JSON、AJAX、i18n

- JSON、AJAX、i18n
  - JSON
    - 什么是JSON?
    - JSON在JavaScript中的使用
      - JSON的定义
      - JSON定义示例
      - JSON的访问
      - JSON的两个常用方法
    - JSON在Java中的使用
      - JavaBean和JSON的互转
      - List和JSON的互转
      - Map和JSON的互转
  - AJAX请求
    - 什么是AJAX请求
    - 原生JavaScript的AJAX请求示例
    - AJAX请求的特点说明
      - 同步请求与异步请求的区别
    - jQuery 中的 AJAX 请求
      - jQuery的ajax方法
      - jQuery的get和post方法
      - jQuery的getJSON方法
      - jQuery的serialize方法
  - i18n
    - 什么是i18n国际化
    - i18n国际化三要素介绍
    - i18n国际化基础示例
    - 通过请求头实现国际化
    - 通过语言类型选择实现国际化
    - 使用JSTL标签库fmt实现国际化

JSON

什么是JSON?

> *JSON (JavaScript Object Notation) 是一种轻量级的数据交换格式。易于人阅读和编写。同时也易于机器解析和生成。JSON 采用完全独立于语言的文本格式，而且很多语言都提供了对json 的支持（包括C, C++, C#, Java, JavaScript, Perl, Python 等）。这样就使得JSON 成为理想的数据交换格式*

json 是一种轻量级的数据交换格式。轻量级指的是跟 xml 做比较。

数据交换指的是**客户端**和**服务器**之间业务数据的传递格式。

JSON在JavaScript中的使用

在客户端使用JSON --- JavaScript

JSON的定义

> *JSON是由键值对组成，并且由花括号（大括号）包围。每个键由引号引起来，键和值之间使用冒号进行分隔， 多组键值对之间进行逗号进行分隔。*

JSON定义示例

```
var jsonObj = {
    "key1":12, //number类型
    "key2":"abc", //String
    "key3":true, //boolean
    "key4":[11,"arr",false], //数组
    "key5":{ //JSON类型
        "key5_1" : 551,
        "key5_2" : "key5_2_value"
    },
    "key6":[{
        "key6_1_1":6611,
        "key6_1_2":"key6_1_2_value"
    },{
        "key6_2_1":6621,
        "key6_2_2":"key6_2_2_value"
    }]
};
```

JSON的访问

JSON本身是一个对象。

JSON中的 key 我们可以理解为是对象中的一个属性。 JSON 中的 key 访问就跟访问对象的属性一样：JSON对象.key

JSON访问示例：

```
alert(typeof(jsonObj));// object json 就是一个对象
alert(jsonObj.key1); //12
alert(jsonObj.key2); // abc
alert(jsonObj.key3); // true
alert(jsonObj.key4);// 得到数组[11,"arr",false]
// json 中 数组值的遍历
for(var i = 0; i < jsonObj.key4.length; i++) {
    alert(jsonObj.key4[i]);
}
alert(jsonObj.key5.key5_1);//551
alert(jsonObj.key5.key5_2);//key5_2_value
alert( jsonObj.key6 );// 得到 json 数组
// 取出来每一个元素都是 json 对象
var jsonItem = jsonObj.key6[0];
// alert( jsonItem.key6_1_1 ); //6611
alert( jsonItem.key6_1_2 ); //key6_1_2_value
```

JSON的两个常用方法

json的存在有两种形式

1. 对象的形式存在，我们叫它 json 对象
2. 字符串的形式存在，我们叫它 json 字符串

一般我们要操作 json 中的数据的时候，需要 json 对象的格式。

一般我们要在客户端和服务器之间进行数据交换的时候，使用 json 字符串

```
JSON.stringify()
    //把 json 对象转换成为 json 字符串

JSON.parse()
    //把 json 字符串转换成为 json 对象
```

示例代码：

```
// 把 json 对象转换成为 json 字符串
var jsonObjString = JSON.stringify(jsonObj); // 特别像 Java 中对象的 toString
alert(jsonObjString)
// 把 json 字符串。转换成为 json 对象
var jsonObj2 = JSON.parse(jsonObjString);
alert(jsonObj2.key1);// 12
alert(jsonObj2.key2);// abc
```

JSON在Java中的使用

在服务器端使用---Java

要导入jar包 / gson-2.2.4.jar

JavaBean和JSON的互转

```
//前置 -- /pojo/Person 构造器、set&get、toString
public void test1(){
    Person person = new Person(1, "zjx");
    //创建Gson对象实例
    Gson gson = new GSon();
    //toJson方法可以把Java对象转换成为JSON字符串
    String personJsonString = gson.toJson(person);
    System.out.println("personJsonString");
    //fromJson把json字符串转换回Java对象
    //第一个参数是JSON字符串
    //第二个参数是转换回去的Java对象类型
    gson.fromJson(personJsonString, Person.class);
}
```

