



深圳大学
Shenzhen University

第2章

中间件通用技术原理与开发

中间件技术

Middleware Technology

深圳大学计算机与软件学院



主要内容

- ◆ 2.1 互操作技术
- ◆ 2.2 基于XML的异构数据存储、交换与传输技术
- ◆ 2.3 JSON技术



JSON技术

- ◆ **2.3.1JSON**

- ◆ **2.3.2JSON语法**

- ◆ **2.3.3使用JSON实现数据交换**



2.3.1 JSON技术

- JSON JavaScript 对象表示法 (JavaScript Object Notation) , 是一种主流的数据交换格式。
- 用于数据交互服务器返回给客户端的数据, 一般都是JSON格式或者XML格式 (文件下载除外) 。
- JSON 是存储和交换文本信息的语法, 类似 XML。
- 简单: 只有六种类型的符号 [] { } ; , 就可以表示任何种类的复杂对象或数组
- 易读: 机器解析还是人的阅读都能够从这种格式中快速找到所需要的信息



JSON特点

- **JSON 是一种轻量级的数据交换格式：**与XML这种数据交换格式相比，速度更快，文档也更小。XML文件中会包含很多对象的层级关系，解析及转换极不方便，字符更多导致文件过大。而这些弊端JSON都已经克服，简单的符号使得解析更快。
- **JSON是一种与平台无关的数据交换：**JSON作为数据交换的格式与平台种类是无关的。不管服务器使用Java语言、C#语言或其他语言都能够从这种字符串格式转换为具体语言的对象。同样，任何语言也都能将自己的对象转换为JSON的字符串。



2.3.2JSON语法

- JSON 的结构：使用JSON进行数据交换时，主要有两种结构
 - (1)使用“名称-值”对儿的形式来表示一个对象
 - (2)值的有序列表，也就是数组
- 在这两种结构基础之上，相互之间可以进行嵌套。即对象的值可以是数组或对象，数组中的某个值可以是对象或数组。
- 不同语言之中的对象、记录、结构、字典、哈希表等都可以作为转换的对象基础结构。

使用JSON表示一个对象

- 使用JSON表示一个对象时，基本的语法结构是 以大括号开始，以大括号结尾。大括号之间的各个属性由逗号分隔，每一组属性由属性名和属性值的成对组合实现。

例如：要表现一个**雇员**对象可以按如下结构书写：
`{'id':1,'name':'王小贱', 'age':24}`

- 不管多么复杂的结构，只要看到大括号就代表这个一个对象，至于内部的属性只要保证用逗号分隔即可。
- 每一组属性在书写时要遵循以下原则：
 - 属性名要使用引号引起来
 - 属性值的类型可以多样化，包括string、number、true、false、null、object、array
 - 如果属性值的类型为字符串，则必须使用引号括起来

访问对象的属性

- 访问某个属性值语法格式：**对象 . 属性名**

例如访问**雇员**对象，获取**name**属性的值“王小贱” 使用代码：

```
var obj = { 'id':1, 'name': '王小贱', 'age':24 };  
  
var name = obj . name ;
```


对象中包含对象的结构及属性访问

- 如果对象的某个属性值依然是对象，那么定义结构如下：

```
var obj = { 'id':1 , 'name':'王小贱' , 'dep':{'depId':1,'depName':'财务部' }};  
  
var depName = obj.dep.depName ;
```

- 对象属性可以是对象，访问这个对象属性的时候依然遵循“对象. 属性名”的方式来获取要想访问属性，使用多级的点格式
- 上面的对象中obj就是一个对象中还包含对象的结构，**dep是属性名**，{ 'depId':1, 'depName':'财务部' }是属性值，即部门属性又包括部门编号、部门名称这两个属性

使用JSON表示一个数组

- 使用JSON表示数组一定以方括号开始，以方括号结尾，数组内的多个值之间用逗号隔开，数组内的多个值之间用逗号隔开，语法如下：value的类型可以是string、number、false、true、null、object、array。


[value, value, value, ...]

