，用来存放 jar 文件，把准备好的 mysql-connector-java-5.1.6.jar 拷贝到 lib 目录下。



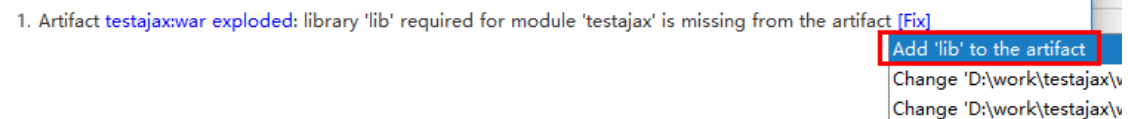
在 Project Structure 窗口中，选择你的 modules，选择 Dependencies



确认修改:



点击“Fix”后选择 Add lib to the artifact



9. 创建类 ProvinceDao 访问数据库

方法定义:

```
/**
 * 使用省份编号获取省份名称
 * @param proid 省份id
 * @return
 */
public String selectProvinceName(Integer proid){
```

定义变量:

```

Connection conn = null;
PreparedStatement pst = null;
ResultSet rs = null;
String url="jdbc:mysql://localhost:3306/springdb";
String username="root";
String password="123456";
//方法返回值
String retName = "";

```

访问数据库:

```

try {
    Class.forName("com.mysql.jdbc.Driver");
    conn = DriverManager.getConnection(url,username,password);

    String sql="select name from province where id=?";
    pst = conn.prepareStatement(sql);
    pst.setInt(1,proid);

    rs = pst.executeQuery();
    if(rs.next()){
        retName = rs.getString("name");
    }
} catch (Exception e){
    e.printStackTrace();
} finally {
    try{
        if( rs!=null ){

```

finally 关闭资源

```

} finally {
    try{
        if( rs!=null ){
            rs.close();
        }
        if( pst != null){
            pst.close();
        }
        if( conn != null){
            conn.close();
        }
    } catch (Exception ex){
        ex.printStackTrace();
    }
}

```

返回结果:

```
return retName;
```

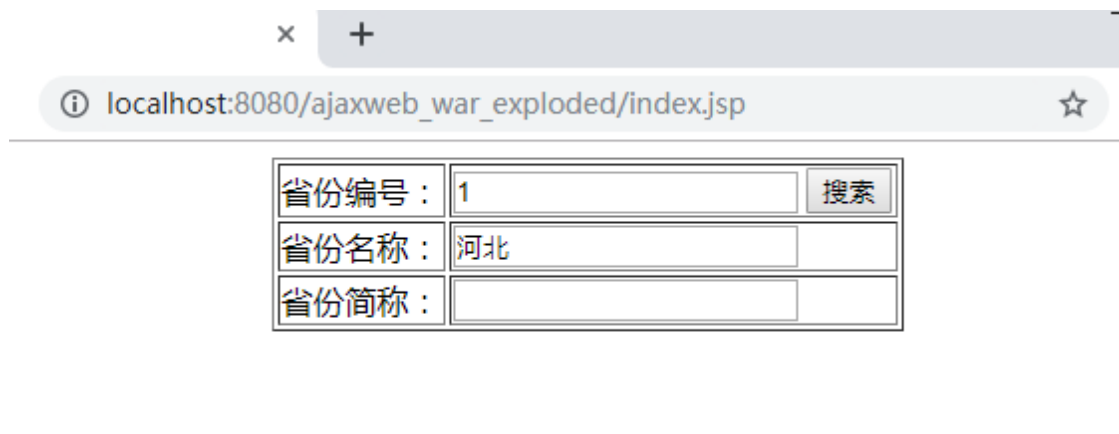
10. 修改之前创建的 Servlet

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String provinceName = "无数据";
    //获取参数 proid
    String param = request.getParameter("proid");
    if(param != null ){
        ProvinceDao dao = new ProvinceDao();
        provinceName =dao.selectProvinceName(Integer.parseInt(param));
    }
    response.setContentType("text/html;charset=utf-8");
    PrintWriter pw = response.getWriter();
    pw.println(provinceName);
    pw.flush();
    pw.close();
}
```

11. 修改 index.jsp 的 js 代码

```
//创建异步请求对象
var xmlhttp = new XMLHttpRequest();
//绑定事件
xmlhttp.onreadystatechange= function () {
    if(xmlhttp.readyState==4 && xmlhttp.status==200){
        var data = xmlhttp.responseText;
        document.getElementById("proname").value=data;
    }
}
```

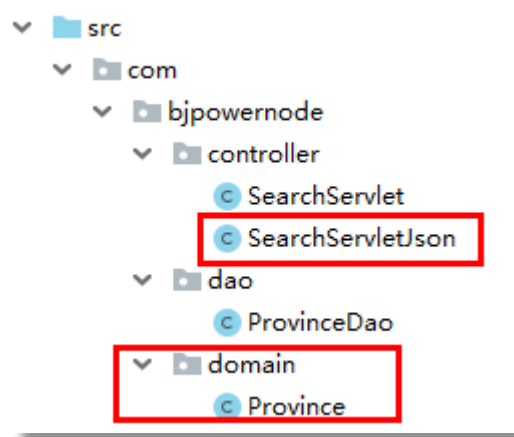
12. 部署项目，在浏览器访问应用



1.4.5 使用 json 作为数据交换格式

需求：根据省份编号 id，查询省份的全部数据，数据格式 json

项目结构：





实现步骤：


1. 添加处理 json 的工具库

jackson：是非常有名的处理 json 的工具库。使用 jackson 可以实现 java 对象到 json 格式字符串的转换，也可以实现 json 字符串转为 json 对象。

把下面三个 jar 文件复制到/WEB-INF/lib 目录中。

 jackson-annotations-2.9.0.jar

 jackson-core-2.9.0.jar

 jackson-databind-2.9.0.jar

其他步骤同 添加 mysql 驱动

2. 创建实体类 Province

```
public class Province {  
    // 主键 id  
    private Integer id;  
    // 省份名称  
    private String name;  
    // 省份简称  
    private String jiancheng;  
    // 省会名称  
    private String shenghui;  
    //set | get 方法
```

3. 在 ProvinceDao 中增加方法，返回对象

方法定义：

```
/**  
 * 使用省份编号获取省份对象  
 * @param proid 省份id  
 * @return  
 */  
public Province selectProvinceObject(Integer proid){
```

数据库操作：

```
Province pro = new Province();  
  
try {  
    Class.forName("com.mysql.jdbc.Driver");  
    conn = DriverManager.getConnection(url,username,password);  
  
    String sql="select id,name,jiancheng,shenghui from province where id=?";  
    pst = conn.prepareStatement(sql);  
    pst.setInt(1,proid);  
  
    rs = pst.executeQuery();  
    if(rs.next()){  
        pro.setId(rs.getInt(1));  
        pro.setName(rs.getString(2));  
        pro.setJiancheng(rs.getString("jiancheng"));  
        pro.setShenghui(rs.getString("shenghui"));  
    }  
} catch (Exception e){  
    e.printStackTrace();  
} finally {
```

其他代码同 selectProvinceName() 方法。

4. 创建新的 Servlet 对象

