基于AJAX的Web应用构件组装技术及工具

郑迪文 沈立炜 彭 鑫 赵文耘
(复旦大学软件学院 上海 201203) (上海市数据科学重点实验室(复旦大学) 上海 201203)

摘 要 基于构件的软件开发方式能够有效提高Web应用的开发效率,它所涵盖的构件组装技术涉及到Web应用的前端页面与后端业务逻辑或第三方服务之间的组合。在分析Web应用的构件类型及其组装方式的基础上,提出了一套基于AJAX的Web应用构件组装技术,该技术尤其关注于前端页面构件与后端业务构件以及Web Service构件之间的自动化组装,包含两种具体的组装实现模式,即采用jQuery调用Servlet的实现模式以及采用DWR技术的实现模式,它们为页面构件提供其与服务端构件交互的能力。另外,这两种组装模式已分别实现为两套在线的Web应用构件组装工具,均支持用户通过图形化的方式定义构件的连接关系,并根据不同模式自动组合构件实体单元来生成Web应用系统。以一个实验性选课网站作为应用开发实例,以验证技术与工具的有用性。

关键词 构件组装技术,Web应用,AJAX
中图法分类号 TP31 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.11.030

Component Composition Technology and Tool Based on AJAX for Web Application

ZHENG Di-wen SHEN Li-wei PENG Xin ZHAO Wen-yun
(School of Software, Fudan University, Shanghai 201203, China)
(Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China)

Abstract For Web applications, component-based software development is also a way to improve the efficiency of development, whose solution involves the combination of the front-end and the back-end or third-party services. We proposed a component composition technology based on AJAX for Web application, based on the analysis on the component types and its composition pattern of the Web application. In this technology, we proposed two different solutions for combination of different component types. One uses jQuery to invoke Servlet while the other uses DWR technology to enable interaction between front-end page component and back-end business component or Web Service component. In addition, based on the proposed method, this paper implemented two online Web application component composition tools through two implementation styles, which allow users to define the composition details through graphical user interface and automatically complete the component composition process to generate Web applications. This paper used an experimental course selection website as an application development example, to verify the usefulness of the technology and the tool.

Keywords Component composition technique, Web application, AJAX

1 引言

在互联网技术蓬勃发展的现在,Web应用已和我们的生活息息相关,小到个人博客、大到社交网站及电子商务网站等,都给我们的生活带来了极大的便利。人们对Web应用的需求愈发多样化,这使得Web应用的复杂性越来越高,从而导致独立开发Web应用的难度及成本大幅增加。

软件复用能够有效提高软件系统的开发效率,它意味着重复使用已有的开发制品而不要求每个软件产品均从头开始开发[1]。在软件复用的背景下,基于构件的软件开发[2,3]已经得到广泛共识。与工业界的产品制造类比,一个软件系统也可以由一组构件通过预定义的方式组合而成。构件是具有独立功能,且具有明确接口的程序实体,一般可由开发组织内部或第三方组织提供。在使用过程中,开发者采用黑盒的方式基于构件接口说明正确、合理地复用构件,而不必关心构件的内部实现。对于Web应用而言,基于构件的软件开发也能够成为提高其开发效率的一种途径[4]。由于Web应用自身的特殊性,其构件组合方案将涉及到应用的前端页面与后端业务逻辑或服务之间的结合。

AJAX技术在Web应用中已被广泛使用,它具有提高系统性能、支持页面内局部刷新、使用简便等优势。另外,许多Web应用要求在不跳转页面的情况下即能反映出前端页面

到稿日期:2013-09-20 返修日期:2013-11-15 本文受国家"863"高技术研究发展计划项目基金(2012AA011202)资助。
郑迪文(1992-),女,硕士生,主要研究领域为软件产品线与开发平台,E-mail:09302010089@fudan.edu.cn;沈立炜(1982-),男,博士,讲师,主要研究领域为构件化软件开发;彭 鑫(1979-),男,博士,副教授,CCF会员,主要研究领域为软件再工程、自适应软件系统;赵文耘(1964-),男,硕士,教授,CCF会员,主要研究领域为软件复用、电子商务。