- 例如表示3个员工的数组为：

```
var jsonArr = [{ 'id':1 , 'name':'王小贱' } , { 'id':2,'name':'黄小仙' } , { 'id':3,'name':'王老头' } ] ;
```

- 如果要访问数组中的第2个元素的姓名，代码为：

```
var name = jsonArr[1].name;
```

- 
- 数组中的元素可以嵌套使用数组，对象的属性也可以嵌套数组，代码例如：

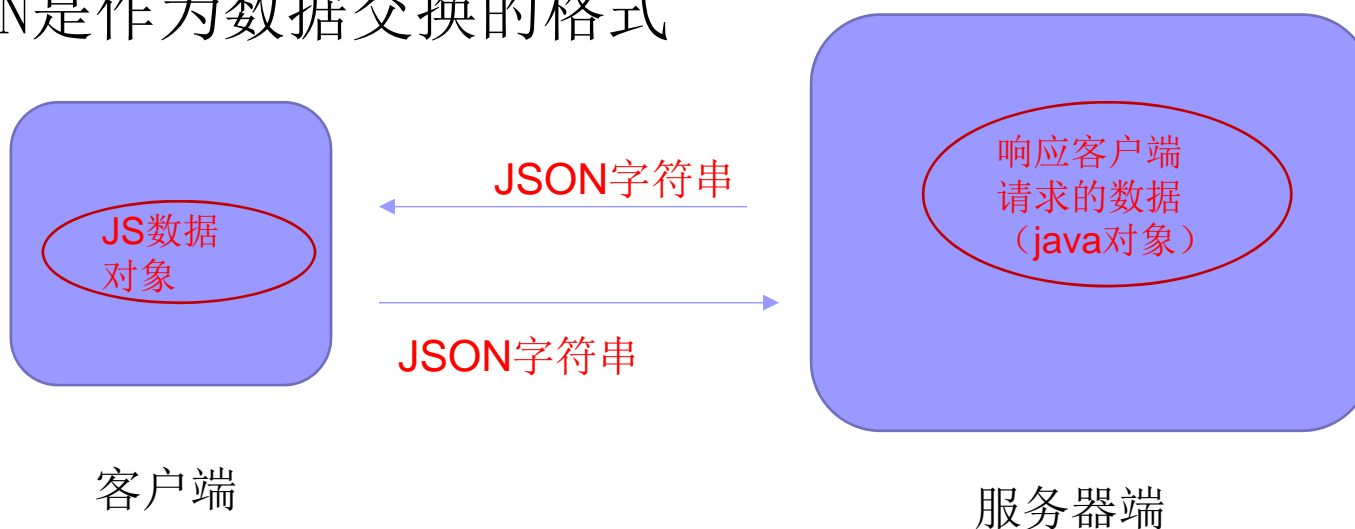
```
var jsonObj = { 'id':1 , 'name':'王小贱' , 'salary':[{'month':1 , 'number':5000},{ 'month':2 , 'number':4000 }]};
```

- 对于嵌套的数据类型，遵循原则就是：
- 是对象就通过 “.” 来访问，
- 是数组就通过 “[] ” 下标来访问。
- 针对上面的对象，要想知道员工2月份的工资是多少，访问形式如下：

```
var number = jsonObj.salary[1].number;
```

2.3.3使用JSON实现数据交换

- 使用JSON数据交换原理
- JSON是作为数据交换的格式



(1)S→C:如果是客户端请求数据，服务器端就将Java对象先转换成JSON字符串，经响应把字符串传到客户端之后，客户端就会接收到这个转换结果，但JavaScript要求把这个字符串变成对象格式才更方便访问，在客户端的JavaScript代码中又需要将这个JSON字符串变成JavaScript能够识别的对象，这样就完成了从服务器端的对象到客户端对象的整个转变过程。

(2)C→S:如果在客户端填写了数据后想提交给服务器，首先是将客户端数据构造成一个JSON对象字符串，经网络传递到服务器端，服务器端再依据转换规则将JSON字符串变成Java识别的对象。

Java对象转换成JSON字符串

- 在服务器端对象的转换过程可以使用官方提供的API
- JSONObject和JSONArray分别为对象和数组的转换类型，
- 使用这两个类型进行对象转换时代码如下：

```
Employee emp = new Employee(1, "王小贱", "男");

JSONObject obj = JSONObject.fromObject(emp);

String jsonStr = obj.toString();
```

