面向对象程序设计

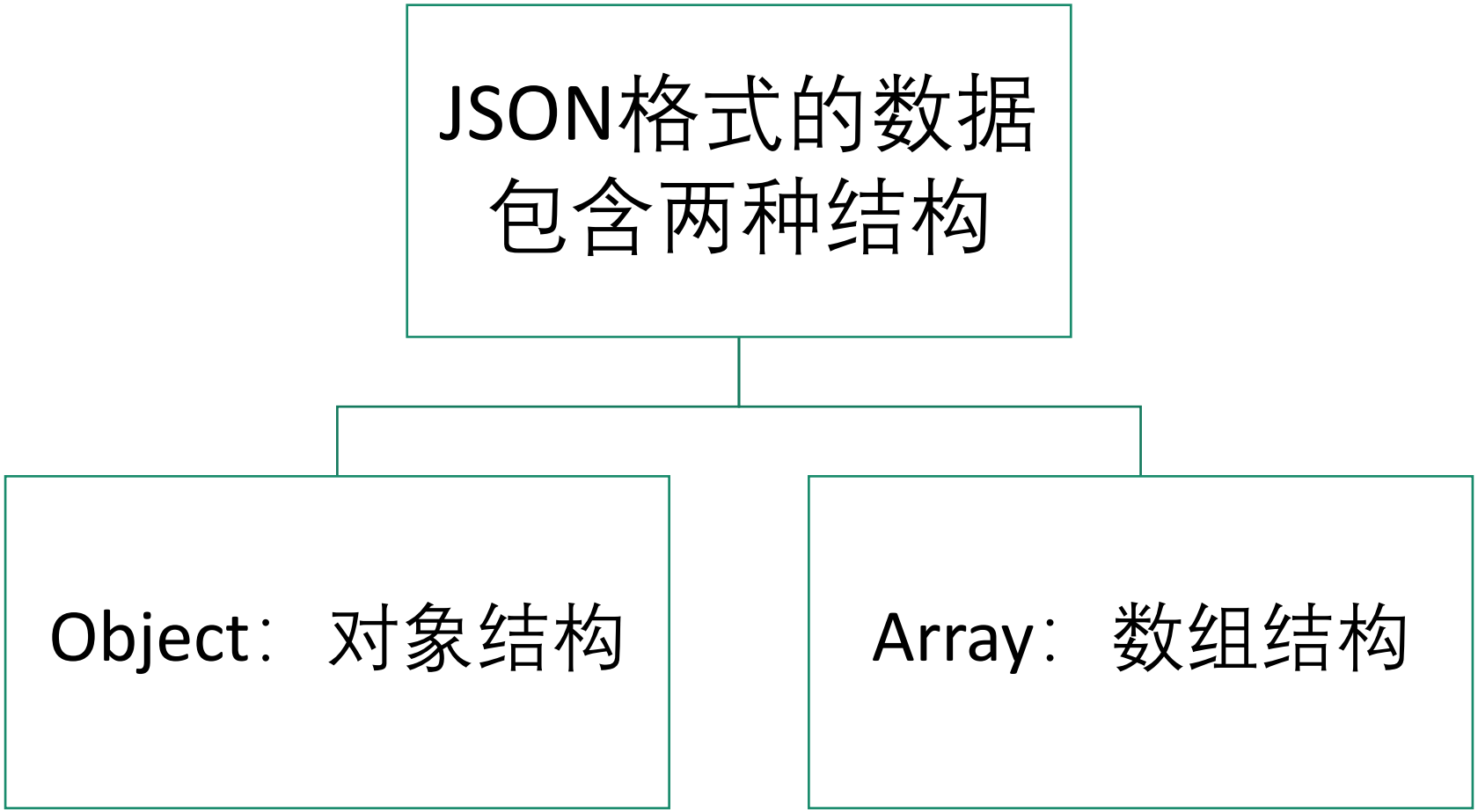
JSON数据解析

赵毅力
大数据与智能工程学院
西南林业大学

JSON概述

- **JSON (JavaScript Object Notation)**，是一种基于文本、独立于编程语言的轻量级的数据交换格式。
  - 从JavaScript的对象格式演变而来：方便JavaScript进行处理
  - 基于文本：可以用任何文本编辑器进行编辑
  - 独立于编程语言：任何编程语言都支持JSON格式的数据解析
  - 轻量级：不像XML那样繁琐
  - 数据交换：目前Web前端和后端数据交换事实上的标准
- https://www.json.org/