List和JSON的互转

```
public class PersonListType extends TypeToken<ArrayList<Person>>{

}
```

```
public void test2(){
    List<Person> personList = new ArrayList<>();

    personList.add(new Person(1, "xiaoming"));
    personList.add(new Person(2, "xiaohong"));

    Gson gson = new Gson();

    //把List转换为JSON字符串
    String personListJsonString = gson.toJson(personList);
    System.out.println(personListJsonString);

    List<Person> list = gson.fromJson(personListJsonString, new PersonListType().getType());

    System.out.println(list);
    Person person = list.get(0);
    System.out.println(person);
}
```

Map和JSON的互转

```
public class PersonMaptype extends TypeToken<HashMap<Integer, Person>>{
    //TypeToken<> --- <>内为转回去的具体的类型
}
```

```
public void test3(){
    Map<Integer,Person> personMap = new HashMap<>();
    personMap.put(1, new Person(1, "xiaoming"));
    personMap.put(2, new Person(2, "xiaohong"));
```

```
    Gson gson = new Gson();
    // 把 map 集合转换成为 json 字符串
    String personMapJsonString = gson.toJson(personMap);
    System.out.println(personMapJsonString);
    // Map<Integer,Person> personMap2 =
gson.fromJson(personMapJsonString, new
PersonMapType().getType());

    //使用匿名内部类的方式
    Map<Integer,Person> personMap2 =
gson.fromJson(personMapJsonString, new
TypeToken<HashMap<Integer,Person>>(){}.getType());
    System.out.println(personMap2);
    Person p = personMap2.get(1);
    System.out.println(p);
}
```

```
public void test3(){
    Map<Integer, Person> personMap = new HashMap<>();
}
```

AJAX请求

AJAX 教程 | 菜鸟教程 (runoob.com)

什么是AJAX请求

> *AJAX即"Asynchronous Javascript And XML"（异步JavaScript和XML），是指一种创建交互式网页应用的网页开发技术。*

AJAX是一种浏览器通过JavaScript异步发起请求。局部更新页面的技术。

Ajax 请求的局部更新，浏览器地址栏不会发生变化。局部更新不会舍弃原来页面的内容

原生JavaScript的AJAX请求示例

Java服务器端代码示例：

在Servlet中解决中文乱码问题：

`resp.setContentType("text/html; charset="UTF-8");`

```
public class AjaxServlet extends BaseServlet{
    //BaseServlet 为基本Servlet方法
    protected void javaScriptAjax(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException{
        System.out.println("Ajax请求过来了");

        Person person = new Person(1, "xiaoming");

        //json格式字符串，用于客户端和服务器之间进行数据交换
        Gson gson = new Gson();
        String personJsonString = gson.toJson(person);
        resp.getWriter().writer(personJsonString);
    }
}
```

JavaScript客户端接收代码示例：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="pragma" content="no-cache" />
        <meta http-equiv="cache-control" content="no-cache" />
        <meta http-equiv="Expires" content="0" />
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Insert title here</title>
            <script type="text/javascript">
            // 在这里使用 javaScript 语言发起 Ajax 请求，访问服务器 AjaxServlet 中 javaScriptAjax
            function ajaxRequest() {
```