```
public class SearchServletJson extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String json="{ }";
        //获取参数 proid
        String param = request.getParameter("proid");
        if( param !=null && !param.trim().isEmpty()){
            ProvinceDao dao = new ProvinceDao();
            Province pro = dao.selectProvinceObject(Integer.parseInt(param));
            if( pro != null){
                ObjectMapper om = new ObjectMapper();
                json = om.writeValueAsString(pro);
            }
            //java对象转为json格式字符串
        }
        response.setContentType("application/json;charset=utf-8");
        PrintWriter pw = response.getWriter();
        pw.println(json);
        pw.flush();
        pw.close();
    }
}
```

5. 创建 searchJson.jsp, 获取 json 数据

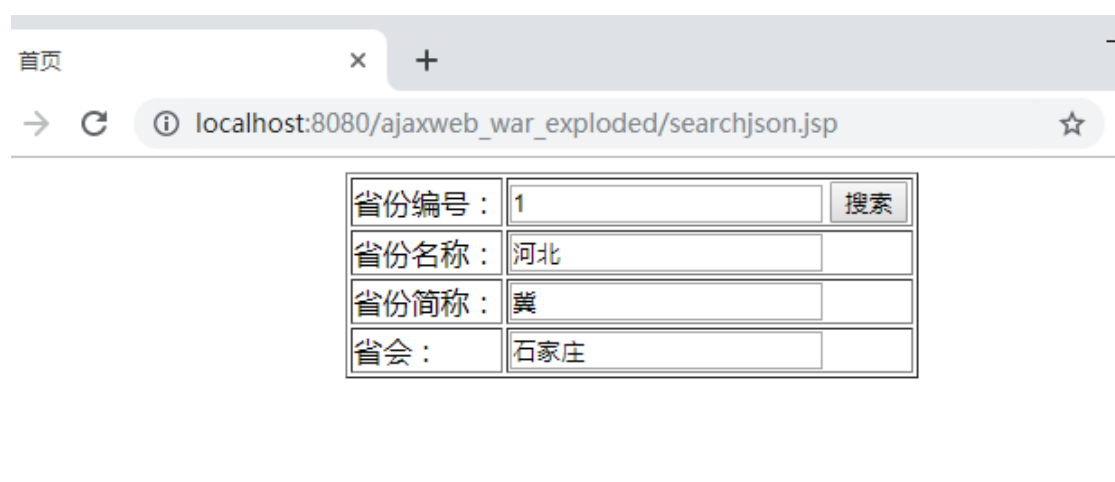
页面定义:

```
<div align="center">
    <table border="1">
        <tr>
            <td>省份编号: </td>
            <td><input type="text" id="proid">
                <input type="button" value="搜索" onclick="search()">
            </td>
        </tr>
        <tr>
            <td>省份名称: </td>
            <td><input type="text" id="praname" /> </td>
        </tr>
        <tr>
            <td>省份简称: </td>
            <td><input type="text" id="jiancheng" /> </td>
        </tr>
        <tr>
            <td>省会: </td>
            <td><input type="text" id="shenghui" /> </td>
        </tr>
    </table>
</div>
```


AJAX 请求处理:

```
<script type="text/javascript">
function search(){
    var proid = document.getElementById("proid").value;
    // 创建异步请求对象
    var xmlhttp = new XMLHttpRequest();
    // 绑定事件
    xmlhttp.onreadystatechange= function () {
        if(xmlhttp.readyState==4 && xmlhttp.status==200){
            var respText = xmlhttp.responseText;
            // 将json格式字符串转为 json object对象
            var jsonobj = eval("(" + respText + ")");
            // 处理结果函数
            callback(jsonobj);
        }
    }
    // 初始请求参数
    xmlhttp.open("get","searchProvinceJson?proid="+proid,false);
    // 发送请求
    xmlhttp.send();
}
function callback(jsonobj) {
    document.getElementById("proname").value = jsonobj.name;
    document.getElementById("jiancheng").value=jsonobj.jiancheng;
    document.getElementById("shenghui").value=jsonobj.shenghui;
}
</script>
```

6. 部署应用，浏览器访问



1.4.6 异步请求

XMLHttpRequest 对象 `open(method , url, true)` 第三个参数 `true` 表示异步请求

异步请求特点:

- 1) 某一个时刻，浏览器可以委托多个异步请求对象发送请求，无需等待请求处理完成。
- 2) 浏览器委托异步请求对象工作期间，浏览器处于活跃状态。可以继续向下执行其他命令。
- 3) 当响应就绪后再对响应结果进行处理

实现步骤:

1. 设置异步对象 `open` 方法第三个参数为 `true`

// 初始请求参数

```
xmlHttp.open("get","searchProvinceJson?proid="+proid,true);
```

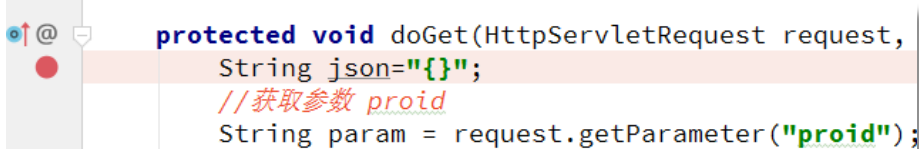
2. `send()`后面，增加 `alert()`

// 发送请求