- 数组的转换代码为：

```
List<Employee> emps = new ArrayList< Employee >();

for(int i=0;i<3;i++){

    Employee s = new Employee ();

    s.setId(i+1);

    s.setName("Bear" + i);

    s.setGender("男");

    emps.add(s);

}

JSONArray jsonArr = JSONArray.fromObject(emps);

String jsonStr = jsonArr.toString();
```

JSON字符串转换成JS对象

- 当JSON字符串经传输到达客户端时，需要完成JSON到JavaScript对象的转换，如果使用JS原生的方法进行转换的话，可以使用eval（）方法，但需要在JSON的前后连接上左右圆括号，如：

```
var jsonObj = eval( "( " + json " )" );
```

- 为什么要 eval这里要添加('(' + json + ')') 呢？
- 原因在于eval本身。 由于json是以”{}”的方式来开始以及结束的，在JS中，它会被当成一个语句块来处理，所以必须强制性的将它转换成一种表达式。加上圆括号的目的是迫使eval函数在处理JavaScript代码的时候强制将括号内的表达式（expression）转化为对象，而不是作为语句（statement）来执行。
- 注意：方式eval（）方法是动态执行其中字符串（可能是js脚本）的，这样很容易会造成系统的安全问题。
- 可以采用一些规避了 eval（）的第三方客户端脚本库。

使用第三方客户端脚本库

- 采用一些规避了 `eval()` 的第三方客户端脚本库，如 `prototype.js` 里面提供的 `evalJSON()` 方法。通过在页面中引入 `prototype.js` 文件后就可以使用这个扩展库。

- 代码例：

```
<script type="text/javascript" src="js/prototype-1.6.0.3.js"></script>

<script type="text/javascript">

    function f1(){

        var xhr = getXhr();

        xhr.open('get','quoto.do',true);

        xhr.onreadystatechange=function(){

            if(xhr.readyState == 4){

                var txt = xhr.responseText;

                //将json字符串转换成javascript对象或者数组

                var arr = txt.evalJSON();

            }

        };

        xhr.send(null);

    }

</script>
```

缓存问题

- IE浏览器提供的AJAX对象，在发送GET请求时，会先查看是否访问过该地址，如果改地址已经访问过，那么浏览器不再发送请求，表现在：页面终究是第一次点击某功能会得到数据，但是如果多次反复点击想获取最新数据时页面不会有任何变化，因浏览器发现地址相同而拒绝发出请求。
- 这种页面表现只出现在IE浏览器中，Chrom浏览器和Firefox浏览器都能够实现数据的获取和页面的刷新。
- **解决问题：** 欺骗浏览器，让它认为每次请求的地址是不一样的。构建不同的URL需要添加上一个随机数。代码：

```
xhr.open('get','getNumber.do?' + Math.random() , true );
```

- Math.random（）可以产生0-1之间的随机数，这样每次都可以生成不一样的地址，让浏览器重新发送请求，以达到数据刷新的效果。
- 第二种方式：不使用GET请求方式，改成POST提交，浏览器不会进行地址的判断。

同步问题

- 同步请求指的是在AJAX对象发出请求后，浏览器要等待响应的回来之后再允许页面继续操作，表现形式就是页面的假死状态。好像是只有做完一件事以后才能再做下一件事情。而异步是不需要这样的等待的，有种两件事情同时进行的感觉。
- 发送同步请求在于open方法的第三个参数，如果填写false则代表同步类型的请求，为true代表异步类型的请求。

```
xhr.open( 'get' , 'check..' ,false);
```

- 同步请求的使用场景：
- 在表单验证这种必须等待验证返回后才能点击提交这样的情况下。
- 或者是在onunload事件时通知服务器客户端的关闭也可以用同步请求，有些应用需要知道客户端是否关闭窗口，以此决定是否将服务器端针对该客户端分配的相关资源什么时候释放。如果客户端关闭了浏览器，那么在 onunload事件中可以通知服务器，但是如果不等服务器返回就关掉的话，会不知道是否释放成功，所以通常情况下就是用同步的方式来等待服务器的响应，在一定程度上能保证服务器资源的释放。