JSON的两种结构



JSON的对象结构

- JSON格式的对象结构：用{}包含的一系列无序的键值对，其中键只能用字符串表示，键和值用冒号分隔，键值对用逗号分隔。

```
{
    "name": "John",
    "email": "john@example.com"
}
```

JSON的数组结构

- JSON格式的数组结构：使用[]包含的元素列表，每个元素用逗号分隔。

```
[
    "Red",
    "Green",
    "Blue"
]
```

JSON支持的数据类型

- **String**：字符串类型
  - “name”
  - “姓名”
- **Number**：数值类型
  - 123
  - 3.1415926
- **Boolean**：布尔类型
  - true
  - false

JSON对象格式

```
{
   "id": 1,
   "name": "Java程序设计",
   "authors": [
      "耿祥义",
      "张跃平"
   ],
   "isbn": "9787302473169",
   "tags": ["Java", “Web"]
   “price": 45.8
}
```

JSON格式的数据解析

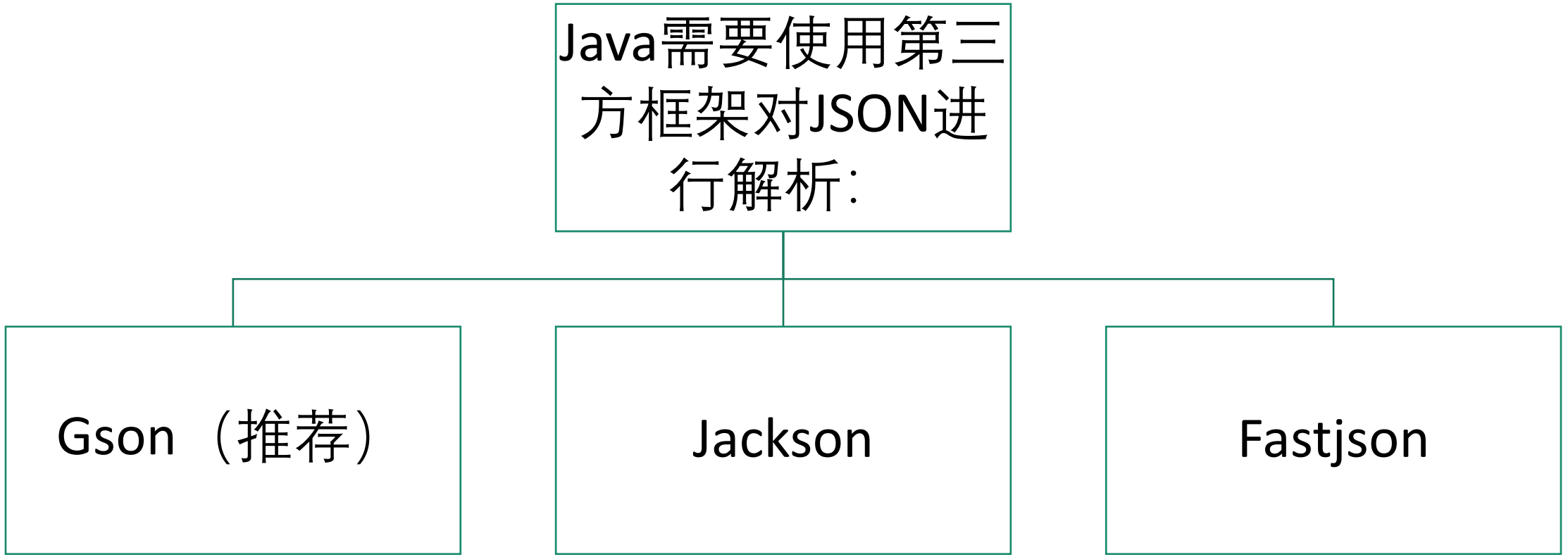
- 现代浏览器直接支持使用JavaScript语言对JSON数据进行读写。

```
// 将JSON字符串转换为JavaScript的对象
jsObj = JSON.parse(jsonStr);

// 将JavaScript对象转换为JSON字符串
jsonStr = JSON.stringify(jsObj);
```

- 可以通过https://playcode.io/javascript进行在线测试。

JSON格式的数据解析



使用Gson对JSON进行解析

- 要使用Gson框架，首先需要在项目的pom.xml文件中添加依赖。

```
<dependencies>
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>2.10.1</version>
    </dependency>
</dependencies>
```