调用后台业务的结果。因此，本文针对这一 Web 应用开发的实际需求，提出了一套基于 AJAX 的 Web 应用构件组装技术。该技术以 Web 应用的构件类型分类为基础，提出了不同类型构件之间的概念层次上的组合方式，尤其关注于前端页面构件与后端业务构件或 Web Service 构件之间的自动化组装。在组装实现方面，该技术包括两种具体的实现模式：一种是采用 jQuery 调用 Servlet 的实现模式；另一种是采用 DWR (Direct Web Remoting)技术[5]的模式。这两种模式均为页面构件提供了其与服务端构件交互的能力。另外，这两种组装模式已被分别实现为两套在线的 Web 应用构件组装工具，这两个工具均支持用户通过图形化的方式定义构件的连接关系，并根据不同模式自动组合构件实体单元，生成 Web 应用系统。最后，本文以一个实验性选课网站作为应用开发实例，验证本文所提技术与工具的有用性。

本文第 2 节主要介绍基于 AJAX 的相关技术的背景知识；第 3 节介绍 Web 应用的构件分类以及在此分类基础上的构件组装方式；第 4 节是本文重点，详细描述 Web 应用的两种不同的构件组装实现模式；第 5 节是对组装工具的描述，以及案例介绍与讨论；最后总结全文。

2 背景知识

AJAX(Asynchronous JavaScript and XML，异步 JavaScript 和 XML)是一种创建交互式网页应用的网页开发技术。AJAX 的异步沟通模式改进了客户端和服务器端的协同工作方式，即并非所有用户操作都会被立刻传递给服务器，而是由 AJAX 引擎处理部分数据直到必须从服务器端获取数据。同时，AJAX 引擎在请求服务器消息时客户端界面可以保持不变，一旦服务器端的消息返回给 AJAX 引擎客户端页面才会被局部刷新。显然，在这种方式下，服务器负担被大大减少，而且用户不用为了获取数据而时常处于页面跳转等待中，不连贯的用户体验被大大优化。

在使用 AJAX 时，经常会利用成熟的库和框架来简化使用过程。其中一种是轻量级的 Javascript 库 jQuery，另一种则是针对 Java 实现 AJAX 功能的框架 DWR。

jQuery 是一个兼容多浏览器的 Javascript 库，其语法设计可以使开发者更加便捷。例如操作文档对象、选择文档对象模型元素、制作动画效果、事件处理、使用 AJAX 以及其他功能。它兼容 CSS3，还兼容各种浏览器，使得开发者无需关注不同浏览器在 AJAX 使用方式上的差异。

DWR，即 Direct Web Remoting，是一款专门为 Java 和 Javascript 开发的框架，是一种远程过程框架且是一个开源的项目。DWR 实际上是一种成熟的、封装好的 AJAX 框架，它可以让开发人员在不熟悉 AJAX 编程的情况下实现 AJAX 的功能，轻松地将客户端请求与服务器调用联系起来。作为封装好的 AJAX 框架，DWR 可以把服务器端的任何 Java 对象公开为可以通过浏览器中的 Javascript 访问的远程对象，其工作原理类似于允许连接及公开 Web service 的远程方法调用[6]。DWR 使用反射来生成 Javascript 对象，以便 Web 页面能够使用这些对象来访问服务器端。然后，Web 页面只须结合到生成的 Javascript 对象，就可实现 AJAX 技术的功能。DWR 基于 Java 类动态创建 Javascript 代码，允许开发人员通过 Javascript 访问服务器的 Java 资源，如同这些资源在浏览器本地一样。而资源的安全性则可以由开发人员保证，因为只有经过开发人员显式地配置 DWR 框架后，Java 对象的指定方法才能在 Javascript 中允许使用。

DWR 框架使 Web 应用开发工作集中在功能设计上，客户端和服务器之间的异步通信和 Javascript 调试工作的问题得以解决。通过配置 Dwr. xml 把客户端的 Javascript 函数和服务器端的 Java 类联系起来，规定了 Javascript 函数和 Java 类的调用关系；通过设计 Java 类来定义 HTML 页面元素结构和调用方法；通过编写 Javascript 函数实现数据回传并控制动态显示。一般的使用 DWR 技术的 Web 应用功能设计围绕这 3 部分展开，它们之间的关系如图 1 所示[7]。因此，在我们的 Web 应用构件组装方法的实现过程中，除了采用辅以 jQuery 框架的原始的 AJAX 方式外，另一种方式即采用了 DWR 技术。