```
12                  // 1、我们首先要创建 XMLHttpRequest
13                  var xmlhttprequest = new XMLHttpRequest();
14                  // 2、调用 open 方法设置请求参数
15                  //如需将请求发送到服务器，使用XMLHttpRequest 对象的
    open()和send()方法：
16
    xmlhttprequest.open("GET","http://localhost:8080/16_json_ajax_i1
    8n/ajaxServlet?action=javaScriptAjax",true);
17                  // 4、在 send() 方法前绑定 onreadystatechange 事件，
    处理请求完成后的操作。
18                  //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。readyState
    属性存有 XMLHttpRequest 的状态信息。详情见教程
19                  xmlhttprequest.onreadystatechange = function(){
20                  if (xmlhttprequest.readyState == 4 &&
    xmlhttprequest.status == 200) {
21                      //将页面显示数据转换为JSON数据
22                      var jsonObj =
    JSON.parse(xmlhttprequest.responseText);
23                      // 把响应的数据显示在页面上
24                      document.getElementById("div01").innerHTML =
    "编号：" + jsonObj.id + " ，姓名：" + jsonObj.name;
25                  }
26              }
27              // 3、调用 send 方法发送请求
28              xmlhttprequest.send();
29              }
30          </script>
31      </head>
32      <body>
33          <button onclick="ajaxRequest()">ajax request</button>
34          <div id="div01">
35          </div>
36      </body>
37  </html>
38
```

AJAX请求的特点说明

同步请求与异步请求的区别

> *同步，可以理解为在执行完一个函数或方法之后，一直等待系统返回值或消息，这时程序是出于阻塞的，只有接收到返回的值或消息后才往下执行其他的命令。*

> *异步，执行完函数或方法后，不必阻塞性地等待返回值或消息，只需要向系统委托一个异步过程，那么当系统接收到返回值或消息时，系统会自动触发委托的异步过程，从而完成一个完整的流程。*

jQuery 中的 AJAX 请求

jQuery的ajax方法

$.ajax方法

url	表示请求的地址
type	表示请求的类型 GET 或 POST 请求 请求的格式有两种:1.name=value&name=value。2.{key:value}
date	表示发送给服务器的数据。
success	请求成功，响应的回调函数
dateType	响应的数据类型 常用的数据类型有: text 表示纯文本 xml 表示xml数据 json 表示json对象

服务器代码示例：

```java
public class AjaxServlet extends BaseServlet{
    //BaseServlet 为基本Servlet方法
    protected void javaScriptAjax(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException{
        System.out.println("jQueryAjax方法调用");

        Person person = new Person(1, "xiaoming");

        //json格式字符串，用于客户端和服务器之间进行数据交换
        Gson gson = new Gson();
        String personJsonString = gson.toJson(person);
        resp.getWriter().writer(personJsonString);
    }
}
```

客户端代码示例：

```javascript
$("#ajaxBtn").click(function(){
    $.ajax({

 url:"http://localhost:8080/16_json_ajax_i18n/ajaxServlet",
        //data:"action=jQueryAjax",
        data:{action:"jQueryAjax"},
        type:"GET",
        success:function (data) {
            //alert("服务器返回的数据是：" + data);
            //var jsonObj = JSON.parse(data);
            $("#msg").html("编号：" + data.id + " , 姓名：" +
data.name);
        },
        dataType : "json"
    });
});
```

jQuery的get和post方法

url	请求的 url 地址
date	发送的数据
callback	成功的回调函数
type	返回的数据类型

```
// ajax--get 请求
$("#getBtn").click(function(){
    //$.get(url, date, callback, type)

$.get("http://localhost:8080/16_json_ajax_i18n/ajaxServlet","action=jQueryGet",function (data) {
        $("#msg").html(" get 编号: " + data.id + " , 姓名: " + data.name);
    },"json");
});
    // ajax--post 请求
$("#postBtn").click(function(){

 $.post("http://localhost:8080/16_json_ajax_i18n/ajaxServlet","action=jQueryPost",function (data){
        $("#msg").html(" post 编号: " + data.id + " , 姓名: " + data.name);
    },"json");
});
```

jQuery的getJSON方法

url	请求的url地址
date	发送给服务器的数据
callback	成功的回调函数

```
// ajax--getJson 请求
$("#getJSONBtn").click(function(){

 $.getJSON("http://localhost:8080/16_json_ajax_i18n/ajaxServlet","action=jQueryGetJSON",function(data) {
        $("#msg").html(" getJSON 编号: " + data.id + " , 姓名: " + data.name);
    });
});
```

jQuery的serialize方法

serialize()可以把表单中所有表单项的内容都获取到，并以 name=value&name=value 的形式进行拼接。

该方法在获取所有表单项提交时，不会是整个页面刷新或者改变地址栏。