使用

```
<html>
  <head>
    <title>json01.html</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <script type="text/javascript">
      function f1(){
        var obj = {'name':'Luffy','age':17};
        alert(obj.name + " " + obj.age);
      }
      function f2(){
        var obj = {'name':'Luffy',
                    'address':{'
                        'city':'Beijing',
                        'street':'dzs',
                        'room':17
                    }};
        alert(obj.name + " " + obj.address+" " +obj.address.city );
      }
    </script>
  </head>

  <body>
    <!-- 创建JavaScript对象并查看属性 -->
    <a href="javascript:;" onclick="f1();">查看对象属性1</a>
    <br>
    <a href="javascript:;" onclick="f2();">查看对象属性2</a>
  </body>
</html>
```

使用

- 应
- 方
- 步
- 试
- 步
- 步
- 步

```
<html>
  <head>
    <title>json02.html</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <script type="text/javascript">
      function f3(){
        var arr = [ {'name':'Luffy','age':17},
                     {'name':'Zoro','age':19}];
        alert(arr[1].name);
      }
      function f4(){
        var obj = {'name':'Luffy',
                   'friends':[{'name':'Zoro','age':19},
                               {'name':'Nami','age':18}]
                   };
        alert(obj.name + " " + obj.friends[1].name);
      }
    </script>
  </head>

  <body>
    <!-- 访问JavaScript对象数组 -->
    <a href="javascript:;" onclick="f3();">查看数组中的对象属性1</a>
    <br>
    <a href="javascript:;" onclick="f4();">查看数组中的对象属性2</a>
  </body>
</html>
```

内测

使用JSON数据交换实例

- 应用问
- 方案:
- 步骤一
- 在文件对象提
- 是因为
- 步骤二
- 步骤三
- 步骤四
- 步骤五
- 步骤六

```
<html>
<head>
  <title>json03.html</title>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">

  <script type="text/javascript" src="js/json.js"></script>
  <script type="text/javascript">

    /*JSON字符串转JSON对象*/
    function f5(){
      var str = '{"name":"Luffy","age":17}';

      //第1种方式(不需要任何js文件)
      //var obj = eval("(" + str + ")");

      //第2种方式(不需要json.js文件)
      //var obj = JSON.parse(str);

      //第3种方式(需要json.js文件)
      //var obj = str.parseJSON();

      alert(obj.name);
    }
  </script>
</head>
</html>
```

原生
结果,

使用

- 应用
- 方案
- 步骤
- 在文档对象模型中
- 步骤
- 步骤
- 步骤
- 步骤
- 步骤

```
/*JSON字符串转JSON数组*/
function f6(){
    var str = ' [{"name":"Luffy","age":17}, ' +
               '{"name":"Zoro","age":19}]';

    //第1种方式(不需要json.js文件)
    var arr = eval("(" + str + ")");

    //第2种方式(需要json.js文件)
    //var arr = str.parseJSON();

    alert(arr[1].name);
}

/*JSON对象转JSON字符串*/
function f7(){
    var obj = {"name":"Luffy","age":17};

    //第1种方式(需要json.js文件)
    //var str = obj.toJSONString();

    //第2种方式(不需要json.js文件)
    var str = JSON.stringify(obj);
    alert(str);
}
</script>
</head>
<body>
    <!-- 使用JSON表示数组 -->
    <a href="javascript:;" onclick="f5();" >JSON字符串-->JSON对象</a><br/><br/><br/>
    <a href="javascript:;" onclick="f6();" >JSON字符串-->JSON数组</a><br/><br/><br/>
    <a href="javascript:;" onclick="f7();" >JSON对象-->JSON字符串</a><br/><br/><br/>
</body>
</html>
```

！。

个原生
出结果，

使用JSON数据

- 应用问题：将
- 方案：使用与
- 步骤：
- 步骤一： 新建
- 步骤二： 新建，
换为JSON字符串
JSON字符串转换
- 步骤三： 引入
<http://sourceforge>
- 步骤四： 添加
- 步骤五： 运行结
- 步骤六： Java
[{"age":19,"name"
- 步骤七： JSON字
- 步骤八： JSON

```
package com.souvc.json;

public class Friend {
    private String name;
    private int age;

    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 toString() {
        return this.name + "    " + this.age;
    }
}
```