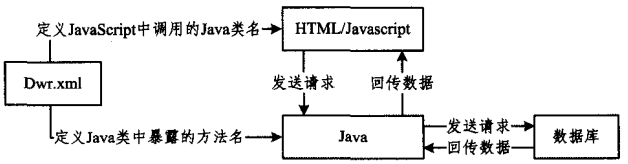

图 1 DWR 应用设计原理

3 Web 应用的构件及其组装方式

3.1 Web 应用构件类型

在对 Web 应用的构件进行分类之前，我们定义构件的组成部分，包括：

(1)构件名称：构件的唯一标识符。

(2)构件描述：对构件功能和非功能需求的描述，是开发者了解所需开发构件的主要途径。

(3)接口：接口是构件之间交互的渠道，可被分为服务接口和请求接口。请求接口是需要被实现的接口，其中的方法需要与其他方法映射才能提供实际的功能；服务接口是具体被实现的接口，可以被其他请求接口以及请求接口中的方法映射，给请求接口提供服务。例如在一个订票网站中，页面构件需要通过请求接口获取票量的实时情况，因此需要后端的业务构件通过其服务接口提供这些信息。

(4)方法：方法是构件之间交互的最小单位，也是构件功能最直接的体现。一个接口可以包含多个方法，一个方法可以包含多个参数。另外，方法可具有返回值，也可以没有。当两个构件需要被组合时，首先需要映射不同构件服务接口和请求接口。接口级的映射实际体现为接口中方法之间以及方法内参数之间的映射。

根据 Web 应用的特点以及从所提供的功能类型角度出发，可将 Web 应用的构件分为 3 类：

(1)页面构件：页面构件主要关注与最终用户之间的交互，包括向用户提供用户界面以及良好的用户体验。页面构件可以是经过打包的 war 文件(Web Archive file，网络应用程序文件，是与平台无关的文件格式，专用于 Web 方面)，其中包含了 html 格式或 jsp 格式的 index 页面以及相应的 css 文件和 js 文件。例如，对于一个订票网站来说，其页面构件就仅包含简单的静态页面以及相应的 js 文件，负责处理前端交互。一般而言，页面构件仅包含请求接口而不包含服务接口。请求接口通过构件的 js 文件部分表示，在 js 中，包含了一组

调用后端服务的方法入口点，这部分的代码示例如图 2 所示。其中，requestMethod 方法由前端页面的用户事件触发。在构件未被组装前，方法内的实现代码无法确定，而是需要通过组装后由工具自动填入。另外，回调函数_callBack 用于处理后端服务返回的结果，其中的代码能够实现对页面部分的局部刷新。

```
//请求方法，用于调用后端代码
function requestMethod(){
  //通过工具自动生成与后端交互的代码
}
//回调函数，需由页面构件开发者预先填写
function requestMethod _callBack(data){
  //回调函数内容，需要预先定义
}
```

图 2　页面构件调用后端逻辑构件的代码

(2)业务构件：业务构件主要关注业务逻辑和数据的处理，但不包含可视化的界面。业务构件通常是经过压缩的 jar 文件，其中包含编译好的 Java class 文件等，一般负责后台逻辑处理和与数据库交互等业务。例如，对于一个订票网站来说，其业务构件可提供处理查票和订票业务等功能。业务构件必须含有服务接口以供其被调用，同时它也可以具有请求接口以调用其他构件的功能。

(3)WebService 构件：WebService 构件主要是以服务的形式向其他构件提供数据处理和业务逻辑处理的功能，一般由第三方部署并发布。只要拥有 WebService 的 WSDL(Web Services Description Language，是一个用来描述 Web 服务和说明如何与 Web 服务通信的 XML 语言)，便可通过 AXIS (Apache EXtensible Interaction System)转化 WSDL 为本地 Java 类的 WebService 客户端，再将编译后的 class 文件压缩成 jar 包后作为 WebService 构件使用。例如，对于一个订票网站来说，其 WebService 构件可包括其他支付系统网站所公开发布的含有支付功能的 WebService 的客户端。在构件组装方面，我们只需要关心 WebService 的服务接口，了解其被访问的方式与能力。

3.2　Web 构件的组装方式

针对 Web 应用的构件类型，我们需要给出页面构件、业务构件以及 WebService 构件间的组装方式，主要有以下 4 种组装类型：

(1)页面构件调用业务构件；

(2)页面构件调用 WebService 构件；

(3)业务构件调用业务构件；

(4)业务构件调用 WebService 构件。

这几类构件之间的组合方式如图 3 所示。页面构件的请求接口需要与业务构件或 WebService 构件的服务接口相绑定，从而能够将用户在应用前端发出的请求传递给服务端，并从服务端得到反馈的结果。在前后端的交互过程中，我们采用 AJAX 技术支持页面的局部刷新。另外，为了实现正确的交互场景，在构件组装之前，开发人员必须预先定义请求接口与服务接口之间的映射，包括接口所包含的方法映射，以及方法下的参数映射。