```
xmlHttp.send();
```

```
alert("我是在异步请求之后的执行代码")
```

3. `SearchServletJson` 类的 `doGet` 方法第一个加入断点



```
protected void doGet(HttpServletRequest request,
String json="{ }");
// 获取参数 proid
String param = request.getParameter("proid");
```

4. 部署应用，在浏览器访问应用。

点击“搜索”按钮，请求发送到 Servlet，程序暂停执行，js 中 `alert` 执行继续执行，没有等待请求处理完成，浏览器窗口弹窗“**我是在异步请求之后的执行代码**”字符串。

1.4.7 同步请求

XMLHttpRequest 对象 `open(method , url, false)` 第三个参数 `false` 表示同步请求

同步请求特点:

- 1)某一个时刻，浏览器只能委托一个异步请求对象发送请求，必须等待请求处理完成。
- 2)浏览器委托异步请求对象工作期间，浏览器处于等待状态。不能执行其他命令。
- 3)不推荐使用。

实现步骤：同 1.4.3 步骤，需要 `open(method,url,false)` 第三个参数设为 `false`

1.5 练习

1. 在文本框内输入省份名称中的某几个字，把符合条件的省份名称显示一个<div>中
2. 在文本框输入省份名称，点击按钮使用 `alert` 显示出这个省份的城市数量， 例如输入山西，`alert` 弹窗显示 3 。表示山西省在 `city` 表中有三个城市。

第2章 jQuery

2.1 开篇基础

jQuery 是一款跨主流浏览器的 JavaScript 库，封装了 JavaScript 相关方法调用，简化 JavaScript 对 HTML DOM 操作

官网地址：<https://jquery.com/>

官网首页 jQuery 介绍：

What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

原文翻译：

jQuery 是一个快速，小巧，功能丰富的 JavaScript 库。它通过易于使用的 API 在大量浏览器中运行，使得 HTML 文档遍历和操作，事件处理，动画和 Ajax 变得更加简单。通过多功能性和可扩展性的结合，jQuery 改变了数百万人编写 JavaScript 的方式。

2.1.1 为什么[why]使用 jQuery

非常重要的理由就是：它能够兼容市面上主流的浏览器，IE 和 FireFox，Google 浏览器处理 AJAX，创建异步对象是不同的，而 jQuery 能够使用一种方式在不同的浏览器创建 AJAX 异步对象。

其他优点：

- (1) 写少代码,做多事情【write less do more】
- (2) 免费，开源且轻量级的 js 库，容量很小
- (3) 兼容市面上主流浏览器，例如 IE，Firefox，Chrome
- (4) 能够处理 HTML/JSP/XML、CSS、DOM、事件、实现动画效果，也能提供异步 AJAX 功能

- (5) 文档手册很全，很详细
- (6) 成熟的插件可供选择，多种 js 组件，例如日历组件（点击按钮显示下来日期）
- (7) 出错后，有一定的提示信息
- (8) 不用再在 html 里面通过<script>标签插入一大堆 js 来调用命令了

例如：使用 JavaScript 定位 DOM 对象常用的三种方式：

- (1) 通过 ID 属性：document.getElementById()
- (2) 通过 class 属性：getElementsByClassName()
- (3) 通过标签名：document.getElementsByTagName()

上面代码可以看出 JavaScript 方法名太长了，大小写的组合太多了，编写代码效率，容易出错。jQuery 分别使用\$("#id")， \$(".class 名")， \$("标签名) 封装了上面的 js 方法。

2.1.2 DOM 对象

文档对象模型（Document Object Model，简称 DOM），是 W3C 组织推荐的处理可扩展标志语言的标准编程接口。

通过 DOM 对 HTML 页面的解析，可以将页面元素解析为元素节点、属性节点和文本节点，这些解析出的节点对象，即 DOM 对象。DOM 对象可以使用 JavaScript 中的方法。

2.1.3 JavaScript 对象和 jQuery 对象

用 JavaScript 语法创建的对象叫做 JavaScript 对象，JavaScript 对象只能调用 JavaScript 对象的 API。

用 JQuery 语法创建的对象叫做 JQuery 对象，jQuery 对象只能调用 jQuery 对象的 API。jQuery 对象是一个数组。在数组中存放本次定位的 DOM 对象。

JQuery 对象与 JavaScript 对象是可以互相转化的，一般地，由于 JQuery 用起来更加方便，我们都是将 JavaScript 对象转化成 JQuery 对象

2.1.4 获取 jQuery

官网下载地址：<https://jquery.com/download/>

jQuery

For help when upgrading jQuery, please see the [upgrade guide](#) most relevant to your version.

[Download the compressed, production jQuery 3.4.1](#) 压缩的文件，线上环境使用

[Download the uncompressed, development jQuery 3.4.1](#) 未压缩文件，开发阶段使用

[Download the map file for jQuery 3.4.1](#)

jQuery 的不同版本中，2.xx 不再支持 IE6/7/8 浏览器。现阶段 IE6/7/8 已经是淘汰的，

非主流。可以不用考虑兼容问题。

对于每一个同一版本号的 jQuery，其库又分为两个。一个是未压缩版，可查看源码，开发时使用；一个是压缩版，将注释、空格均做了删除，将变量字符数减少，产品上线时使用。

2.1.5 牛刀小试

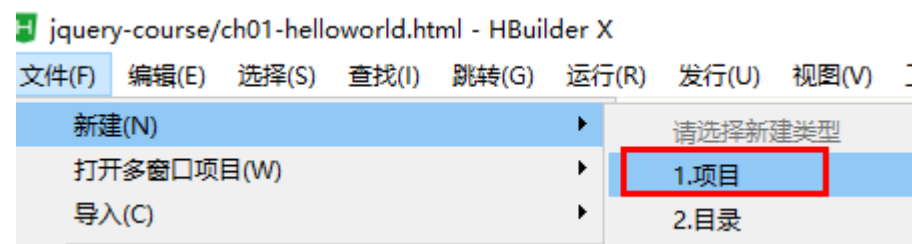
编写 jQuery 的工具很多，能编写 HTML 的工具都支持 jQuery。例如记事本，EditPlus, webStorm, Visual Studio Code, HBuilder, HBuilderX, IDEA。

单独学习 jQuery 库使用，可以轻量的开发工具，例如 EditPlus, HBuilder, HbuilderX 编写项目可以使用集成开发工具，例如在 IDEA, Eclipse, MyEclipse, WebStorm 等

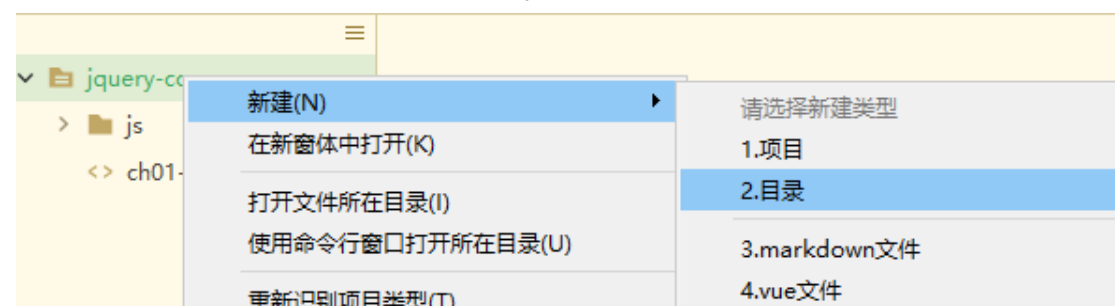
第一个例子完成：浏览器完全装载 html 页面 DOM 后,显示一个提示信息框

实现步骤：

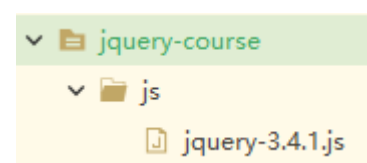
1. 使用 HBuilder 或 HbuilderX, idea 都可以，以 HbuilderX 为工具，创建一个项目（名称：jquery-course），给项目选择一个文件存放目录。



2. 在项目中再创建一个目录
右键项目名称—新建—目录，常用名称为 js



- 3.拷贝下载的 jQuery.js 文件到目录



4. 使用 jQuery，首先要将 jQuery 库引入。使用如下语句：
<script type="text/javascript" src="js/jquery-3.4.1.js"></script>
5. \$(document)，将 DOM 对象 document 转换为 jQuery 对象。\$(document).ready()函数是当

DOM 对象加载完毕后，马上执行的函数。

`$(document).ready()`与`$(())`、`jQuery()`、`window.jQuery()`是等价的，所以`$(document).ready()`可以写成 `$(function() { alert("Hello jQuery") });`;

6. 完整代码

```
<html>
  <head>
    <meta charset="utf-8">
    <title>第一个jQuery例子</title>
    <script type="text/javascript" src="js/jquery-3.4.1.js"></script>
    <script type="text/javascript">
      //把JavaScript中的document对象转为jQuery对象
      //使用jQuery中的方法ready，当DOM对象全部创建完成后,执行function中的内容
      $(document).ready(function(){
        alert("Hello jQuery")
      })
    </script>
  </head>
  <body>
  </body>
</html>
```

2.1.6 DOM 对象和 jQuery 对象

DOM 对象是用 JavaScript 语法创建的对象，也看做是 js 对象。

1. DOM 对象转换 jQuery 对象:

使用`$(DOM 对象)`方式，可以 DOM 对象转换为 jQuery 对象，转换为 jQuery 对象才可以使用 jQuery 中提供的方法，操作 DOM 对象。一般情况下，在命名 jQuery 对象时，为了与 DOM 对象进行区分，习惯性的以`$`开头，这不是必须的。

例:

新建 html 页面文件 `domTojQuery.html`

1. 页面加入按钮 button