试。

式Java对象转
Java对象，

工具包

"Zoro3"]]

17
19

使用JSON数据交换实例

■ JSONTest 代码

json-lib还需要以下依赖包:

```
jakarta commons-lang 2.5  
jakarta commons-beanutils 1.8.0  
jakarta commons-collections 3.2.1  
jakarta commons-logging 1.1.1  
ezmorph 1.0.6
```

```
package com.souvc.json;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import net.sf.json.JSONArray;  
import net.sf.json.JSONObject;  
  
public class JSONTest {  
    /**  
     * Java对象转换为JSON字符串  
     */  
    public static void test1() {  
        Friend f = new Friend();  
        f.setName("Zoro");  
        f.setAge(19);  
        JSONObject jsonObj = JSONObject.fromObject(f);  
        String jsonStr = jsonObj.toString();  
        System.out.println(jsonStr);  
    }  
}
```

使用JSON数据交换实例

■ JSONTest 代码

```
/**
 * Java数组转换为JSON字符串
 */
public static void test2() {
    List<Friend> fs = new ArrayList<Friend>();
    for (int i = 0; i < 3; i++) {
        Friend f = new Friend();
        f.setName("Zoro" + (i + 1));
        f.setAge(19 + i);
        fs.add(f);
    }
    JSONArray jsonArr = JSONArray.fromObject(fs);
    String jsonStr = jsonArr.toString();
    System.out.println(jsonStr);
}

/**
 * JSON字符串转换为Java对象
 */
public static void test3() {
    String jsonStr = "{\"name\":\"Luffy\",\"age\":17}";
    JSONObject jsonObj = JSONObject.fromObject(jsonStr);
    Friend friend = (Friend) JSONObject.toBean(jsonObj, Friend.class);
    System.out.println(friend);
}
```


使用JSON数据交换实例

■ JSONTest 代码

```
/**
 * JSON字符串转换为Java数组
 */
public static void test4() {
    String jsonStr = "[{\"name\":\"Luffy\",\"age\":17},"
        + "          {\"name\":\"Zoro\",\"age\":19}]";
    JSONArray jsonArr = JSONArray.fromObject(jsonStr);
    List<Friend> friends = (List<Friend>) JSONArray.toCollection(jsonArr,
        Friend.class);
    for (Friend f : friends) {
        System.out.println(f);
    }
}

public static void main(String[] args) {
    test1();
    test2();
    test3();
    test4();
}
}
```



使用JSON数据交换实例

- 应用问题：结合异步请求，实现城市的级联下拉列表。
- 方案：分别编写客户端脚本和服务端处理程序。服务器端要实现将Java对象转换为传输的JSON字符串。客户端在收到这个字符串以后进行转换，变成JavaScript对象，使用对象的各个属性构造下拉框的Option选项后添加到select下面。
- 步骤：
 - 步骤一：新建实体类City
 - 步骤二：新建ActionServlet
 - 步骤三：新建city.html文件
 - 步骤四：运行查看结果

使用JSON数据

■ 实体类City

```
package com.fasterxml.jackson;\n\npublic class City {\n    private String cityName;\n    private String cityValue;\n\n    public City() {\n        super();\n    }\n\n    public City(String cityName, String cityValue) {\n        super();\n        this.cityName = cityName;\n        this.cityValue = cityValue;\n    }\n\n    public String getCityName() {\n        return cityName;\n    }\n\n    public void setCityName(String cityName) {\n        this.cityName = cityName;\n    }\n\n    public String getCityValue() {\n        return cityValue;\n    }\n\n    public void setCityValue(String cityValue) {\n        this.cityValue = cityValue;\n    }\n}
```

使用JSON数

■ ActionServlet

```
package com.souvc.json;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import net.sf.json.JSONArray;

public class ActionServlet extends HttpServlet {
    public void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        PrintWriter out = response.getWriter();

        //获得请求路径
        String uri = request.getRequestURI();
        //截取请求路径
        String action = uri.substring(uri.lastIndexOf("/"), uri
            .lastIndexOf("."));
    }
}
```