本文将主要关注第(1)与第(2)这两种组装方式的具体实现技术，并在下一节中详细展开。而第(3)与第(4)两种方式可采用传统的业务构件组装技术，因此不在本文中赘述。

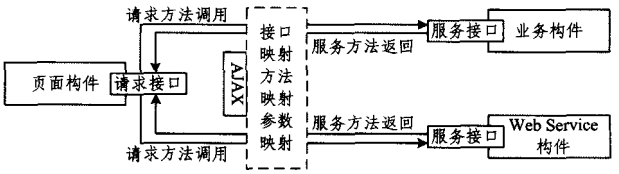

图 3　页面构件连接业务构件、WebService 构件的组装方式

4　构件组装实现技术

基于 Web 构件的组装方式，本文提出一套基于 AJAX 的构件组装技术。在其具体实现方面，该技术涵盖两种组装实现模式，分别为采用 jQuery 调用 Servlet 的模式，以及采用 DWR 技术的模式。

4.1　jQuery 调用 Servlet 实现模式

以页面构件调用业务构件的组装为例，按照前章所述的构件组装方式，在具体的技术实现中，我们需要在页面构件的 js 文件中添加 AJAX 请求，并在两个构件中创建中间代理 Servlet 来实现对业务构件的 Java 方法的调用，如图 4 所示。对于页面构件而言，其请求接口即是 js 文件，而其请求的功能则是由业务构件的 jar 包提供，实现页面构件和业务构件的组合实际上即是需要实现在 js 文件中调用业务构件的服务接口中相应的 Java 方法。为了达成这一目标，我们采用 AJAX 技术，通过在页面构件中发送异步通讯请求至代理 Servlet，再由代理 Servlet 根据接收到的 get 或 post 请求在 Servlet 内部直接调用业务构件中的 Java 方法，再将得到的 Java 方法返回值传递回页面构件，即完成了页面构件的请求接口调用业务构件服务接口的需求。对于组装过程的逻辑实现，此处简要进行介绍。

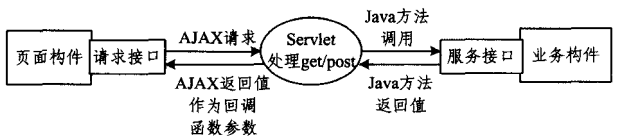

图 4　jQuery 调用 Servlet 实现模式

首先对页面和业务 2 个待组装构件间的所有接口映射建立接口映射表，并遍历该表循环进行以下操作：

a)从当前接口映射中获取构件的 ID、名字和相应的方法映射表。

b)遍历方法映射表，对每个方法映射创建独立的代理 Servlet，其具体过程如下：

更改页面构件中的 WEB-INF/web. xml 配置文件，添加代理 Servlet 的信息。

创建并填写 Servlet 的 Java 文件，并使用“服务接口_服务方法_Servlet”格式命名，其中包含了对所有 post 和 get 方法的处理，并根据方法映射调用相应业务构件中的 Java 方法；另外，为了 AJAX 使用时书写的简便，此处使用了 jQuery 框架的语法。

将 Servlet 的 Java 文件编译后把对应的 class 文件拷贝至 WEB-INF/classes 下。

c)将业务构件的 jar 包拷贝至 WEB-INF/lib 下。

d)更改页面构件的 js 文件：遍历方法映射表，向每个请求方法中添加 AJAX 调用。

在经过 axis 插件将 WebService 的 WSDL 转化为本地 Java 文件(WebService 客户端)并打包成 jar 后,WebService 构件实际上等同于业务构件。在该方式下,页面构件调用 WebService 构件的组装的过程类似,区别仅在于在创建 servlet 的 Java 文件时,由于 WebService 桩文件的特殊性,调用 WebService 构件中的 Java 方法时语法需稍作修改,不同于连接业务构件时的直接调用服务方法,需要在代码中根据 WebService 构件的定义实例化一个 proxy,并由 proxy 代为进行对服务方法的调用。

4.2 采用 DWR 的实现模式

由于 DWR 对 AJAX 的封装,我们无需再关注中间 Servlet 的创建过程,但需要对 DWR 进行相应配置以直接在 js 文件中调用业务构件或 WebService 构件中的 Java 方法。此处同样以页面构件调用业务构件的组装为例,如图 5 所示,由于 DWR 对 AJAX 进行了封装,因此一旦对 DWR 的配置完成便无需关心请求接口中的方法是如何实现调用服务接口中的方法的,而是直接在页面构件的请求接口中直接使用对业务构件的 Java 方法调用。这个过程的实现实际上是由 DWR 根据配置信息,根据反射生成对象或者由 Spring 注入对象,把客户端的 Javascript 对象转换成服务器端的 Java 对象,然后调用 Java 方法,将处理结果即返回值返回给浏览器,最后将返回值作为参数传给回调函数。