```
<body>
  <div align="center">
    <input type="button" id="btn" value="我要成为jQuery对象" onclick="btnClick()"/>
  </div>
</body>
```

2. 转换 DOM 对象

```

<script type="text/javascript" src="js/jquery_3.4.1.js"></script>
<script type="text/javascript">
    function btnClick(){
        //使用js方法获取DOM对象
        var domBtn = document.getElementById("btn");
        alert("1=" + domBtn.value );
        //使用jQuery函数 $() 把DOM对象转换jQuery对象
        var $btn = $(domBtn);
        //调用jQuery方法val()
        alert("2=" + $btn.val());
    }
</script>

```

2. jQuery 对象转为 DOM 对象

jQuery 对象本身为数组对象，该数组中的第 0 个元素即为该 jQuery 对象对应的 DOM 对象。所以有两种方式可以获取到 DOM 对象：get(0) 方式与下标[0]

例：新建 html 文件 jQueryToDom.html

1. 页面添加 text，button

```

<body>
    <div align="center">
        <input type="button" id="btn" value="计算平方" onclick="btnClick()"/>
        <input type="text" id="txt" value="输入整数"/>
    </div>
</body>

```

2. jQuery 对象.get(0) 或 jQuery 对象[0] 均可完成 jQuery 对象转 DOM 对象

```

<script type="text/javascript" src="js/jquery_3.4.1.js"></script>
<script type="text/javascript">
    function btnClick(){
        //从jQuery对象数组中,第0个对象为DOM对象
        //var domObj = $("#txt").get(0);
        var domObj = $("#txt")[0];
        //使用DOM对象的属性value
        var num = domObj.value;
        domObj.value = num * num;
    }
</script>

```


2.2 选择器

选择器: 就是定位条件; 通知 jquery 函数定位满足条件的 DOM 对象

2.2.1 基本选择器

根据 ID, class 属性, 标签类型名定位 HTML 元素, 转为 jQuery 对象.

1.id 选择器

语法: `$("#id")`

2.class 选择器

语法: `$(".class 名称")`

3. 标签选择器

语法: `$("标签名")`

例: 新建 selector.html

1.在页面 head 部分加入 css

```
<style type="text/css">
    div{
        background: gray;
        width:200px;
        height:100px;
    }
</style>
```

2.加入 jQuery 引用

```
<script type="text/javascript" src="js/jquery_3.4.1.js"></script>
```

3.body 部分定义 div

```

<body>
  <div id="one">我是id=one的div</div>
  <br>
  <div class="two">我是class=two的div</div>
  <br>
  <div>我是没有id, class的div</div>
  <br>
  <span>我是一行数据</span>
  <br>
  操作按钮: <br>
  <input type="button" value="选取id=one" onclick="fun1()" />
  <input type="button" value="选取class=two" onclick="fun2()" />
  <input type="button" value="选取div" onclick="fun3()" />
</body>

```

4. 创建 js 函数

```

<script type="text/javascript">
  function fun1(){
    $("#one").css("background","blue");
  }
  function fun2(){
    $(".two").css("background","red");
  }
  function fun3(){
    //所有的div, $("div")是jQuery数组,对数组内全部成员统一处理
    $("div").css("background","orange");
  }
</script>

```

4. 所有选择器

语法: \$("*") 选取页面中所有 DOM 对象。

5. 组合选择器

组合选择器是多个被选对象间使用逗号分隔后形成的选择器,可以组合 id, class, 标签名等。

语法: \$("id,class,标签名")

例:

1. 上面的 selector.html 页面中加入按钮

```

<input type="button" value="所有DOM" onclick="fun4()" />
<input type="button" value="选取两个DOM对象" onclick="fun5()" />

```

2.增加 js 函数

```
function fun4(){
    //页面中全部
    $("*").css("background","yellow");
}
function fun5(){
    // id是one, 标签<span>
    $("#one,span").css("background","pink")
}
```

2.2.2 表单选择器

表单相关元素选择器是指文本框、单选框、复选框、下拉列表等元素的选择方式。该方法无论是否存在表单<form>，均可做出相应选择。表单选择器是为了能更加容易地操作表单，表单选择器是根据元素类型来定义的

```
<input type="text">
<input type="password">
<input type="radio">
<input type="checkbox">
<input type="button">
<input type="file">
<input type="submit">
<input type="reset">
```

\$("#tr"): 不能用，tr 不是 input 标签

语法： \$("#:type 属性值")

例如：

\$("#text")选取所有的单行文本框

\$("#password")选取所有的密码框

\$("#radio")选取所有的单选框

\$("#checkbox")选取所有的多选框

\$("#file")选取所有的上传按钮

例：

新建 form.html

页面定义元素：

```

文本框 : <input type="text" value="我是type=text"><br>
性别 : <br>
<input type="radio" name="sex" value="man">男<br>
<input type="radio" name="sex" value="woman">女<br>
爱好 : <br>
<input type="checkbox" value="bike">骑行<br>
<input type="checkbox" value="football" >足球<br>
<input type="checkbox" value="music" >音乐<br>
<br>
<p>功能按钮</p>
<input type="button" value="读取text值" onclick="fun1()" />
<input type="button" value="读取radio值" onclick="fun2()" />
<input type="button" value="读取checkbox" onclick="fun3()" />

```

定义 js 函数:

```

<script type="text/javascript">
    function fun1(){
        var $obj = $(":text");
        //获取第一个DOM对象的值
        alert( $obj.val() );
    }
    function fun2(){
        var $obj = $(":radio");
        for(var i=0;i<$obj.length;i++){
            //jQuery对象转为DOM对象
            var obj = $obj[i];
            alert(obj.value);
        }
    }
    function fun3(){
        var $obj = $(":checkbox");
        for(var i=0;i<$obj.length;i++){
            //jQuery对象转为DOM对象
            var obj = $obj[i];
            alert(obj.value);
            //jQuery对象的用法
            //alert( $($obj[i]).val() );
        }
    }
}
</script>

```

2.3 过滤器

jQuery 对象中存储的 DOM 对象顺序与页面标签声明位置关系

```
<div>1</div>    dom1
<div>2</div>    dom2
<div>3</div>    dom3
```

```
$("div") == [dom1,dom2,dom3]
```

过滤器就是过滤条件，对已经定位到数组中 DOM 对象进行过滤筛选，过滤条件不能独立出现在 jquery 函数，如果使用只能出现在选择器后方。

2.3.1 基本过滤器

1.选择第一个 first, 保留数组中第一个 DOM 对象

语法: \$("选择器:first")

2.选择最后个 last, 保留数组中最后 DOM 对象

语法: \$("选择器:last")

3.选择数组中指定对象

语法: \$("选择器:eq(数组索引)")

4.选择数组中小于指定索引的所有 DOM 对象

语法: \$("选择器:lt(数组索引)")

5.选择数组中大于指定索引的所有 DOM 对象

语法: \$("选择器:gt(数组索引)")

实例操作

1.定义样式

```
<style type="text/css">
  div {
    background: gray;
  }
</style>
```

2. 页面加入 div

