第30卷 第3期 计 算 机 工 程 2004年2月
Vol.30 №3 Computer Engineering February 2004
·软件技术与数据库· 文章编号：1000—3428(2004)03—0071—03 文献标识码：A 中图分类号：TP311

C/S向B/S系统迁移的技术

王心路，赵文耘，余 春
（复旦大学计算机系软件工程实验室，上海 200433）

摘 要：随着Internet技术的兴起，浏览器/服务器(B/S)系统相对于传统的客户/服务器(C/S)系统的优势越来越明显，更多的企业正着手将C/S系统向B/S迁移。该文在Jakarta Struts框架和Web Service技术的基础上，提出了一种将传统的C/S系统向B/S迁移的方法，并对其中的关键技术如用户接口的自动迁移、处理逻辑的包装以及Web应用的集成进行了深入的探讨。

关键词：Struts；Web Service；迁移技术

Migration Technology from C/S to B/S

WANG Xinlu, ZHAO Wenyun, YU Chun
(S. E. Lab, Dept. of C. S, Fudan Univ., Shanghai 200433)

【Abstract】 With the development of the Internet, more and more corporations have recognized that the Browser/Server system is much better than the Client/ Server system and have began to transfer the C/S system to the B/S system. This article describes a new technique of system transfer from C/S to B/S, which is based on the Struts framework and Web Service. The paper also presents the key techniques in realizing the system transfer, such as the automatic migration of the user interface, how to wrap the logic of the operation and the integration of the Web System.

【Key words】 Struts; Web service; Migration technology

1 概述

1.1 C/S结构与B/S结构

随着分布式计算机技术的发展，出现了客户/服务器(Client/Sserver)系统，主要具有如下特点：(1) 负载均衡。通过将任务合理地分配到Client端和Server端，降低了系统的通信开销，并可充分利用两端硬件环境的优势；(2)数据安全及完整性。具有强壮的数据操纵和事务处理能力，保证数据的安全性和完整性约束。

但随着企业规模的日益扩大，应用程序的复杂程度不断提高，C/S系统逐渐暴露出以下不足：(1) 系统难以部署和维护，特别是在客户端数量巨大的情况下，安装及升级成本急剧上升；(2) 兼容性差，不同的开发工具及系统运行平台相互之间很难兼容。

因此，随着Internet技术的兴起，出现浏览器/服务器（Browser/Server）系统，在这种结构下，用户界面完全通过WWW浏览器实现，只处理少量事务逻辑；主要事务逻辑在服务器端实现，并且根据服务器处理业务的层次划分，形成所谓3-tier或n-tier结构，具有以下的特点：(1) 具有分布性特点，可以随时随地进行业务处理；(2) 业务扩展维护简单方便，只需要改变服务器，即实现所有用户的同步更新；(3) 开发简单，共享性强。

显然B/S系统相对于传统的C/S系统是巨大的进步，越来越多的企业考虑采用B/S系统来解决实际应用。但由于开发平台的不同，在C/S向B/S迁移时，所有的代码必须重新编写，如果能使用一种技术，使得已有程序能够在很少改动的基础上被跨平台使用，这将极大地提高程序的复用率，降低迁移成本。

1.2 Jakarta Struts框架

Jakarta Struts是一个使用Model 2 JSP Web应用程序框架，它用Servlet作为请求调度器，用JavaBean承载请求数据，并用JSP将数据呈现给用户。使用Struts构建Web应用程序，通常要用到以下构件（见图1）：

(1) JSP页面。它可利用Struts html标志库简化开发。

(2) ActionServlet控制器。它根据配置文件struts-config.xml，将接收到的用户HTTP请求进行转发。

(3) Struts ActionForm类。它是一种同JSP页面上的Form域属性自动相关联的JavaBean. ActionServlet将其实例化，并在其中填入Form域中的数据，将其存入request或session中。

(4) Struts Action类。Action子类的Perform()方法中包含应用程序的商业逻辑，执行相应的操作，并在request或session中存入JSP所需的数据。Perform()返回一个ActionForward——一个Struts类，被用来表示提供应答的JSP页面。

(5) ActionMapping类。它表示客户请求中使用的URL和ActionForm以及Action子类之间的映射，所有的映射关系都定义在struts-config.xml文件中。