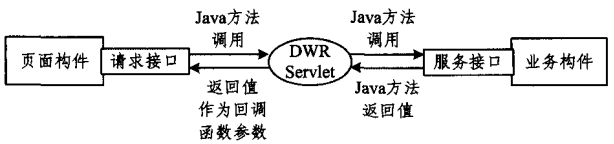

图 5 采用 DWR 技术的实现模式

对于组装过程的逻辑实现,此处简要进行介绍:

a)从当前接口映射中获取构件的 ID、名字和相应的方法映射表。

b)将工具中的 DWR 包和业务构件的 jar 包复制到页面构件的 Web-INF/lib 下。

c)配置 web. xml,添加应用 DWR 的信息。

d)配置 dwr. xml:遍历方法映射表,记录每个服务方法的信息,包括其所在的 Java 类名。

e)在页面构件的 html 文件中添加使用 DWR 所必须的 js 文件引用。更改页面构件的 js 文件添加相关调用信息:遍历方法映射表,向每个请求方法中添加 DWR 调用。默认 js 中存在添加标记,映射的类名为"dwr_服务接口名",方法名为服务方法。

在该方式下,页面构件调用 WebService 构件的组装的过程类似,但同样由于 WebService 桩文件的特殊性,需要事先创建及编译接口 connector 类放入 WEB-INF/classes 下,用于连接 WebService 桩文件。

4.3 两种实现模式的比较

在实现上,这两种模式均基于 AJAX 技术来实现页面构件和业务构件、WebService 构件之间的交互,仅在具体使用的框架上有所差异。

两者在 Web 应用组装中的共同点大体包括:都需要更改 web. xml 中的配置;都需要将业务构件或 WebService 构件移入页面构件的类库下;都需要在业务构件请求接口即 js 文件中插入相应的调用代码。

相比而言,这两种模式在组装过程中的不同点则有:具体的配置方式不同;在 js 文件中插入调用代码的方式不同(前者直接填充 AJAX 调用,而后者直接对后台 Java 方法进行调用)。

具体而言,DWR 的实现模式侧重于与 J2EE 应用程序的整合,通过该技术可以直接在 Javascript 中调用 Java 类,降低了组装的学习成本。同时,DWR 模式也有助于隐藏 AJAX 中的过程实现,并加强数据传输的保密性。然而,在这种方式下组装成品即暴露了后台方法,且只能使用 Java 语言作为业务构件、WebService 构件的开发语言,在效率上亦要比前一种模式低。除此之外,虽然所生产出的成品中 AJAX 部分的代码量较少,但仍然需要对 DWR 进行较多的配置。

而在 jQuery 调用 Servlet 的实现模式中,由于轻量级的 jQuery 库支持开发人员编写更简单、功能更强大的 Javascript 代码,因此这种模式的组装成品相较之下更为小巧、在配置方面更为简便,另外 jQuery 对 AJAX 进行了基本封装,避免了侵入性。虽然在这种模式下由于将 Servlet 作为代理不会暴露后台方法,但组装成品中会存在较多显式的 AJAX 调用代码。

5 Web 应用的构件组装工具及实例

5.1 Web 应用的构件组装工具架构及其实现

基于 Web 应用的构件组装技术,我们开发了两套分别针对两种不同实现模式的组装工具。对于这两个工具而言,其架构相同,区别点在于工具应用于遵循不同实现规范的页面构件,另外工具自动生成的组装代码的方式也有所不同。这两个工具的统一架构如图 6 所示,其中包括前端网站、工具引擎、构件库和数据库等模块。

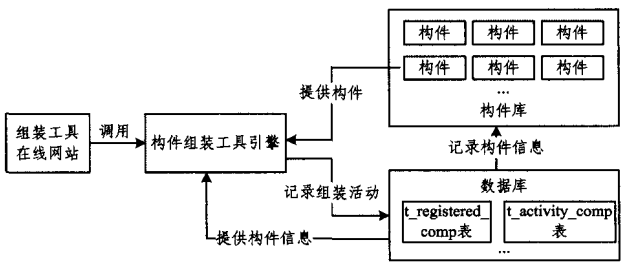

图 6 构件组装工具架构

对于工具的使用者而言,其能接触到的只有在线工具的在线网站,而该网站前端页面是利用开源框架 Flex 制作的。Flex 用于构建具有表现力的 Web 应用程序,这些应用程序利用 Adobe Flash Player 和 Adobe AIR,运行时跨浏览器、桌面和操作系统实现一致的部署。