使用JSON数据

■ ActionServlet

```
if (action.equals("/city")) {
    String name = request.getParameter("name");
    if (name.equals("bj")) {
        City c1 = new City("海淀", "hd");
        City c2 = new City("东城", "dc");
        City c3 = new City("西城", "xc");
        List<City> cs = new ArrayList<City>();
        cs.add(c1);
        cs.add(c2);
        cs.add(c3);
        JSONArray obj = JSONArray.fromObject(cs);
        String str = obj.toString();
        out.println(str);

    } else if (name.equals("sh")) {
        City c1 = new City("徐汇", "xh");
        City c2 = new City("静安", "ja");
        City c3 = new City("黄浦", "hp");
        List<City> cs = new ArrayList<City>();
        cs.add(c1);
        cs.add(c2);
        cs.add(c3);
        JSONArray obj = JSONArray.fromObject(cs);
        String str = obj.toString();
        out.println(str);
    }

}

out.close();
}
```

使用JSON数据交换实例

■ city.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>city.html</title>

    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="this is my page">
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">

    <script type="text/javascript" src="js/json2.js"></script>
    <script type="text/javascript">
function getXmlHttpRequest(){
    var xhr = null;
    if((typeof XMLHttpRequest)!='undefined'){
        xhr = new XMLHttpRequest();
    }else {
        xhr = new ActiveXObject('Microsoft.XMLHttp');
    }
    return xhr;
}
```

使用JSON

■ city.ht

```
function change(v1){
    var xhr = getXmlHttpRequest();
    xhr.open('post','city.do',true);
    xhr.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");
    xhr.onreadystatechange=function(){
        if(xhr.readyState == 4){
            var txt = xhr.responseText;
            var citys = txt.parseJSON();
            document.getElementById('s2').innerHTML = '';
            for(i=0;i<citys.length;i++){
                var op =
                    new Option(citys[i].cityName,
                        citys[i].cityValue);
                document.getElementById('s2').options[i] = op;
            }
        }
    };
    xhr.send('name=' + v1);
}
</script>
</head>

<body>
    <select id="s1" style="width: 120px;" onchange="change(this.value);">
        <option value="sh">
            上海
        </option>
        <option value="bj">
            北京
        </option>
    </select>
    <select id="s2" style="width: 120px;">
    </select>
</body>
</html>
```



使用JSON数据交换实例

- 热卖商品动态显示
- 应用问题：每隔5秒钟，显示当前卖的最好的三件商品。
- 方案：每5秒钟发送一次Ajax请求，将返回的JSON数组数据显示到页面的div中。
- 步骤：
 - 步骤一：新建Sale类
 - 步骤二：新建dao包下面的DBUtil类和SaleDAO类，SaleDAO.java文件执行sql语句，实现用于查询销量前三的方法。
 - 步骤三：新建ActionServlet类
 - 步骤四：新建sales.html文件
 - 步骤五：运行查看结果

使用JSON数据交换实例

■ Sale类

```
package com.souvc.json;

public class Sale {
    private int id;
    private String prodName;
    private int qty;

    public Sale() {
        super();
    }

    public Sale(int id, String prodName, int qty) {
        super();
        this.id = id;
        this.prodName = prodName;
        this.qty = qty;
    }

    public int getId() {
        return id;
    }
}
```

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

    public String getProdName() {
        return prodName;
    }

    public void setProdName(String prodName) {
        this.prodName = prodName;
    }

    public int getQty() {
        return qty;
    }

    public void setQty(int qty) {
        this.qty = qty;
    }
}
```

使用JSON数

■ DBUtil.java

```
package com.souvc.json;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

/**
 * JDBC管理连接的工具类，可以获取连接和关闭连接
 */
public class DBUtil {
    /**
     * 获取连接对象
     */
    public static Connection getConnection() throws Exception {
        Connection conn = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/csdn", "root", "123456");
        } catch (Exception e) {
            e.printStackTrace();
            throw e;
        }
        return conn;
    }
}
```

使用JSON数据交换实例

■ DBUtil.java

```
/**
 * 关闭连接对象
 */
public static void close(Connection conn) throws Exception {
    if (conn != null) {
        try {
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
            throw e;
        }
    }
}
```