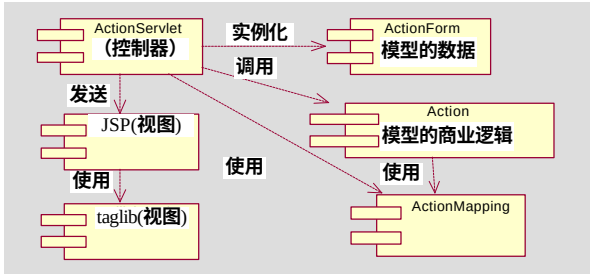

图1 与MVC模式相关的Struts组件

使用Struts框架可以将商业逻辑、控制和数据显示模块区分开来，对其中一个模块的改动不会影响到其他模块。这样在C/S系统向B/S迁移时，使用Struts框架，就可对迁移工作进行划分，使其能分步骤地有效展开。而且使用这种框架，许多代码已经生成并通过测试，可以极大地简化迁移工作。

基金项目：“863”计划课题“基于Internet以构件库为核心的软件平台”资助项目（2001AA1100241）
作者简介：王心路（1978—），男，硕士生，研究方向：软件工程；赵文耘，教授；余 春，硕士生
收稿日期：2003-05-30 E-mail：wangxinlu@netease.com

1.3 Web Service服务

Web Service的基本概念就是使用一个标准的输出接口来定义实现程序代码提供的功能，以便让外界可以通过这个标准的输出接口来调用，这个标准的输出接口就是WSDL (Web Service Description Language). WSDL是一个由XML组成的文件，这个文件内容描述了实现程序代码对外提供的函数原型，也就是各种可供调用的函数名称以及参数信息。在实现程序代码提供了WSDL之后，客户端就可以通过WSDL来调用实现程序代码提供的服务。在调用过程中客户端必须使用SOAP（Simple Object Access Protocol）来封装调用信息和返回结果。SOAP使用了XML作为封装和交换信息的标准，以HTTP 作为第一个实现的通信协议。这些都是现在所有平台和操作系统已经接受的标准，而且SOAP几乎已经被所有的实现语言所支持。这样各种不同的客户端可以使用SOAP标准调用和使用各种不同的后端平台提供的信息。

在迁移过程中通过对原有C/S架构下的代码做少量修改，使其能够提供Web Service服务，这样Web服务器就可使用SOAP来调用这些Web Service，而无须重新编写一整套业务逻辑。图2描述了在B/S系统中使用Web Service的结构。