使用Gson对JSON对象结构进行解析

- 其次在项目中添加和JSON数据对应的实体类。

```
{
   "id": 1,
   "name": "Java程序设计",
   "authors": [
      "耿祥义",
      "张跃平"
   ],
   "isbn": "9787302473169",
   "tags": ["Java", “Web"],
   "price": 45.8
}
```

```
// 书籍实体类
public class BookBean {
   private int id;
   private String name;
   private List<String> authors;
   private String isbn;
   private List<String> tags;
   ……
```

使用Gson对JSON对象结构进行解析

- 然后在项目中生成**Gson**类的对象，并调用对象的**fromJson**方法。

```
public class Demo {
    public static void main(String[] args) {
        final String jsonFileName = "data.json";
        Demo demo = new Demo();
        Gson gson = new Gson();
        String jsonStr = demo.readJsonFile(jsonFileName);
        BookBean bookBean = gson.fromJson(jsonStr, BookBean.class);
        ......
    }
}
```

使用Gson对JSON数组结构进行解析

- 第一种方法在fromJson方法中传递数组类型信息。

```
public class Demo {
    public static void main(String[] args) {
        final String jsonFileName = "data.json";
        Demo demo = new Demo();
        Gson gson = new Gson();
        String jsonStr = demo.readJsonFile(jsonFileName);
        BookBean[] bookBeans = gson.fromJson(jsonStr, BookBean[].class);
        for (BookBean book : bookBeans) {
            System.out.println(book);
        }
    }
}
```

使用Gson对JSON数组结构进行解析

- 第二种方法是基于反射，使用**TypeToken**在**fromJson**方法中传递**List<T>**的类型信息。

```
public class Demo {
    public static void main(String[] args) {
        final String jsonFileName = "data.json";
        Demo demo = new Demo();
        Gson gson = new Gson();
        String jsonStr = demo.readJsonFile(jsonFileName);
        TypeToken<List<BookBean>> typeToken = new TypeToken<List<BookBean>>(){};
        List<BookBean> bookBeans = gson.fromJson(jsonStr, typeToken.getType());
        ......
    }
}
```

Java泛型的限制

- Java的泛型采用“**类型擦除**”的方式进行实现：
  - **ArrayList<BookBean>**经过编译以后变为**ArrayList<Object>**
  - 即经过编译以后就丢失了具体的类型信息

- 在使用Gson的**fromJson**方法时需要提供具体类型的**class**对象

- 如何提供**ArrayList<BookBean>**的**class**对象？
  - 直接使用ArrayList<BookBean>.class不可以，因为编译后丢失BookBean的类型信息。

使用TypeToken获取泛型类的class对象

- Gson框架使用反射机制和匿名类来提供泛型类型的class对象。
- 使用TypeToken保存泛型类的类型信息：

```
TypeToken<List<BookBean>> typeToken = new TypeToken<List<BookBean>>(){};
```

- 调用TypeToken类的getType方法获取泛型类的class对象。

```
List<BookBean> bookBeans = gson.fromJson(jsonStr, typeToken.getType());
```

使用node.js快速创建一个Web服务器

- **node.js**是一个基于 Chrome V8 引擎的 JavaScript 语言运行环境
  - 底层网络通信使用**libuv**，是一个基于事件驱动、非阻塞式 I/O 的模型
  - 使得 JavaScript 语言能够运行在服务器端
  - 使用**npm**进行包的管理
- https://nodejs.org/

使用node.js快速创建一个Web服务器

```
let http = require('http');
// 定义服务器监听端口号
const port = 8899;
let data = { "id":1, "name":"Java程序设计","authors":["耿祥义","张跃平"],"isbn":"9787302473169","tags":["Java","Web"],"price":45.8
};
// 定义HTTP响应函数对象
const requestListener = function (req, res) {
    res.writeHead(200,
        {'Content-Type':'application/json;charset=UTF-8'}
    );
    res.end(JSON.stringify(data));
}
// 创建HTTP服务器并注册回调函数
const server = http.createServer(requestListener);
// 启动服务器并监听端口
server.listen(port, () => {
    // 在终端打印信息
    console.log(`服务器正在监听端口： ${port}`);
});
```

使用Java处理HTTP协议

- 使用第三方框架Apache HttpClient
- https://mvnrepository.com/artifact/org.apache.httpcomponents.client5/httpclient5