

Ajax

1.AJAX介绍

2.AJAX实现

2.1 get提交

2.2 post提交

3.JSON

3.1 什么是JSON

JSON ； JavaScript Object Notation

3.2 使用JS实现JSON操作

在JS中的JSON操作，基本上都依靠于JS的原生操作。

JSON.parse(), 将JSON字符串转换为JS对象

JSON.stringify(),将JS对象转换为string字符串

3.2.1 将JSON字符串转换为JS对象

使用JSON.parse()方法能够将一个JSON格式的字符串转换为JS对象

```
<script language="javascript">
  var jsonStr = '{"name":"sanfeng.zhang","age":"18","gender":"Male"}';
  var jsonObj = JSON.parse(jsonStr);
  console.log(jsonObj.name);
  console.log(jsonObj.age);
  console.log(jsonObj.gender);
</script>
```

3.2.2 将JS对象转换为JSON字符串

使用JSON.stringify()方法能够将一个JS对象转换为JS的字符串。

```
<script language="javascript">
    var obj = {
        name : "sanfeng.zhang",
        age : 12 ,
        gender : "male"
    }

    var jsonStr = JSON.stringify(obj);
    console.log(jsonStr);
</script>
```

输出结果：{"name":"sanfeng.zhang","age":12,"gender":"male"}

3.3 使用Java实现JSON操作

在Java中实现JSON的操作有很多中，比较常用且功能比较完善的是Jackson提供的JSON操作。

使用Jackson提供的JSON操作需要将包下载到本地并在classpath中导入

包名：Jackson-databind, Jackson-annotations, Jackson-core

3.3.1 将JSON字符串转换为Java对象

语法：使用ObjectMapper类型提供的 readValue(String value,Class classType)方法

注意：目标类中必须要提供无参的构造函数，否则会报异常。

```
package com.zx.main;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.ObjectMapper;

public class Run {

    public static void main(String[] args) throws JsonMappingException,
    JsonProcessingException {
        // TODO Auto-generated method stub
        ObjectMapper mapper = new ObjectMapper();
        String jsonStr = "{\"id\":1,\"name\":\"sanfeng.zhang\",\"age\":18}";
        Person person = mapper.readValue(jsonStr, Person.class);
        System.out.println("name : " + person.getName());
        System.out.println("age : " + person.getAge());
        System.out.println("id : " + person.getId());
    }

}

class Person {
    private int id;
    private String name;
    private int age;
}
```

```

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}
}

```

3.3.2 将Java对象转换为JSON字符串

语法：使用ObjectMapper类型提供的writeValueAsString()方法将一个对象转换为符合JSON格式的字符串。

特点：只会将提供了getter方法的属性写进JSON字符串

```

package com.zx.main;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.ObjectMapper;

public class Run {

    public static void main(String[] args) throws JsonMappingException,
    JsonProcessingException {
        // TODO Auto-generated method stub
        Person person = new Person();
        person.setId(1);
        person.setName("Test");
        person.setAge(18);

        ObjectMapper mapper = new ObjectMapper();
        String jsonStr = mapper.writeValueAsString(person);
    }
}

```

```

        System.out.println(jsonStr);

    }

}

class Person {
    private int id;
    private String name;
    private int age;
    private String gender;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }
}

```

4.jQuery实现Ajax

其实在ajax中我们需要关注的点和普通发送请求是一样的。通常只想关注要用什么样的方式，将哪些数据发送到哪里去。并且会接受什么样的结果回来。所以依靠原生的JS去实现AJAX的话实际上有一大部分的内容是在做它的基础设置，在jq中，已经帮我们实现了AJAX的封装，像那些设置级别的东西基本上都被封装好了，我们只需要将重要的信息实现就可以了。

4.1 \$.get

简易版的实现get请求操作

语法：\$.get(url, data, success, dataType)

```
$.get('https://api.example.com/users', { id: 1 }, function(response) {  
    console.log(response);  
}, 'json');
```

4.2 \$.post

简易版的实现post请求

语法：\$.post(url, data, success, dataType)

```
$.post('https://api.example.com/register', {  
    username: 'newUser',  
    email: 'example@example.com'  
}, function(response) {  
    console.log('註冊成功', response);  
}, 'json');
```

4.3 \$.ajax

可以更具自己的需求灵活定制操作。

语法：

\$.ajax({ url: '<https://api.example.com/data>', // 請求地址 type: 'GET', // 請求方法 (GET, POST, PUT, DELETE) dataType: 'json', // 期望返回的数据格式 (json, text, html, javascript等)

contentType: "application/json" //发送给后端数据的格式 (application/json , application/x-www-form-urlencoded , multipart/form-data)

data: //发送到服务器端的数据

success: function(response) { // 成功回調 console.log(response); }, error: function(xhr, status, error) { // 錯誤回調 console.error(error); } });