第35卷 第19期 计 算 机 工 程 2009年10月
Vol.35 No.19 Computer Engineering October 2009
·软件技术与数据库· 文章编号：1000—3428(2009)19—0071—04 文献标识码：A 中图分类号：TP311.52

基于 JSON 和 IoC 的 AJAX-RMI 插件

黄 强[1]，王 薇[2]，张晓梅[3]，李 敏[3]
(1. 四川农业大学信息与工程技术学院，雅安 625000；2. 西南交通大学峨嵋校区计算机与通信工程系，峨嵋 614202；
3. 西南交通大学信息网络中心，成都 610031)

摘 要：异步 JavaScript 和 XML(AJAX)客户端的 JavaScript 与服务组件之间的通信存在较大阻抗，AJAX 服务构件难以与现有 MVC 框架进行无侵略集成。基于结构化 XML 的数据交换体系占用系统资源和传输带宽较大，其结构不能有效适配客户和服务器系统。针对上述问题设计基于 JSON 数据交换、能与目前主流 MVC 框架进行无缝集成的 AJAX 远程方法调用插件 MyAJAX，结合控制反转容器，提出服务组件的 JavaScript 动态存根生成和 JSON 序列化模型，实现 JavaScript 与容器内服务组件间的透明远程调用，给出 MyAJAX 的应用实例。
关键词：异步 JavaScript 和 XML 技术；JSON 序列化；动态存根生成；同构对象

AJAX-RMI Plug-in Based on JSON and IoC

HUANG Qiang[1], WANG Wei[2], ZHANG Xiao-mei[3], LI Min[3]
(1. School of Information and Engineering Technology, Sichuan Agricultural University, Yaan 625000;
2. Department of Computer and Communication Engineering, Southwest Jiaotong University Emei Campus, Emei 614202;
3. Information and Network Center, Southwest Jiaotong University, Chengdu 610031)

【Abstract】 Asynhronous JavaScript and XML(AJAX) has a big resistance to the communication between JavaScript at client side and the service module. AJAX service component is difficult to be integrated with current MVC framework non-invasively. Data exchange system based on structural XML occupies many system resources and transmission bandwidth, so its structure can not be matched effectively between client and server system. Aiming at these problems, a plug-in called MyAJAX based on JSON data exchange which can be seamlessly integrated with current mainstream MVC frameworks and can be interfaced by AJAX remote method is designed. With the Inverse of Control(IoC) container, the JavaScript dynamic stub generation and the JSON serializing model of service module are proposed to realize transparent remote call between JavaScript and the service module inside the container. Application instance of MyAJAX is given.
【Key words】 Asynhronous JavaScript and XML(AJAX) technology; JSON serialization; dynamic stub generation; isomorphic object

1 概述

异步 JavaScript 和 XML(Asynchronous JavaScript and XML, AJAX)技术很好地弥补了 Web 界面上 B/S 系统对用户自然和响应灵敏度方面的不足[1]，但现有 AJAX 服务构件与 MVC 框架之间的集成过程较繁琐，且侵略性大。由于 AJAX 技术主要使用基于 XML 结构化文档的数据交换模式，因此不能很好地融入面向对象的服务系统，其数据生成和解析都需要程序维护，不便于系统开发。此外，JavaScript 中基于 DOM 的 XML 解析模型会占用较多系统资源，导致系统性能低下，JavaScript 和服务器组件之间的协作问题增加了 AJAX 系统的复杂性。鉴于此，本文提出一种基于 JSON(JavaScript Object Notation)的对象序列化算法，使 AJAX 客户端能使用 JavaScript 透明地处理服务器端产生的结果对象。使用 JSON 作为桥梁避免 XML 的产生和解析过程。设计基于 JSON 对象序列化的 AJAX 远程方法调用插件 MyAJAX，为客户端提供透明的远程组件调用服务，并支持 JavaScript 调试模式和数据跟踪，有效简化并规范 AJAX 系统开发。

2 MyAJAX 插件设计和核心算法

2.1 MyAJAX 插件在 J2EE 服务系统中的作用

现有企业应用构架均采用多层结构，MyAJAX 作为表现层(presentation layer)的前端为浏览器客户端提供 AJAX 远程组件调用服务的 ORM 框架均在系统持久层(persistence layer)使用，MyAJAX 在 J2EE 服务系统中的作用如图 1 所示[2]。