在前台网站之后,实际的组装是调用组装工具引擎完成的,这一部分的具体实现技术已在上一节有所介绍。另外由于有两种不同的组装实现方式,故实际有两种组装引擎,但这两种方式对于工具整体而言仅在组装引擎、页面构件开发部分有所差别,其他部分的架构完全相同。

在工具架构中,组装工具引擎是一个重要组成部分。除去代码部分,组装工具的实际部署是以 Web 应用的形式发布的,其下自然包含了 WebContent 文件夹。其中内置了使用 Flex 制作的 swf 文件(shock wave flash,一种 flash 的专用格

式)以作前台展示,以及在 composition-space\reuseable-component 文件夹下存放通用构件以供组装复用的通用构件库,以及在 composition-space\connector-tool 文件夹下保存的供组装工具使用的工具文件如 jQuery. js 文件,另外开辟了 composition-space\work-space 专用空间供组装时使用并在组装完成后清空该文件夹。

在数据库中,储存了有关构件信息、组装活动的内容,以及与在此讨论的构件组装工具密切相关的用于储存通用构件的描述信息的 t_registered_comp 表和用于记录组装活动的 t_activity_comp 表。

我们以采用 jQuery 调用 Servlet 的实现模式为例,与其对应的组装工具界面示意图如图 7 所示。其中,左侧为操作栏和构件栏,右侧为构件组装活动的编辑面板。在构件开发完成后即可通过该可视化界面完成接口映射、方法映射、参数映射、构件组装及部署的操作。

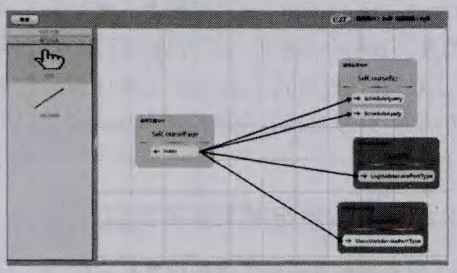

图 7　组装工具在线网站界面

5.2　应用开发示例

为了测试在线组装工具的有效性,我们实验性地开发了一个选课系统 Web 应用,该应用面向学生提供基本的课程推荐、输入代码选课并显示课程表的功能。由于仅需测试页面构件调用业务构件、页面构件调用 WebService 构件两种组装类型,因此根据"高内聚低耦合"的理念将功能模块进行构件划分,设计该选课应用的架构如图 8 所示。

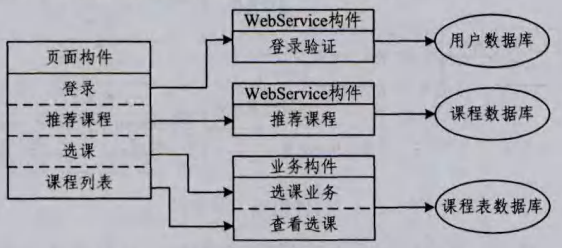

图 8　选课网站架构

根据应用架构,设计应用使用的后台数据库包含 user 表保存用户信息、course 表保存课程信息、schedule 表保存用户的课程表信息。

在构件开发方面,根据该选课应用的构件划分,我们需要开发负责前端交互的页面构件、负责选课业务将所选课程记录至课程表及获取课程表的业务构件、负责登录验证控制的 WebService 构件、获取推荐课程的 WebService 构件。

页面构件开发。编写了基本的 jsp 页面、css 文件及相应的 Javascript 文件。在 Javascript 文件中,预留了需要从其他构件中请求数据的接口,在 jQuery 调用 Servlet 的实现模式下,一般形式为在请求方法的空方法体中添加格式为"//ajax_请求方法名"的标签或以标准格式书写该请求方法的方法头,则在组装进行时会自动定位至此处向其中加入 AJAX 调用;或者在 DWR 模式下进行组装时,则留有"//dwr_请求方法名"的标签,待组装时在此处插入对其他构件中的 Java 方法的调用。另外,由于是以 AJAX 方式实现的,故请求得到的数据应由回调函数进行处理,这里默认所有回调函数以"请求方法名称_callBack"格式命名。