```
<div id="one">我是div-0</div>
<div id="two">我是div-1</div>
<div>
  我是第三个div-2
  <div class="son">我是div-3</div>
  <div class="son">我是div-4</div>
</div>
<div>我是div-5</div>
<br>
<span>我是span</span>
<br>
<p>功能按钮</p>
<input type="button" id="btn1" value="选择第一个div" /> <br>
<input type="button" id="btn2" value="选择最后一个div" /><br>
<input type="button" id="btn3" value="选择索引等于3的div" /><br>
<input type="button" id="btn4" value="选择索引小于3的div" /><br>
<input type="button" id="btn5" value="选择索引大于3的div" /><br>
```

3. 定义 js 函数

```

<script type="text/javascript">
    $(function(){
        //jQuery绑定事件
        $("#btn1").click(function(){
            $("div:first").css("background","blue")
        })

        $("#btn2").click(function(){
            $("div:last").css("background","pink");
        })

        $("#btn3").click(function(){
            $("div:eq(3)").css("background","orange");
        })

        $("#btn4").click(function(){
            $("div:lt(3)").css("background","yellow");
        })

        $("#btn5").click(function(){
            $("div:gt(3)").css("background","yellow");
        })
    })
</script>

```

2.3.2 表单对象属性过滤器

1. 选择可用的文本框

`$(":text:enabled")`

2. 选择不可用的文本框

`$(":text:disabled")`

3. 复选框选中的元素

`$(":checkbox:checked")`

4.选择指定下拉列表的被选中元素

选择器>option:selected

例：

创建 filterForm.html

页面：

```
<p>文本框</p>
<input type="text" id="text1" value="text1" /> <br>
<input type="text" id="text2" value="text2" disabled /><br>
<input type="text" id="text3" value="text3" /><br>
<input type="text" id="text4" value="text4" disabled /><br>
<br>
<p>复选框</p>
<input type="checkbox" value="游泳" />游泳<br>
<input type="checkbox" value="健身" checked />健身<br>
<input type="checkbox" value="电子游戏" checked />电子游戏<br>

<br>
<p>下拉框</p>
<select id="lang">
    <option value="java" >java语言</option>
    <option value="go" selected>go语言</option>
    <option value="sql">sql语言</option>
</select>
<p>功能按钮</p>
<button id="btn1">所有可用的text设值hello</button> <br>
<button id="btn2">显示被选中的复选框的值</button><br>
<button id="btn3">显示下拉列表选中的值</button><br>
```

js 函数


```

<script type="text/javascript" src="js/jquery-3.4.1.js"></script>
<script type="text/javascript">
    $(function(){
        //jQuery绑定事件
        $("#btn1").click(function(){
            $(".text:enabled").val("Hello")
        })

        $("#btn2").click(function(){
            var $obj = $(".checkbox:checked");
            $obj.each(function(i,n){
                alert("第"+i+"个成员，DOM值是："
                    +n.value+" jQuery对象取值："+$(n).val());
            })
        })

        $("#btn3").click(function(){
            // 有选中的第一个select的option
            var $obj = $("select > option:selected");
            alert($obj.val() + " " + $obj.text());

            //选择 id=lang的select
            //var $obj=$("#lang > option:selected");
            //alert($obj.val()+" " + $obj.text());
        })
    })
</script>

```

2.4 函数

2.4.1 第一组

1. val

操作数组中 DOM 对象的 value 属性。

\$(选择器).val()：无参数调用形式，读取数组中第一个 DOM 对象的 value 属性值

\$(选择器).val(值)：有参形式调用;对数组中所有 DOM 对象的 value 属性值进行统一赋值

2.text

操作数组中所有 DOM 对象的【文字显示内容属性】

`$(选择器).text()`:无参数调用, 读取数组中所有 DOM 对象的文字显示内容, 将得到内容拼接为一个字符串返回

`$(选择器).text(值)`:有参数方式, 对数组中所有 DOM 对象的文字显示内容进行统一赋值

3.attr

对 val, text 之外的其他属性操作

`$(选择器).attr(“属性名”)`: 获取 DOM 数组第一个对象的属性值

`$(选择器).attr(“属性名”, “值”)`: 对数组中所有 DOM 对象的属性设为新值

例:

创建 fun1.html

样式:

```
<style type="text/css">
    div {
        background: blue;
    }
</style>
```

页面:

```
<p>文本框val</p>
<input type="text" value="刘备" /> <br>
<input type="text" value="关羽" /><br>
<input type="text" value="张飞" /><br>
<p>文本数据text</p>
<div>我是第一个div</div>
<div>我是第二个div</div>
<div>我是第三个div</div>
<p>图片</p>
<br>
<br>
<p>功能按钮</p>
<button id="btn1">获取第一个文本框的值</button> <br>
<button id="btn2">设置所有文本框为新值</button><br>
<button id="btn3">获取div的所有文本</button><br>
<button id="btn4">获取第一个div的文本</button><br>
<button id="btn5">设置div新文本</button><br>
<button id="btn6">设置img图片</button><br>
```

js 函数

```

<script type="text/javascript">
    $(function(){
        //jQuery绑定事件
        $("#btn1").click(function(){
            var txt = $(":text").val();
            alert("我是多个text的第一个："+txt);
        })
        $("#btn2").click(function(){
            $(":text").val("王者荣耀还是三国志")
        })
        $("#btn3").click(function(){
            alert($("#div").text());
        })
        $("#btn4").click(function(){
            alert( $("#div:first").text() )
        })
        $("#btn5").click(function(){
            $("#div").text("我是小强")
        })
        $("#btn6").click(function(){
            alert($("#img").attr("src"))
            $("#img").attr("src","img/ex3.jpg");
        })
    })
</script>

```

2.4.2 第二组

1.hide

\$(选择器).hide():将数组中所有 DOM 对象隐藏起来

2.show

\$(选择器).show():将数组中所有 DOM 对象在浏览器中显示起来

3.remove

\$(选择器).remove(): 将数组中所有 DOM 对象及其子对象一并删除

4.empty

\$(选择器).empty(): 将数组中所有 DOM 对象的子对象删除

5.append

为数组中所有 DOM 对象添加子对象

\$(选择器).append("<div>我动态添加的 div</div>")

6.html

设置或返回被选元素的内容（innerHTML）。

\$(选择器).html(): 无参数调用方法，获取 DOM 数组第一个匹配元素的内容。

\$(选择器).html(值): 有参数调用，用于设置 DOM 数组中所有元素的内容。

7.each

each 是对数组，json 和 dom 数组等的遍历,对每个元素调用一次函数。

语法 1: \$.each(要遍历的对象, function(index,element) { 处理程序 })

语法 2: jQuery 对象.each(function(index, element) { 处理程序 })

index: 数组的下标

element: 数组的对象

例:

新建 fun2.html

样式:

```
<style type="text/css">
    div {
        background: blue;
    }
</style>
```

页面:

```

<p>show和hide</p>
<div id="one">我是one</div>
<div id="two">我是two</div>
<p>remove和empty</p>
<select>
    <option>老虎</option>
    <option>狮子</option>
    <option>豹</option>
</select>
<br>
<select>
    <option>大洋洲</option>
    <option>欧洲</option>
    <option>美洲</option>
</select>
<br>
<p>append</p>
<div id="father" style="background: red;">我是父div</div>
<br>
<p>html</p>
<span>mysql是一个<b>数据库</b></span><br>
<span>使用jdbc访问数据库</span><br>

<p>功能按钮</p>
<button id="btn1">隐藏所有div</button> <br>
<button id="btn2">显示所有div</button><br>
<button id="btn3">清除所有下拉列表中option子标签</button><br>
<button id="btn4">删除所有下拉列表</button><br>
<button id="btn5">添加元素</button><br>
<button id="btn6">获取第一个span文本内容</button><br>
<button id="btn7">设置span的内容</button><br>
<button id="btn8">each循环</button><br>

```

js 函数