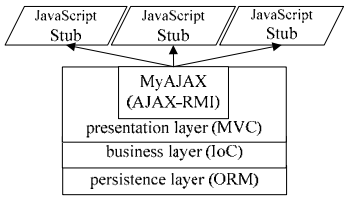

图 1 MyAJAX 在 J2EE 服务系统中的作用

与其他 AJAX 框架相比，MyAJAX 是一个轻量级插件，它只关注浏览器和服务器间的远程数据交换而不是用户界面，无须任何配置。只要将 MyAJAX 挂接到一个 URL 上就可以为浏览器客户端提供 AJAX 远程方法调用服务，它内置了对各种 MVC 框架的集成接口，很容易实现和各种 MVC 框架之间的无缝集成，有效减少系统移植代价。

2.2 MyAJAX 体系结构设计

MyAJAX 远程调用框架采用 IoC (Inverse of Control)容器进行组件管理，实现了基于 JSON 的上下行数据转换算法，

基金项目：四川省软件重点实验室基金资助项目“移动农业智能专家系统的设计与开发”(J07006)
作者简介：黄 强(1981－)，男，讲师、硕士，主研方向：软件工程，虚拟机，分布式计算系统，电子商务；王 薇，助教、硕士；张晓梅、李 敏，硕士
收稿日期：2009-04-29 **E-mail：**yellowicq@126.com

并针对不同 MVC 框架提供了表现层门面接口，能较大程度简化 AJAX 开发，并实现与 MVC 框架的无缝集成。MyAJAX 使用 JavaScript 动态存根生成技术使客户端存根与组件库的使用融为一体，让远程方法调用变得容易。MyAJAX 插件的体系结构如图 2 所示。

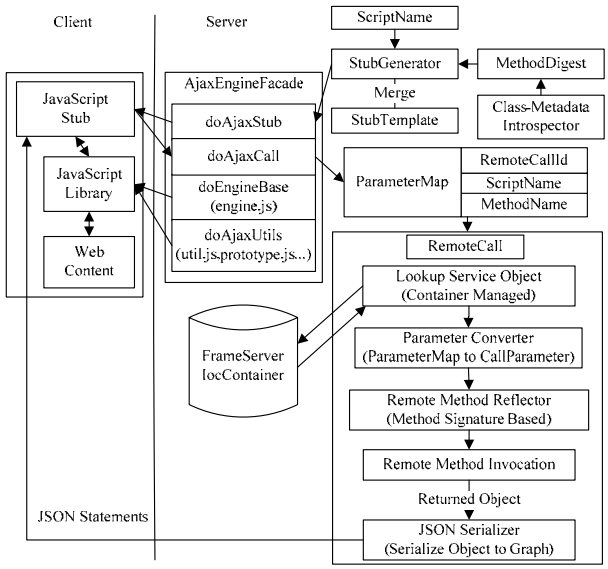

图 2　MyAJAX 插件的体系结构

下文将详细介绍 MyAJAX 插件最显著的特点，即动态存根生成，远程方法调用和对象的 JSON 序列化。

2.3　MyAJAX 的动态存根设计

MyAJAX 的动态存根生成器由类元数据自省器和存根模版 2 个部分构成。自省器先使用反射机制分析服务对象完成方法信息提取，然后按图 3 定义的格式进行填充，最后使用模版引擎将信息填充进 JavaScript 存根模版，完成存根的动态生成工作。存根包含了服务名、方法名、参数类型等信息供远程调用使用[3-4]。MyAJAX 存根生成器逻辑结构见图 3。