```
// ajax 请求
$("#submit").click(function(){
// 把参数序列化

 $.getJSON("http://localhost:8080/16_json_ajax_i18n/ajaxServlet","action=jQuerySerialize&" +
    $("#form01").serialize(),function (data) {
        $("#msg").html(" Serialize 编号: " + data.id + " , 姓名: " + data.name);
    });
});
```

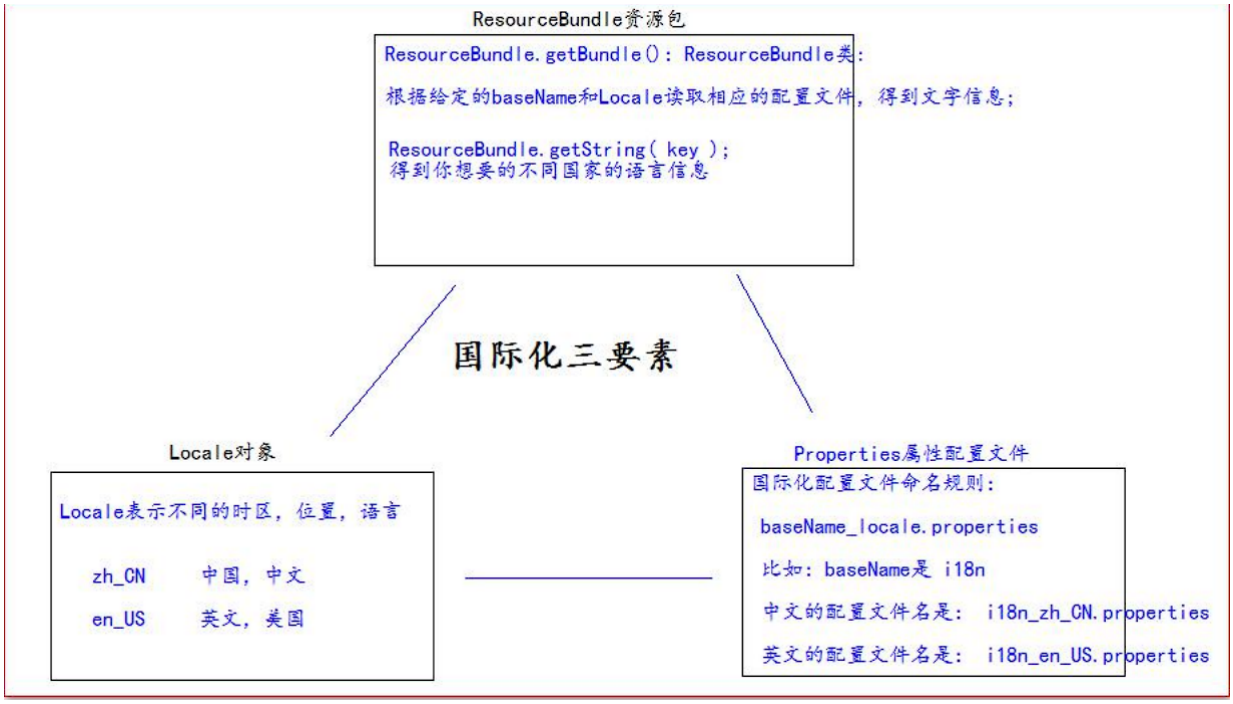
i18n

什么是i18n国际化

- 国际化（Internationalization）指的是同一个网站可以支持多种不同的语言，以方便不同国家，不同语种的用户访问。

- 关于国际化我们想到的最简单的方案就是为不同的国家创建不同的网站，比如苹果公司，他的英文官网是： http://www.apple.com 而中国官网是 http://www.apple.com/cn。
- 苹果公司这种方案并不适合全部公司，而我们希望相同的一个网站，而不同人访问的时候可以根据用户所在的区域显示 不同的语言文字，而网站的布局样式等不发生改变。
- 于是就有了我们说的国际化，国际化总的来说就是同一个网站不同国家的人来访问可以显示出不同的语言。但实际上这 种需求并不强烈，一般真的有国际化需求的公司，主流采用的依然是苹果公司的那种方案，为不同的国家创建不同的页 面。所以国际化的内容我们了解一下即可。
- 国际化的英文 Internationalization，但是由于拼写过长，简单的写法叫做i18n，代表的是 Internationalization 这个单词，以i开头，以n结尾，而中间是 18 个字母，所以简写为i18n。以后我们说i18n和国际化是一个意思。

i18n国际化三要素介绍



i18n国际化基础示例

国家化资源properties测试，如下：

i18n_en_US.properties

```
username=username
password=password
sex=sex
age=age
```

i18n_zh_CN.properties

```
username=用户名
password=密码
sex=性别
age=年龄
```

```
public class i18nTest{
     @Test
    public void testLocale(){
        //获取系统默认语言及国家信息
        Locale locale = Locale.getDefault();
        System.out.println(LOcale);

        //获取中文 / 美国 的常量的Locale对象
        System.out.println(Locale.CHINA);
        System.out.println(Locale.US);
    }

    @Test
    public void testI18n(){
        // 得到我们需要的 Locale 对象
        Locale locale = Locale.CHINA;
        // 通过指定的 basename 和 Locale 对象，读取 相应的配置文件
        ResourceBundle bundle = ResourceBundle.getBundle("i18n",
locale);

        System.out.println("username：" +
bundle.getString("username"));
        System.out.println("password：" +
bundle.getString("password"));
        System.out.println("Sex：" + bundle.getString("sex"));
        System.out.println("age：" + bundle.getString("age"));
```