图2 B/S系统中使用Web Service的结构

2 基于Struts架构的系统迁移工作

2.1 迁移步骤

在C/S系统向B/S迁移时，根据Struts框架，对迁移工作进行了划分，图3描述了具体的迁移步骤。

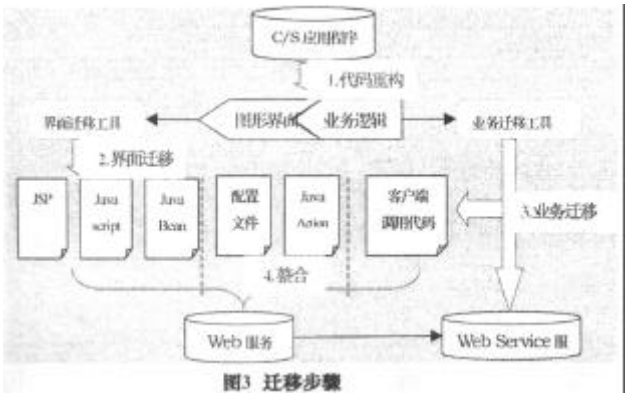

图3 迁移步骤

从图3中可以看到整个迁移工作将分4步展开。

(1) 代码重构。将原C/S系统进行划分，使图形界面和业务逻辑分离，并调整业务逻辑函数模块，使得提供的服务满足Web调用的需要。

(2) 界面迁移。使用工具将C/S图形界面转化成Web页面（包括JSP、Java Script和JavaBean（ActionForm的子类）文件）。

(3) 业务迁移。使用工具将原有的代码转变成Web Service 服务，并产生客户端调用的Java代码供Web服务使用。

(4) 整合。通过添加Struts-config.xml配置文件、Java文件(Action的子类）完成从Web页面到Web Service的调用。

其中(2)、(3)两步可以同时进行，互不干扰，这两步也是迁移的重点，将在下面通过一个例子详细介绍。

2.2 图形界面的迁移

这一步的迁移工作又可以分为两步:信息获取和信息迁移。信息获取即通过分析现有的C/S系统代码来获取用户界面的控件信息。这一步根据编程语言的不同，在实际处理中有很大的差异。以Delphi编写的程序为例，一个Delphi图形界面的程序通常包含两个文件：.DFM文件和.PAS文件。其中.DFM文件是描述窗体及其组件属性的二进制文件。直接分析.DFM文件获取控件信息的方法比较简单，但存在一个问题：运行过程中动态添加或改变的控件信息无法获得。实际使用一种“注入”技术，即在原程序中“注入”获取控件信息的代码（封装成一个TComInfor控件），重新编译运行，这样做可以获取动态组件的信息。图4显示了实际运行时的状况。

图4 实际运行的状况

所有控件信息采用两种描述：仿DFM文件和XML文件。使用XML文件是为了下一步信息迁移的方便，同时也为其他编程语言的控件提供了一个相同的接口。在具体的信息收集的过程中,还有一些控件匹配的细节需要注意,包括：

(1) 控件的粗化。如Delphi中的按钮有Button、BitBtn、SpeedButton等，而Web页面中只有Button一种控件，在收集时可以从它们共同的父类来获取信息。

(2) 控件的细化。在Delphi中文本输入框Edit是不分明文还是密码的，而Web页面中是通过两个不同的控件：Text和Password来实现的，在收集时要根据Edit的PasswordChar属性加以区分。

(3) 控件的转化。有些Delphi中的控件，在Web页面中并没有对应的控件，如表格Grid需要通过Table来实现。

信息迁移即将以获取的控件信息，通过XSLT转变成最终的Web页面，以及处理客户端事件的JavaScript文件和作为页面信息载体的JavaBean文件（Struts ActionForm的子类）。如图5所示。实际处理时还有一些细节需要注意：

(1) 事件的划分。Web应用程序的事件可以分为两类：由客户端处理的JavaScript事件和有服务器处理的Web事件，在实际迁移时需要区分。

(2) Form域的选定。即为相关控件指定一个Form域，并指定它的提交和重置操作，这一步也需要手工完成。

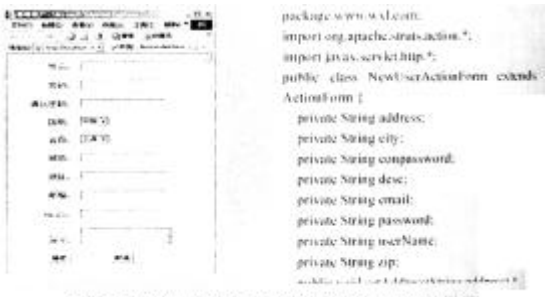

图5 迁移之后的Web页面和部分Java Bean代码

2.3 业务逻辑的迁移

通过改写原有Delphi代码，为其增加Web Service 功能，使不同客户端如Servlet可跨平台调用业务逻辑的处理。先为每一个类作改写并添加一个接口，用InvRegistry.RegisterInvokableClass和InvRegistry.RegisterInterface方法注

册类和接口。下面是通过一段代码描述了所需要做的修改（斜体加粗的部分）。

```
unit AddNewUserImpl;
interface
uses InvokeRegistry, Types, XSBuiltIns, AddNewUserIntf;
type
TAddNewUser = class(TInvokableClass, IAddNewUser)
Public function AddNewUser(user,pword,country,province,city,
address,zip,email,desc:String):Boolean; stdcall;
end;
implementation
Public function AddNewUser(user,pword,country,province,city,
address,zip,email,desc:String):Boolean; stdcall;
begin
…
end;
initialization
InvRegistry.RegisterInvokableClass(TAddNewUser);
end.
```

新添加的接口文件：

```
unit AddNewUserIntf;
interface
uses InvokeRegistry, Types, XSBuiltIns;
type
IAddNewUser = interface(IInvokable) ['{5B962D57-C112-4D3C-
B0BD-29CCD34BDDD0}']
Public function AddNewUser(user,pword,country,province,city,
address,zip,email,desc:String):Boolean stdcall;
end;
implementation
initialization InvRegistry.RegisterInterface(TypeInfo(IAddNew
User));
end.
```