业务构件开发。在选课业务的设计中,两部分功能分别由 ScheduleApply 接口和 ScheduleQuery 接口实现,两者都是在其提供的相应服务方法中连接数据库进行查找、更新操作以获取、更改数据。

WebService 构件开发。在 WebService 构件开发中,我们仅需获得对应 WebService 的 WSDL 以生成桩文件。在 WebService 部署成功后,在 Eclipse 中使用自带的 axis 插件选择生成 WebService 客户端,即可通过输入 WSDL 地址的方式自动生成 Java Proxy,获得 WebService 的桩文件。

在构件开发完成并将这些构件引入至组装工具后,在工具内完成该选课系统应用的组装工作(见图 7),在组装完成后组装成品相较原先构件有如下变化:

(1)当采用 jQuery 调用 Servlet 作为实现模式时,在应用的类空间下含有编译好的 Servlet 文件,其负责调用业务构件 jar 包中的方法,这部分 Servlet 需要被定义在 web. xml 中。另外,页面构件的请求接口 js 文件中被插入了 AJAX 方法的交互代码。这两部分自动生成的代码如图 9 所示。

```
//在应用的 web. xml 中自动增加的 servlet 定义
〈servlet〉
  〈servlet-name〉ScheduleQuery_getUserCourse_Servlet
  〈/servlet-name〉
  〈servlet-class〉cn. edu. fudan. servlet. ScheduleQuery _ getUser-
  Course_Servlet〈/servlet-class〉
〈/servlet〉
〈servlet-mapping〉
  〈servlet-name〉ScheduleQuery_getUserCourse_Servlet
  〈/servlet-name〉
  〈url-pattern〉/servlet/ScheduleQuery_getUserCourse_Servlet
  〈/url-pattern〉
〈/servlet-mapping〉
//页面构件 js 中插入 AJAX 方法,使用 jQuery 的规范
function getUserCourse(username){
  $. ajax({
    url:'servlet/ScheduleQuery_getUserCourse_Servlet',
    type:'POST',
    data:{username:username},
    success:function(data){
      getUserCourse_callBack(data);
    }
  });
}
```

图 9　采用 jQuery 调用 Servlet 模式下的新增代码

(2)当采用基于 DWR 技术作为实现模式时,Web 应用中需要 DWR 所需的 dwr. xml 配置文件。另外,页面构件的请求接口 js 文件中也被插入了基于 DWR 对业务构件方法进行调用的代码。这两部分自动生成代码如图 10 所示。

(下转第 191 页)

数明显减少，以上现象得到缓解，如表 1 中 Flower 在低带宽下码率下降尤为明显。

因此，本文的方法是在不降低图像质量的前提下，降低码流，进而降低延迟。

结束语 本文首先对现有编码标准的固有延迟进行了分析，以标准清晰度的视频为例，编解码延迟要在 260ms 以上，无法满足低延迟系统的需求。因而，本文从采集与编码两个环节来降低延迟，首先，采集端将每帧图像以条带为单位进行采集，条带的大小与视频分辨率和延迟要求相关；然后，以条带为单位进行编码，为了提高预测精度，进一步降低延迟，将每个条带划分成 4 个子图，以其中一个子图为预测单元进行帧内预测，这样可以有效提升预测的精度。本文提出的基于条带的低延迟编解码结构能有效降低编解码系统的固有延迟，对于标准清晰度视频，可以降低到 150ms。

参 考 文 献

[1] 郑翔，叶志远，周秉锋. JVT 草案中的核心技术综述[J]. 软件学报，2004，15(1)：58-68

[2] 毕厚杰. 新一代视频压缩编码标准-H. 264/AVC[M]. 北京：人民邮电出版社，2005：78-89

[3] Wiegand T，Sullivan G J，Bjontegaard G，et al. Overview of the H. 264/AVC Video Coding Standard[J]. IEEE Transactions on Circuits and Systems for Video Technology，2003，13(7)：560-576

[4] Sullivan G J，Ohm J-R，Han W-J，et al. Overview of the High Efficiency Video Coding (HEVC) Standard[J]. IEEE Transactions on Circuits and Systems for Video Technology，2012，22(12)：1649-1668

[5] Ohm J-R，Sullivan G J，Schwarz H，et al. Comparison of the Coding Efficiency of Video Coding Standards-Including High Efficiency Video Coding (HEVC)[J]. IEEE Transactions on Circuits and Systems for Video Technology，2012，22(12)：1669-1684