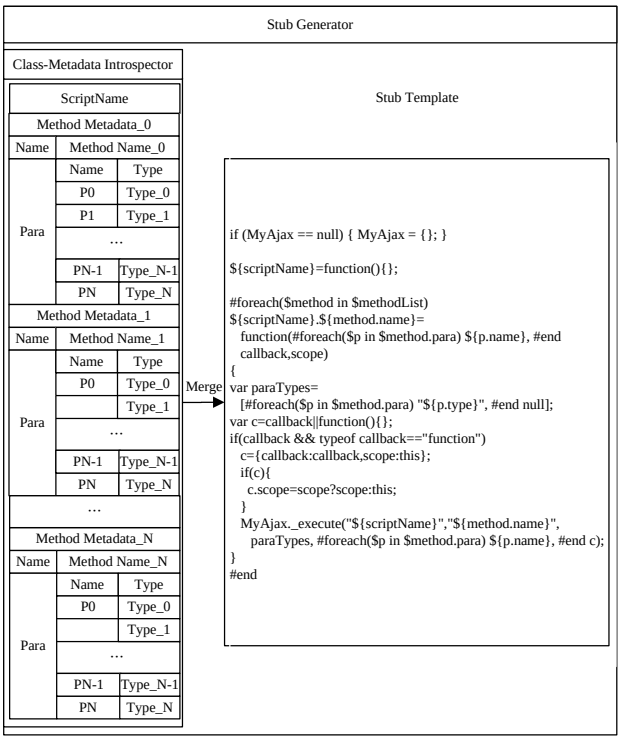

图 3　MyAJAX 存根生成器逻辑结构

MyAJAX 的存根生成器会对非公有方法进行过滤并禁止对私有信息的远程访问，有效提高了系统的安全性[3-4]。一个简单的 DateService 服务源代码及其生成的动态存根描述如下：

```
//Java Source File
…//Package and imports
public class DateService(){
    public    Stromg defaultDate(){
    return new java.util.Date().toString();
     }
  …//Other methods
}
//Javascript Stub
if (MyAJAX==null){MyAJAX={};}
DateService=function(){};
DateService.defaultDate()=function(callback,scope){
  var paraTypes=[null];
  c=callback;
  if(callback && typeof callback=="function")
        c={callback:callback,scope:this};
  if(c){
        c.scope=scope?scope:this;
  }
  MyAJAX._execute("DateService","defaultDate",paraTypes,c);
}
…//Stub for other methods
```

2.3.1　MyAJAX 的远程方法调用与参数传递

MyAJAX 的远程方法调用由参数封装、请求调用和结果处理 3 个部分组成。在用户发出请求命令后，MyAJAX 的客户端组件库使用 JavaScript 脚本提取表单数据，并使用生成的存根对远程调用接口进行翻译，发送符合 AJAX 规范的异步数据到远程服务器，然后由 MyAJAX 的 Facade 解析相关请求参数并调用客户端指定的服务方法，最后 MyAJAX 使用 JSON 对象序列化器对结果对象进行序列化并传输到客户端，客户端反序列化 JSON 对象为 JavaScript 内存对象并完成结果数据的提取和用户界面更新[5]。

MyAJAX 发送请求参数时按 HTTP 协议要求的键值对格式组织，对于每个远程方法参数需要传递参数类型和参数值 2 个信息，格式如下：

```
ajax-call-p[i].value=[valueString]&
ajax-call-p[i].java-type=[className]
```

插件在服务器端会分析上述格式字符串并提取出括号中的[i], [valueString]和[className]信息，将其作为方法反射的参数[5]。

2.3.2　Java 的同构 JavaScript 对象

MyAJAX 在进行服务器端远程方法调用后的结果是一个 Java 对象，而客户端使用的是 JavaScript，具有相同逻辑结构的 Java 对象和 JavaScript 对象是一对同构对象。Java 作为一种完全面向对象的语言，可以直接使用对象之间的引用关系来表示一个对象图，而 JavaScript 是一种动态脚本语言，只能通过动态变量来模拟对象之间的相互关系，MyAJAX 使用 JSON 生成器(JSONGenerator)作为桥梁实现 Java 对象的序列化和 JavaScript 同构对象的 JSON 描述，MyAJAX 的 JSON 对象序列化模型如图 4 所示[2]。

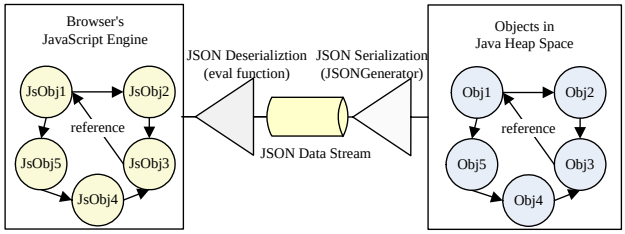

图 4　MyAJAX 的 JSON 对象序列化模型

由图 4 可知，在服务器端，由 JSONGenerator 完成 Java 对象的 JSON 序列化，客户端则通过浏览器使用 JavaScript 的 eval 函数完成同构对象的 JSON 反序列化，生成 JavaScript 同构对象供回调函数使用，Java 对象和同构 JavaScript 对象在逻辑结构上完全相同，具有相同的对象图，只是所存在的空间不同。上述存储结构对结果处理具有较高透明性，便于数据获取。

2.3.3　JSONGenerator 的结构模型

MyAJAX 使用 JSONGenerator 完成对象序列化工作，JSONGenerator 中包括 ObjectResolver 和 JavaScript Composer 2 个模块。ObjectResolver 负责分析结果对象，并生成同构 JavaScript 对象的 JSON 语句描述，ObjectResolver 使用有限状态机模型遍历对象图，能序列化简单对象并很好地处理集合对象、对象递归和对象循环引用等问题。ObjectResolver 在遍历对象的同时将对象信息数据存储在堆栈中传递给 JavaScript 组装器，组装器将对象信息转换成同构的 JavaScript 变量声明、变量赋值和函数体语句，完成对象的 JSON 序列化。JSONGenerator 的结构模型如图 5 所示。