然后使用Delphi的IDE生成一个Web Service程序框架，将生成的类和接口文件加入项目中，调试编译通过后，将产生的Web Service程序分发到Web Server的虚拟目录下，Web程序就可以用Java编写的客户端存根代码来跨平台的调用它。使用Apache AXIS工具包（可以从shttp://xml.apache.org/axis下载）的wsdl2java来完成从wsdl文件生成客户端代码的工作（包括接口文件AddNewUser.java、代码存根AddNewUserbindingStub.java、用于获取存根实例的工厂类AddNewUserServiceLocator.java）.

2.4 整合

主要通过添加Struts-config.xml配置文件，将前面生成的各项成果包括JSP页面、Java Script文件、Java Bean和Web Service客户端代码整合成完整的B/S系统。

首先，给JSP页面中的Form域添加Web事件：

```
<form name="form1" method="post" action="/addNewUserAction.
do">
```

实际执行的Web事件，是根据Struts-config.xml文件中的事件映射转发的：

```
<action-mappings>
<action path="/addNewUserAction"
type=" AddNewUserAction"
name="newUserActionForm"
scope="request" />
</action-mappings>
```

根据配置定义，/addNewUserActtion.do事件将会去调用AddNewUserAction.java这个Action类的子类，并且Form域的信息是通过newUserActionForm.java这个ActionForm类的子类来传递的。在AddNewUserAction类的Perform方法中，通过调用Web Service客户端代码，将信息传递给实际的Web Service服务，由它来完成最终的业务逻辑。下面是AddNewUserAction类的细节：

```
public class AddNewUserAction extends Action {
public ActionForward perform(ActionMapping mapping,
ActionForm actionform, HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse) {
newUserActionForm form = (newUserActionForm) actionForm;
try {
 AddNewUserserviceLocator sl
= new AddNewUserserviceLocator ();
AddNewUser a = sl.getAddNewUserPort();
if(a. AddNewUser( (form.userName,form.password,form.country,
form.province,form.city,form.zip,form.address,form.desc)) {
return servlet.findForward(Login);  }
else { throw new LoginException("Wrong User Information!");}
}
catch (Exception ex) {throw ex;
 }
```

其中粗体部分为访问Web Service服务的代码，在具体应用中还有一些细节需要注意：访问Web Service的客户端代码本身并不具有线程安全，必要时要加上同步机制；由于网络的不稳定性，所有与事务相关的机制，不应该放在Web程序端，而最好包含统一在Web Service调用中。

2.5 部署

通过上面步骤产生的B/S系统，由于反映用户界面的Web程序和反映业务逻辑的Web Service是严格分离的，因此部署是相当方便和灵活的。出于效率考虑，可以将Web程序和Web Service部署在同一台服务器上；如果更多考虑安全性的问题，可以将Web Service部署在不同的服务器上，甚至是严格的防火墙之后（SOAP调用可以通过防火墙）；对于大型的系统，可以使用集群技术，用多台服务器来实现负载平衡。

3 结论

使用本文介绍的基于Struts架构和Web Service的软件迁移技术，能够很好地完成C/S架构向B/S的迁移，使用了大量的自动化工具，不仅简化了工作，还提高了系统的质量；使用的Jakarta Struts框架，使得业务逻辑和界面显示完美分离；使用Web Service技术可以保证业务逻辑和Web程序的无缝连接。Struts架构和Web Service这两种技术将推动迁移技术的不断发展。

参考文献

1 [美] Varhol P. Applying the MVC Design Pattern Using Struts. Java World Web Site, 2002

2 [美] Kurniawan B. Develop Faster With Tag Libraries. Java Pro WebSite, 2003

3 [美] Duffey K, Goyal V, Husted T. Professional JSP Site Design. Wrox, 2002-07

4 [美] Hendricks M, Galbraith B. Professional Java Web Service. Wrox , 2002-10

5 李 维. Delphi6 Kylix2 SOAP/Web Service 程序设计篇. 北京: 机械工业出版社, 2002-01