```

$("#btn1").click(function(){
    $("#div").hide();
})
$("#btn2").click(function(){
    $("#div").show();
})
$("#btn3").click(function(){
    $("#select").empty();
})
$("#btn4").click(function(){
    $("#select").remove();
})
$("#btn5").click(function(){
    $("#father").append("<input type='text' value='我是动态添加的input' />");
})

```

```

$("#btn6").click(function(){
    alert($("#span").html());
})
$("#btn7").click(function(){
    $("#span").html("我是span标签");
})
$("#btn8").click(function(){
    $("#span").each(function(i,n){
        alert("索引:"+i+" "+ n.innerHTML)
    })
})

```

each 用法

页面上加入 text

```

<p>文本框val</p>
<input type="text" value="刘备" /><br>
<input type="text" value="关羽" /><br>
<input type="text" value="张飞" /><br>

```

加入 三个按钮， 创建点击事件

```

$("#9").click(function(){
    //遍历数组
    var arr=["a","b","c"];
    $.each(arr,function(i,n){alert("数组下标："+i+",成员是："+n)})
})

```

```
$("#btn10").click(function(){
    //遍历json对象
    var arr={name:"zhangsan",age:20};
    $.each(arr,function(i,n){alert("key是："+i+", value是："+n)}))
})
```

```
$("#btn11").click(function(){
    //遍历dom数组
    $.each($(":text"),function(i,n){alert("下标是："+i+", dom的value是："+n.value)}))
})
```

或者

```
$("#btn11").click(function(){
    //遍历dom数组
    $(":text").each(function(i,n){alert("下标是："+i+", dom的value是："+n.value)}))
})
```

2.5 事件

为页面元素绑定事件，即对于指定页面元素，当某个事件发生后，执行指定动作

2.5.1 定义元素监听事件

语法：\$(选择器).监听事件名称(处理函数);

说明：监听事件名称是 js 事件中去掉 on 后的内容，js 中的 onclick 的监听事件名称是 click

例如：

为页面中所有的 button 绑定 onclick,并关联处理函数 fun1

```
$("#button").click(fun1)
```

为页面中所有的 tr 标签绑定 onmouseover，并关联处理函数 fun2

```
$("tr").mouseover(fun2)
```

2.5.2 on() 绑定事件

on() 方法在被选元素上添加事件处理程序。该方法给 API 带来很多便利，推荐使用该方法

语法：\$(选择器).on(event,.,data,function)

event: 事件一个或者多个，多个之间空格分开

data: 可选。规定传递到函数的额外数据，json 格式

function: 可选。规定当事件发生时运行的函数。

例：

新建 event.html

样式:

```
<style type="text/css">
    div {
        background: gray;
        width:200px;
        height:80px;
    }
</style>
```

页面:

```
<body>
    <p>功能按钮</p>
    <button id="btn1">动态添加Button</button>
    <br><br>
    <div id="mydiv">
    </div>
    <p>可以 click me</p>
</body>
```

js 函数

```
$(function(){
    $("#btn1").on("click",function(){
        $("#mydiv").append(
            "<button id='mybutton'>我是添加的NewButton</button>"
        );

        $("#mybutton").on("click",function(){
            alert("NewButton被点了");
        });
    })