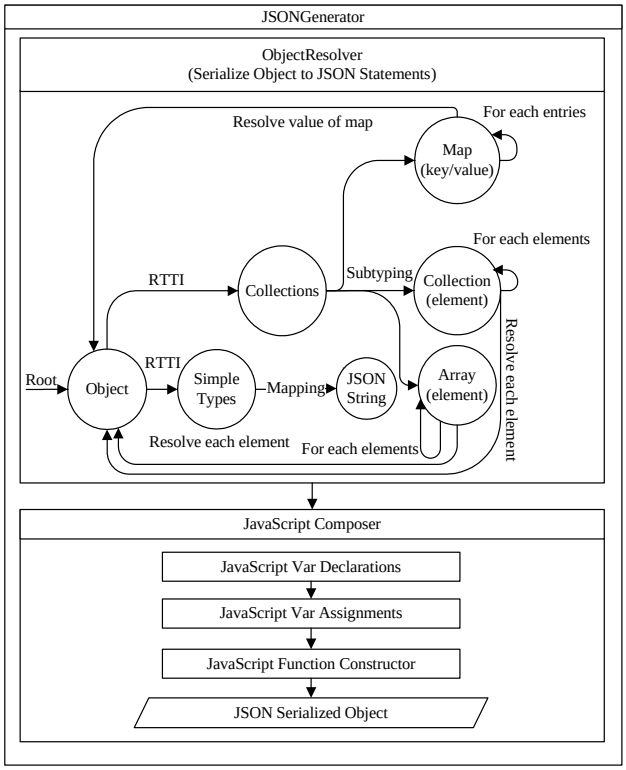

图 5　JSONGenerator 的结构模型

2.4　MyAJAX 在实际项目中的应用

MyAJAX 插件配合 MVC 框架能够极大减少 AJAX 系统的代码，帮助开发人员快速完成系统的表现层开发。使用直接远程方法调用模式可以完全省去服务器端的表现层代码，MyAJAX 的 JavaScript 库对很多通用功能进行了封装(如分页组件)，可以直接使用，避免代码冗余。下文以 IFarmer 系统的用户信息分页查询模块为例，要使用 AJAX-RMI 模式实现该模块，只要进行 3 步简单操作。先创建 Java 远程服务，其代码片段如下：

```
…
public class SpeedUserService implements
   UserService {
   …
   //使用电话号码查询用户信息
   public PageList queryUsers(String tel,
                  int currentPage, int pageSize) {
   Pagination p = new Pagination();
   p.setCount(sizePerPage); //defaulting to 15
   p.setPage(currentPage);
 List<SpeedUser> list =
this.getSpeedTemplate().
              getResults(
                 "select * from ifuser where tel
                    like '%"+tel+"%'",
                    new String[] {}, IFUser.class, p);
 PageList pageList = new SpeedPageList(p, list);
 return pageList;
}
}
```

然后配置服务组件到服务器的 IoC 容器中：

```
<component key="UserService"
   class="org.javawing.ifarmer.service.
          SpeedUserService" scope="singleton">
</component>
```

最后在客户端使用以下 JavaScript 脚本导入并使用动态存根调用远程服务，完成结果数据的处理(MyAJAX 规定动态存根名必须于组件在容器中的 key 相同)：

```
<!-- 导入 JavaScript 库文件 -->
<script type="text/javascript"
    src="do/myajax/prototype.js"></script>
<script type="text/javascript"
    src="do/myajax/engine.js"></script>
<script type="text/javascript"
    src="do/myajax/util.js"></script>
<!-- 导入 JavaScript 动态存根 -->
<script type="text/javascript"
src="do/myajax/UserService.js"></script>
…
function queryUsers(tel,currentPage) {
<!-- 进行远程服务调用 -->
   UserService.queryUsers(tel,currentPage,
                          15,showResult);
}
<!-- 局部更新用户界面 -->
function showResult(pageList) {
   MyAJAXUtil.pageList=pageList;
   var currentPage=pageList.currentPage;
   var count=(pageList.currentPage-1)*
                    pageList.pageSize+1;
   var celFuns=[
        function(data){return count++;},
        function(data){return data.userName;},
        function(data){return data.tel;},
        function(data){return data.birthday;}
```

```
    ];
    MyAJAXUtil.removeAllRows("mainTable");
    MyAJAXUtil.addRows("mainTable",
                       pageList.result,celFuns);
    MyAJAXUtil.showPageInfo();
}
<!-- 处理分页 -->
MyAJAXUtil.queryPage=function(currentPage){
    queryUsers($("tel").value,currentPage);
}
```