```
    }
}
```

通过请求头实现国际化

浏览器F12 / 设置可查看和设置请求头的语言优先级

```
<%@ page import="java.util.Locale" %>
<%@ page import="java.util.ResourceBundle" %>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="pragma" content="no-cache" />
        <meta http-equiv="cache-control" content="no-cache" />
        <meta http-equiv="Expires" content="0" />
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>Insert title here</title>
    </head>
    <body>
        <%
            // 从请求头中获取 Locale 信息（语言）
            Locale locale = request.getLocale();
            System.out.println(locale);
            // 获取读取包（根据 指定的 baseName 和 Locale 读取 语言信
息）
            ResourceBundle i18n =
ResourceBundle.getBundle("i18n", locale);
        %>
        <a href="">中文</a>|
        <a href="">english</a>
        <center>
            <h1><%=i18n.getString("regist")%></h1>
            <table>
                <form>
```

```
                    <tr>
                        <td><%=i18n.getString("username")%></td>
                        <td><input name="username" type="text"
/></td>
                    </tr>
                    <tr>
                        <td><%=i18n.getString("password")%></td>
                        <td><input type="password" /></td>
                    </tr>
                    <tr>
                        <td><%=i18n.getString("sex")%></td>
                        <td>
                            <input type="radio" />
<%=i18n.getString("boy")%>
                            <input type="radio" />
<%=i18n.getString("girl")%>
                        </td>
                    </tr>
                    <tr>
                        <td><%=i18n.getString("email")%></td>
                        <td><input type="text" /></td>
                    </tr>
                    <tr>
                        <td colspan="2" align="center">
                            <input type="reset" value="
<%=i18n.getString("reset")%>" />  
                            <input type="submit" value="
<%=i18n.getString("submit")%>" />
                        </td>
                    </tr>
                </form>
            </table>
            <br/> <br/> <br/> <br/>
        </center>
        国际化测试:
        <br /> 1、访问页面，通过浏览器设置，请求头信息确定国际化语言。
        <br /> 2、通过左上角，手动切换语言
    </body>
</html>
```

通过语言类型选择实现国际化

```
<body>
<%
    //根据上面的代码修改部分可得
    // 从请求头中获取 Locale 信息（语言）
    Locale locale = null;
    String country = request.getParameter("country");
    if ("cn".equals(country)) {
        locale = Locale.CHINA;
    } else if ("usa".equals(country)) {
        locale = Locale.US;
    } else {
        locale = request.getLocale();
    }
    System.out.println(locale);
    // 获取读取包（根据 指定的 baseName 和 Locale 读取 语言信息）
    ResourceBundle i18n = ResourceBundle.getBundle("i18n",
locale);
%>
<a href="i18n.jsp?country=cn">中文</a>|
<a href="i18n.jsp?country=usa">english</a>
```

使用JSTL标签库fmt实现国际化

```
<%--1 使用标签设置 Locale 信息--%>
<fmt:setLocale value="" />
<%--2 使用标签设置 baseName--%>
<fmt:setBundle basename=""/>
<%--3 输出指定 key 的国际化信息--%>
<fmt:message key="" />
```

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"
%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="pragma" content="no-cache" />
        <meta http-equiv="cache-control" content="no-cache" />
        <meta http-equiv="Expires" content="0" />
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Insert title here</title>
    </head>
    <body>
    <%--1 使用标签设置 Locale 信息--%>
    <fmt:setLocale value="${param.locale}" />
    <%--2 使用标签设置 baseName--%>
    <fmt:setBundle basename="i18n"/>
    <a href="i18n_fmt.jsp?locale=zh_CN">中文</a>|
    <a href="i18n_fmt.jsp?locale=en_US">english</a>
    <center>
        <h1><fmt:message key="regist" /></h1>
        <table>
            <form>
                <tr>
                    <td><fmt:message key="username" /></td>
                    <td><input name="username" type="text" /></td>
                </tr>
                <tr>
                    <td><fmt:message key="password" /></td>
                    <td><input type="password" /></td>
                </tr>
                <tr>
                    <td><fmt:message key="sex" /></td>
                    <td>
                        <input type="radio" /><fmt:message key="boy" />
                        <input type="radio" /><fmt:message key="girl" />
                    </td>
```

```
38                  </tr>
39                  <tr>
40                      <td><fmt:message key="email" /></td>
41                      <td><input type="text" /></td>
42                  </tr>
43                  <tr>
44                      <td colspan="2" align="center">
45                          <input type="reset" value="<fmt:message
    key="reset" />" />  
46                          <input type="submit" value="<fmt:message
    key="submit" />" />
47                      </td>
48                  </tr>
49
```

返回文首