    $("p").on("click",{name:"zhangsan"},function(event){
        // event.data 获取传入的参数 event是自定义名称
        alert(event.data.name)
    })
})
```

2.6 AJAX

jQuery 提供多个与 AJAX 有关的方法。通过 jQuery AJAX 方法, 您能够使用 HTTP Get 和 HTTP Post 从远程服务器上请求文本、HTML、XML 或 JSON 同时能够把接收的数据更新到 DOM 对象。

2.6.1 \$.ajax()

\$.ajax() 是 jQuery 中 AJAX 请求的核心方法，所有的其他方法都是在内部使用此方法。

语法：

\$.ajax({ name:value, name:value, ... })

说明：参数是 json 的数据，包含请求方式，数据，回调方法等

async：布尔值，表示请求是否异步处理。默认是 true

contentType：发送数据到服务器时所使用的内容类型。默认是：

"application/x-www-form-urlencoded"。

data：规定要发送到服务器的数据，可以是：string，数组，多数是 json

dataType：期望从服务器响应的数据类型。jQuery 从 xml, json, text,, html 这些中测试最可能的类型

"xml" - 一个 XML 文档

"html" - HTML 作为纯文本

"text" - 纯文本字符串

"json" - 以 JSON 运行响应，并以对象返回

error(xhr,status,error): 如果请求失败要运行的函数，其中 xhr, status, error 是自定义的形参名

success(result,status,xhr): 当请求成功时运行的函数，其中 result, status, xhr 是自定义的形参名

type：规定请求的类型（GET 或 POST 等），默认是 GET，get, post 不用区分大小写

url：规定发送请求的 URL。

以上是常用的参数。

error(), success()中的 xhr 是 XMLHttpRequest 对象。

2.6.2 \$.get()

\$.get() 方法使用 HTTP GET 请求从服务器加载数据。

语法：\$.get(url,data,function(data,status,xhr),dataType)

url 必需。规定您需要请求的 URL。

data 可选。规定连同请求发送到服务器的数据。

function(data,status,xhr)可选。当请求成功时运行的函数。data,status,xhr 是自定义形参名。

参数说明：

data - 包含来自请求的结果数据

status - 包含请求的状态 ("success"、"notmodified"、"error"、"timeout"、"parsererror")

xhr - 包含 XMLHttpRequest 对象

dataType 可选。规定预期的服务器响应的数据类型。默认地，jQuery 会智能判断。可能的类型：

"xml" - 一个 XML 文档

"html" - HTML 作为纯文本

"text" - 纯文本字符串

"json" - 以 JSON 运行响应，并以对象返回

2.6.3 \$.post()

\$.post() 方法使用 HTTP POST 请求从服务器加载数据。

语法：\$.post(URL,data,function(data,status,xhr),dataType)

参数同\$.get()

2.6.4 使用 AJAX 级联查询

效果图：

省份列表： 福建 ▼

城市列表： 请选择... ▼

- 请选择...
- 福州市
- 厦门市
- 泉州市
- 龙岩市

数据库：springdb

province：省份表

字段	索引	外键	触发器	选项	注释	SQL 预览		
名				类型	长度	小数点	不是 null	
▶ id				int	11	0	<input checked="" type="checkbox"/>	🔑 1
name				varchar	255	0	<input type="checkbox"/>	
jiancheng				varchar	255	0	<input type="checkbox"/>	
shenghui				varchar	255	0	<input type="checkbox"/>	

默认:

注释:

☒ 自动递增

city：城市表

字段	索引	外键	触发器	选项	注释	SQL 预览
名						
id						1
name						
provinceid						

默认:

注释:

☒ 自动递增

1.IDEA 创建 web 项目（ajax-jquery）

New Project

Project SDK: 1.8 (java version "1.8.0_40")

Additional Libraries and Frameworks:

JBoss

☐ Arquillian JUnit

☐ Arquillian TestNG

☐ JBoss Drools

Java EE (8)

☒ Web Application (4.0)

☐ Struts

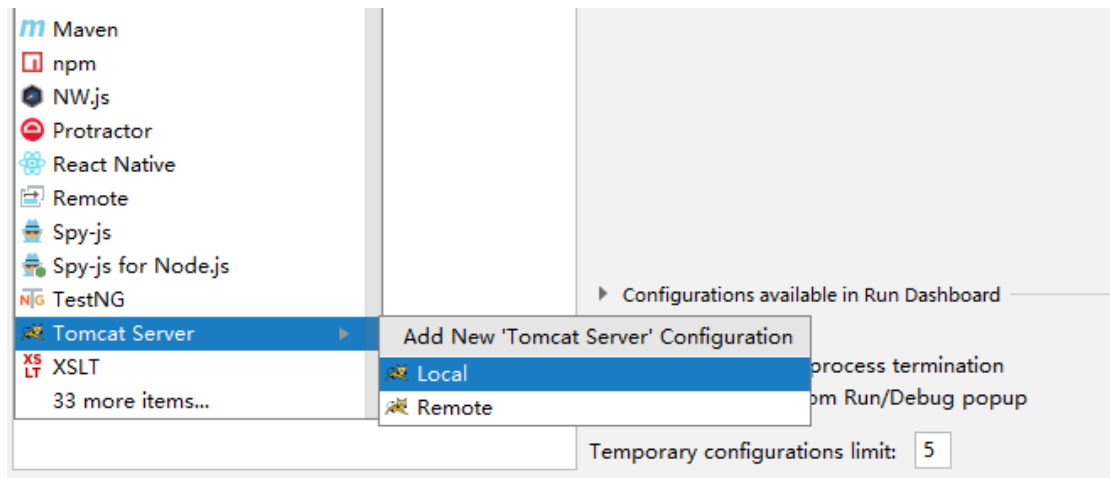
☐ Struts 2

☐ WebServices

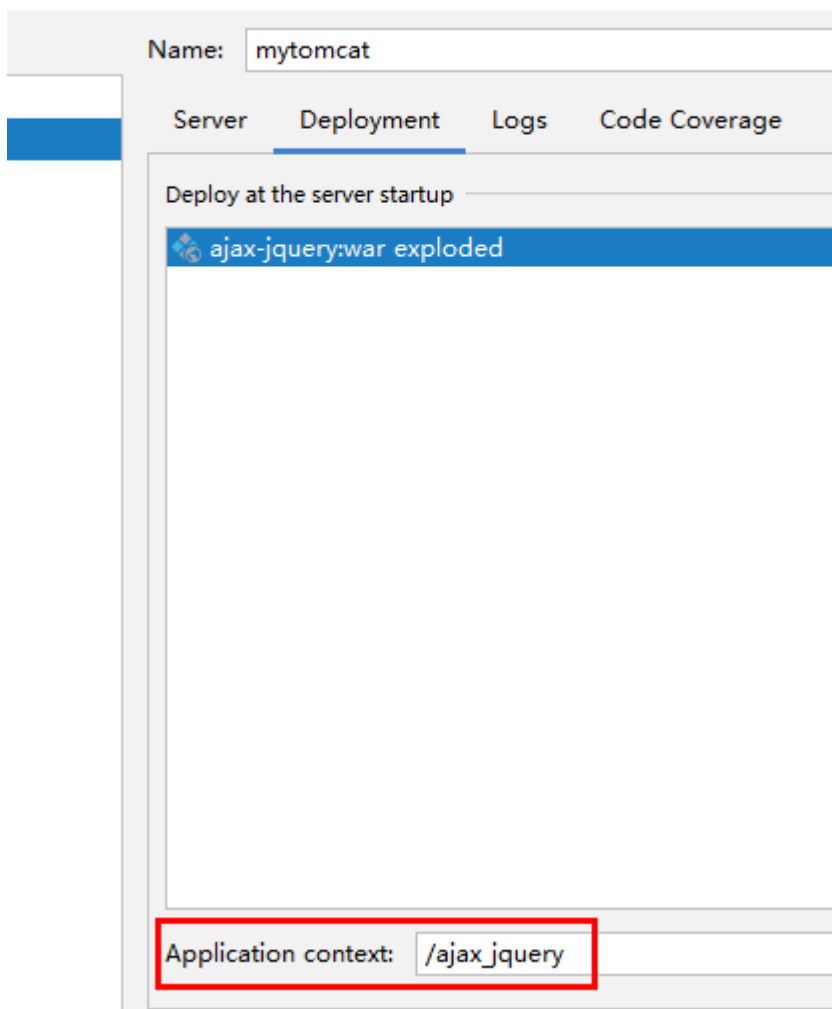
☐ CDI: Contexts and Dependency Injection

☐ Thymeleaf

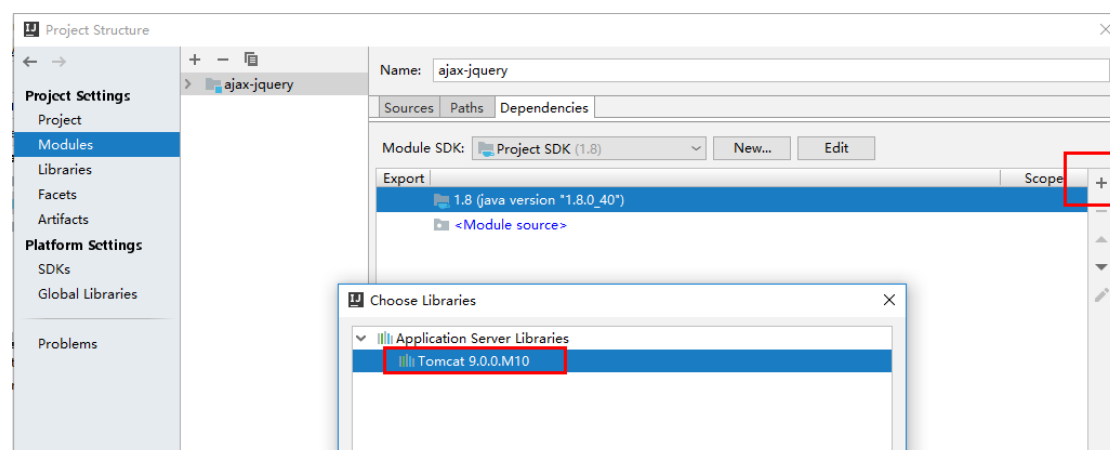
2.配置 web 运行环境 tomcat



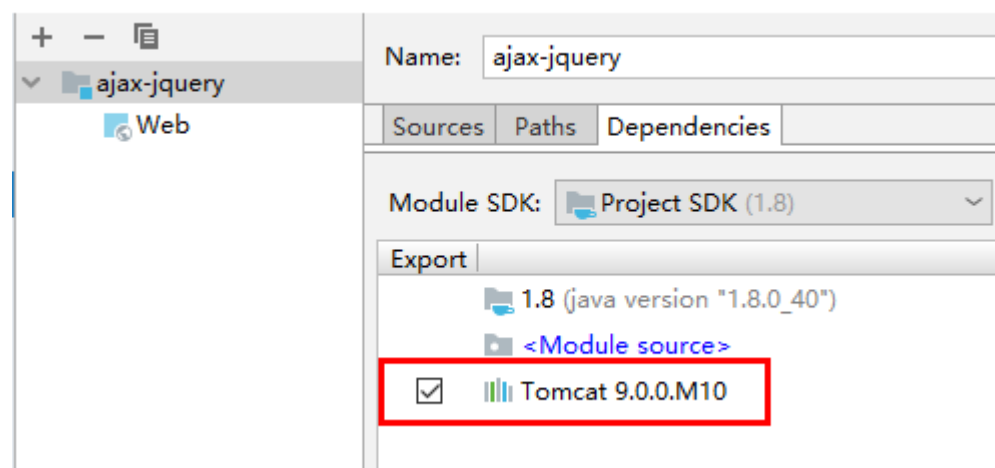
发布项目：



3.module 添加 servlet 的 jar 包







之后选中 tomcat



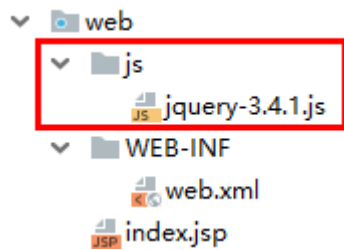
5.添加 mysql 驱动和 jackson 的 jar

在 WEB-INF 目录下创建 lib 目录，拷贝 jar 文件

-  jackson-annotations-2.9.0.jar
-  jackson-core-2.9.0.jar
-  jackson-databind-2.9.0.jar
-  mysql-connector-java-5.1.6.jar

其他处理步骤同添加 mysql 驱动。

6.创建目录 js, 拷贝 jQuery.js 文件



7.创建实体类