[6] Bossen F，Bross B，Sühring K，et al. HEVC Complexity and Implementation Analysis[J]. IEEE Transactions on Circuits and Systems for Video Technology，2012，22(12)：1685-1672

[7] Lin P X，Rahardja S，et al. Fast Intra Mode Decision Algorithm for H. 264/AVC Video Coding[J]. International Journal of Computer Science and Network Security，2007，7(1)：356-361

[8] Li Bo，Li Wei，Tu Ya-ming. A fast block matching algorithm using smooth motion vector filed adaptive search technique[J]. Journal of Computer Science and Technology，2003，18(1)：14-21

[9] Yang Zhi，Cai Hua，Li Jiang. A framework for fine-granular computational-complexity scalable motion estimation[C]// IEEE International symposium on Circuits and Systems (ISCAS). 2005：5473-5476

[10] Yu Dan-chu，Huang Sung-cheng. Study of Reprojection Methods in Terms of Their Resolution Loss and Sampling Errors[J]. IEEE Transaction on Nuclear Science，1993，40(4)：1174-1178

[11] Rodrigues L，Leandro Borges D，Marcos Gonalves L. A Locally Adaptive Edge-preserving Algorithm for Image Interpolation[C]// XV Brazilian Symposium on Computer Graphics and Image Processing. 2002：300-305

(上接第 156 页)

```
//在应用的 dwr. xml 中自动增加的对类的定义
〈dwr〉
  〈allow〉
    〈create creator="new" javascript="ScheduleQuery"〉
      〈param name="class"
             value="cn. edu. fudan. tr23. ScheduleQuery"/〉
    〈/create〉
  〈/allow〉
〈/dwr〉
//页面构件 js 中插入 DWR 调用，调用业务构件中
  //ScheduleQuery 类的 getUserCourse 方法，取得返回值后调用
  //getUserCourse_callBack 回调函数
  function getUserCourse(username){
    ScheduleQuery. getUserCourse(username, getUserCourse_callBack);
}
```

图 10 采用基于 DWR 模式下的新增代码

5.3 分析讨论

应用该技术后，Web 应用构件的组装不再需要关注组装细节，一切连接代码的生成、编译等工作都交由组装引擎来完成，保证了实用性和简便性；另外，亦满足了 Web 应用中异步通信、局部刷新的需求。但由于该方法在设计时定义的限制较多，导致适用性较差，例如：当前该技术仅支持页面构件、业务构件和 Web Service 构件的组装，难以满足未来所需，限制了 Web 应用构件开发者的创造性；在本技术实现中，也未对页面构件中亦存在服务接口、提供功能服务给其他页面构件的可能性做出考虑。

结束语 本文基于软件复用的思想为 Web 应用的高效开发提供了一种途径。我们首先对 Web 应用的构件类型以及这些不同类型构件之间的组合方式进行了分析，随后提出了一套基于 AJAX 的 Web 应用构件组装技术。该技术主要包括两种实现模式，1)采用 jQuery 调用 Servlet 的模式，2)采用基于 DWR 技术的模式。使用这两种模式都能够实现前端页面构件与后端业务构件或 Web Service 构件的交互。基于该技术，我们开发了采用不同实现模式的组装工具，并使用一个选课网站系统作为实例描述了组装的过程与结果。由于本文所提出的技术仍然具有局限性，因此，我们计划在以后的工作中涵盖更多类型的构件实体及其组装类型。同时，逐步改进工具使其更具有可用性。

参 考 文 献

[1] Jacobson I，Griss M，Jonsson P. Software reuse：architecture，process and organization for business success[M]. ACM Press/Addison-Wesley Publishing Co.，1997

[2] Fuqing Y，Hong M，Keqin L. Software Reuse and Software Component Technology [J]. Acta Electronica Sinica，1999，27(2)

[3] George T H，William T C. Component-based software engineering：putting the pieces together[M]. Reading：Addison-Wesley，2001

[4] 陈霄，吴毅坚，彭鑫，等. 采用构件组装技术协同开发 Web 应用的方法[J]. 计算机科学与探索，2013，7(2)：114-125

[5] Direct Web Remoting [OL]. http://directwebremoting. org/dwr/index. html. Accessed：5 June 2013

[6] Serrano N，Aroztegi J P. Ajax frameworks in interactive Web apps[J]. Software，IEEE，2007，24(5)：12-14

[7] 张栗，张凤元，危胜军. 基于 DWR 框架的 Web 应用的设计与实现[J]. 计算机技术与发展，2008，18(8)：84-87