使用JSC

■ SaleDAO

```
package com.souvc.json;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class SaleDAO {
    public List<Sale> findAll() throws Exception {
        List<Sale> sales = new ArrayList<Sale>();
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try {
            conn = DBUtil.getConnection();
            stmt = conn.prepareStatement("select * from (select rownum r,"
                + "a.* from (select * from t_sale order by qty desc) a) "
                + "c where c.r < 4");
            rs = stmt.executeQuery();
            while (rs.next()) {
                Sale s = new Sale(rs.getInt("id"), rs.getString("prodname"), rs
                    .getInt("qty"));
                sales.add(s);
            }
        } catch (Exception e) {
            e.printStackTrace();
            throw e;
        } finally {
            DBUtil.close(conn);
        }
        return sales;
    }
}
```

使用JS

■ ActionServlet

```
package com.souvc.json;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import net.sf.json.JSONArray;

public class ActionServlet extends HttpServlet {
    public void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        PrintWriter out = response.getWriter();

        // 获得请求路径
        String uri = request.getRequestURI();
        // 截取请求路径
        String action = uri.substring(uri.lastIndexOf("/"), uri
            .lastIndexOf("."));
    }
}
```

使用JSON数据

■ ActionServlet

```
if (action.equals("/city")) {
    String name = request.getParameter("name");
    if (name.equals("bj")) {
        City c1 = new City("海淀", "hd");
        City c2 = new City("东城", "dc");
        City c3 = new City("西城", "xc");
        List<City> cs = new ArrayList<City>();
        cs.add(c1);
        cs.add(c2);
        cs.add(c3);
        JSONArray obj = JSONArray.fromObject(cs);
        String str = obj.toString();
        out.println(str);

    } else if (name.equals("sh")) {
        City c1 = new City("徐汇", "xh");
        City c2 = new City("静安", "ja");
        City c3 = new City("黄浦", "hp");
        List<City> cs = new ArrayList<City>();
        cs.add(c1);
        cs.add(c2);
        cs.add(c3);
        JSONArray obj = JSONArray.fromObject(cs);
        String str = obj.toString();
        out.println(str);
    }

} else if (action.equals("/sale")) {
```

使用JSON数据交换实例

■ ActionServlet

```
        SaleDAO dao = new SaleDAO();
        try {
            List<Sale> all = dao.findAll();
            JSONArray array = JSONArray.fromObject(all);
            out.println(array.toString());
        } catch (Exception e) {
            e.printStackTrace();
            throw new ServletException(e);
        }
    }
    out.close();
}
```

使用JSON数据交换实例

■ sales.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>sales.html</title>

    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="this is my page">
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <style>
#d1 {
  width: 500px;
  height: 180px;
  background-color: #fff8dc;
  border: 1px solid red;
  margin-left: 350px;
  margin-top: 50px;
}
</style>
```




使用JSON数据交换实例

■ sales.html

```
<script type="text/javascript" src="js/prototype-1.6.0.3.js"></script>
<script type="text/javascript">
function getXmlHttpRequest() {
    var xhr = null;
    if((typeof XMLHttpRequest) != 'undefined') {
        xhr = new XMLHttpRequest();
    } else {
        xhr = new ActiveXObject('Microsoft.XMLHttp');
    }
    return xhr;
}
function f1() {
    setInterval(f2, 5000);
}
```

使用JSON

■ sales.h

```
function f2() {
    var xhr = getXmlHttpRequest();
    xhr.open('post', 'sale.do', true);
    xhr.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");
    xhr.onreadystatechange=function() {
        if(xhr.readyState == 4){
            var txt = xhr.responseText;
            var sales = txt.evalJSON();
            var saleInfo = '当前销量最好的商品是:<br/>';
            for(i=0;i<sales.length;i++){
                saleInfo += '商品名称:'
                    + sales[i].prodName + ' 销量:' +
                    sales[i].qty + '<br/>';
            }
            $('d1').innerHTML = saleInfo;
        }
    };
    xhr.send(null);
}
</script>
</head>

<body onload="f1();">
    <div id="d1"></div>
</body>
</html>
```



思考与讨论

- 你所理解的JSON?
- JSON有哪些语法规则要遵守?
- 如何应用JSON编程交换数据?

课后兴趣作业

- 练习在客户机和服务器之间通信应用JSON编程交换数据