```
public class Province {  
    // 主键 id  
    private Integer id;  
    // 省份名称  
    private String name;  
    // 省份简称  
    private String jiancheng;  
    // 省会名称  
    private String shenghui;  
    //set | get 方法  
  
    public class City {  
        //城市id  
        private Integer id;  
        //城市名称  
        private String name;  
        //省份id  
        private Integer provinceId;  
    }  
}
```

8.创建 QueryDao 查询数据

1) 定义成员变量

```
public class QueryDao {  
  
    private Connection conn = null;  
    private PreparedStatement pst = null;  
    private ResultSet rs = null;  
    private String url="jdbc:mysql://localhost:3306/springdb";  
    private String username="root";  
    private String password="123456";  
}
```

2) 关闭资源的方法

```
private void close(Connection conn , Statement pst, ResultSet rs){
    try{
        if( rs!=null ){
            rs.close();
        }
        if( pst != null){
            pst.close();
        }
        if( conn != null){
            conn.close();
        }
    }catch (Exception ex){
        ex.printStackTrace();
    }
}
```

3) 查询所有省份名称

```
public List<Province> selectProvince(){
    List<Province> retList = new ArrayList<>();
    try {
        Class.forName("com.mysql.jdbc.Driver");
        conn = DriverManager.getConnection(url,username,password);
        String sql="select id,name,jiancheng,shenghui from province order by id";
        pst = conn.prepareStatement(sql);
        rs = pst.executeQuery();
        while (rs.next()){
            Province pro = new Province();
            pro.setId( rs.getInt(1));
            pro.setName(rs.getString(2));
            pro.setJiancheng(rs.getString("jiancheng"));
            pro.setShenghui(rs.getString("shenghui"));
            retList.add(pro);
        }
    }catch (Exception e){
        e.printStackTrace();
    } finally {
        close(conn,pst,rs);
    }
    return retList;
}
```

4) 查询省份的城市列表

```

public List<City> selectCity(Integer proId){
    List<City> retList = new ArrayList<>();
    try {
        Class.forName("com.mysql.jdbc.Driver");
        conn = DriverManager.getConnection(url,username,password);
        String sql= "select id,name,provinceId "
            + "from city where provinceId=? order by id";
        pst = conn.prepareStatement(sql);
        pst.setInt(1,proId);
        rs = pst.executeQuery();
        while (rs.next()){
            City city = new City();
            city.setId(rs.getInt("id"));
            city.setName(rs.getString("name"));
            city.setProvinceId(rs.getInt("provinceId"));
            retList.add(city);
        }
    } catch (Exception e){
        e.printStackTrace();
    } finally {
        close(conn,pst,rs);
    }
    return retList;
}

```

9.创建查询省份的 Servlet – QueryProvince

```

public class QueryProvince extends HttpServlet {
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {

        String json = "{}";
        QueryDao queryDao = new QueryDao();
        List<Province> provinces = queryDao.selectProvince();
        if(provinces != null){
            //List转为 JSONArray
            ObjectMapper mapper = new ObjectMapper();
            json = mapper.writeValueAsString(provinces);
        }
        // 设置输出内容的是json，编码是utf-8
        response.setContentType("application/json;charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println(json);
        out.flush();
        out.close();
    }
}

```


10.创建查询城市的 Servlet --- QueryCity

```
public class QueryCity extends HttpServlet {

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        String json = "{}";
        String proId = request.getParameter("proId");
        QueryDao queryDao = new QueryDao();
        List<City> cities = queryDao.selectCity(Integer.parseInt(proId));
        if(cities != null){
            //List转为 JSONArray
            ObjectMapper mapper = new ObjectMapper();
            json = mapper.writeValueAsString(cities);
        }
        //设置输出内容是是json, 编码是utf-8
        response.setContentType("application/json;charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println(json);
        out.flush();
        out.close();
    }
}
```

英

11.web.xml 注册 Servlet

```
<servlet>
    <servlet-name>queryProvince</servlet-name>
    <servlet-class>com.bjpowernode.controller.QueryProvince</servlet-class>
</servlet>
<servlet>
    <servlet-name>queryCity</servlet-name>
    <servlet-class>com.bjpowernode.controller.QueryCity</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>queryProvince</servlet-name>
    <url-pattern>/queryProvince</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>queryCity</servlet-name>
    <url-pattern>/queryCity</url-pattern>
</servlet-mapping>
```

12.修改 index.jsp

页面:

```
<div align="center">
  <table>
    <tr>
      <td>省份列表 : </td>
      <td>
        <select id="province">
          <option value="0">请选择...</option>
        </select>
      </td>
      <td> <button id="addProvince">获取省名</button></td>
    </tr>
    <tr>
      <td> 城市列表 : </td>
      <td>
        <select id="city">
          <option value="0">请选择...</option>
        </select>
      </td>
    </tr>
  </table>
</div>
```

js 函数:

```

<script type="text/javascript" src="js/jquery-3.4.1.js"></script>
<script type="text/javascript">
    $(function () {
        //页面加载完成后，单击按钮获取省份名称
        $("#addProvince").click(function(){
            $.ajax({
                url:"queryProvince",
                type:'GET',
                dataType:'json',
                success:function (result) {
                    $("#province").empty();
                    $("#province").append(
                        "<option value='0'>请选择...</option>"
                    );
                    $.each(result,function (i,n) {
                        $("#province").append(
                            "<option value='"+n.id+"'>"+n.name+"</option>"
                        );
                    })
                },
                error:function (xhr, status, error) {
                    alert("请求错误："+error)
                }
            })
        })
    })

```

```

//选择省份,查询城市列表
$("#province").on("change",function (){
    var proId = $("#province > option:selected").val();
    if( proId == "0"){
        alert("请选择一个有效的省份")
    } else {
        $.post("queryCity",{proId:proId},callback,"json");
    }
})

function callback(result) {
    $("#city").empty();
    $("#city").append(
        "<option value='0'>请选择...</option>"
    );
    $.each(result,function (i,n) {
        $("#city").append(
            "<option value='"+n.id+"'>"+n.name+"</option>"
        );
    })
}
})
</script>

```