打开 MyAJAX 调试模式可以跟踪一次查询的请求参数串和返回的 JSON 数据，如图 6 所示。

图 6　Debug 模式下一次查询请求参数串和 JSON 结果数据的跟踪

3　结束语

本文介绍基于 JSON 数据交换和 IoC 容器的 AJAX 远程方法调用插件 MyAJAX 的模型设计、实现和特点。MyAJAX 插件具有以下优点：(1)使用基于 JSON 数据交换的直接远程服务调用模式有效消除了表现层代码；(2)通过客户端的 JavaScript 函数库可以方便地使用和控制驻留在服务器 IoC 容器中的服务组件；(3)提供丰富的 UI 辅助函数，帮助开发人员快速完成 AJAX 系统的视图更新；(4)以服务组件为核心的构架让系统设计人员可以专注于核心业务设计而不用考虑很多客户端与服务器的协作问题，且轻量级结构可以让 MyAJAX 在各种 MVC 框架之间进行无侵略集成。MyAJAX 的易操作性使其在中小型 AJAX 项目的实践中取得了较好的效果，简洁清晰的结构使开发人员之间有效合作，迅速完成业务模块的搭建，提高团队整体效率。MyAJAX 在进行嵌套对象保存的参数分析时存在一些不足，需要在后续开发中逐渐完善。

参考文献

[1] Garrett J J. AJAX: A New Approach to Web Applications[EB/OL]. (2005-02-11). http://adaptivepath.com/ideas/essays/archives/000385.php.

[2] JSON Organization. Introducing JSON[EB/OL]. [2008-11-20]. http://www.json.org.

[3] 黄　强. MyAJAX 插件接口设计规范[Z]. 2007.

[4] 黄　强. DWR 与 MyAJAX 的性能对比分析[EB/OL]. [2008-11-20]. http://jsjx.sicau.edu.cn/wingstudio/myajax.

[5] Crane D. AJAX in Action[M]. [S. l.]: Manning Publications, 2005.

编辑　陈　晖

（上接第 70 页）

3.4　SOA 核心服务实现

核心服务整体建立在 J2EE 平台上，服务代码使用 Java 语言实现。服务调用使用简单对象访问协议(SOAP)，采用 XML 文档作为调用参数及返回结果。本文以航班动态查询服务为例介绍核心服务实现。其实现流程如图 4 所示。

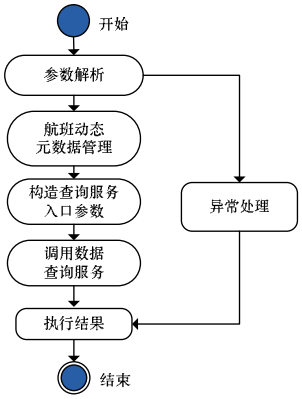

图 4　航班动态查询流程

首先通过服务接口组件对输入的 XML 文档进行解析，得到查询条件。然后构造查询请求，通过服务网关调用数据查询服务，进行数据查询。将获取的结果以 XML 文档格式返回给调用者。其中，航班计划元数据管理主要针对特定的业务数据，对其信息组织方式进行管理(查询)。

4　结束语

设计模式的使用有效地提高了软件的设计质量，加强了开发人员对系统的理解。在民航综合信息平台的设计中，通过使用接口模式和网关模式，将负责应用程序业务逻辑的元素与负责和服务通信并参与服务合约的元素分开，从而使设计出的系统有较高的灵活性、可维护性及可测试性。

参考文献

[1] 麻志毅, 陈泓婕. 一种面向服务的体系结构参考模型[J]. 计算机学报, 2006, 29(7): 1-9

[2] IMB. SOA 术语概述：服务、体系结构、治理和业务术语[EB/OL]. [2008-12-04]. http://www.ibm.com/developerworks/cn/webservices/ws-soa-term1/index.html.

[3] Thomas E. SOA 概念技术与设计[M]. 北京: 机械工业出版社, 2007: 162-172.

[4] 李长云, 李赣生, 李　莹, 等. 面向服务的软件开发[J]. 计算机工程, 2005, 31(17): 52-54.

[5] Microsoft. .NET 的企业解决方案模式[EB/OL]. [2008-06-04]. http://www.microsoft.com/china/MSDN/library/architecture/pattern/esp/espdefault.mspx.

[6] Erich G. 设计模式: 可复用面向对象软件的基础[M]. 北京: 机械工出版社, 2005: 121-122.

编辑　张　帆