一种柔性 Web 展现框架模型①

刘一田, 刘士进

(国网电力科学研究院, 南京 211100)

摘 要: 在 Web2.0 时代, 越来越多的网站采用了动态脚本的方式和用户进行交互, 大量客户端脚本的应用, 造成了代码的可适应性、可维护性、可扩展性较差, 无法兼容各种主流浏览器, 页面之间的跳转仍然较多, 资源的加载没有规则等问题, 影响了应用性能和用户体验. 提出了一种柔性 Web 展现框架模型 FWF, 构造了符合 AJAX+MVC 模式的框架模型, 定义了组件模型并通过策略适配器的驱动及事件机制, 较好解决了软件适应性问题; 对 UI 组件进行面向对象的封装, 实现模型(Model)、视图(View)和控制(Controller)的合理分层, 并通过内置的资源加载规则, 缩短资源加载时间, 从而提升用户应用体验, 通过 OSGI 框架的模块扩展机制实现了 Web 组件的可扩展. 此外, 通过原型实例实验证明了框架的柔性和性能.

关键词: 柔性; 展现框架; 面向对象

Flexible Web Framework Model

LIU Yi-Tian, LIU Shi-Jin

(State Grid Electric Power Research Institute, Nanjing 210003, China)

Abstract: In the Web2.0 era, more sites are using dynamic scripting approach and user interaction, a large number of client-side scripting applications, resulting in a code of adaptability, maintainability, scalability is poor, and not compatible with all major browsers, jump between pages still more, resource loading has no rules and other issues affecting the application performance and user experience. Presents a flexible framework model FWF, constructed accord AJAX + MVC pattern framework model that defines the component model, and through a policy driven and event adapter mechanism solves the problem of software adaptation. package UI components in object-oriented pattern, the implementation of model (Model), view (View) and control (Controller) are layered rationally, and through the built-in resource loading rules, shorten resource loading time, thereby enhancing the user application experience through OSGI framework's module extension mechanism to achieve a scalable Web components. In addition, through a framework prototype instance proved the flexibility and performance.

Key words: flexible; web framework; object-oriented

柔性展现框架是一个比较新的领域. 软件柔性是指软件行为能符合用户预期, 动态适应软件环境的变化. 例如, 对于长生命周期的大规模软件系统, 软件的环境是动态变化的, 它有可能超出开发阶段的预期, 需要软件具备适应能力; 对于开发阶段未预期的环境及停机维护代价高的系统, 这种环境适应能力更需要在线调整, 本文主要关注于 Web 在线应用的柔性, 这种自适应体现在 Web 应用可预期的一些能力, 包括兼容不同浏览器, 界面元素展现主题的在线动态可配置, 界面元素位置的在线动态可配置, 本地化和国际化的动态可配置, Web 资源的按需加载能力, 以及不可预期的外部环境变化时的动态调整能力、动态组件扩展能力等.

AJAX(Async hronous JavaScript and XML)技术的运用使得在浏览器与服务端交互时, 客户端的 AJAX 请求代替了页面刷新请求, 业务逻辑中对规范

① 收稿时间:2013-07-19;收到修改稿时间:2013-09-09

客户操作的约束放到了客户端, 可以提前验证客户操作的合法性, 提高了向服务器端传递的数据质量. 但是, 大量客户端脚本的应用, 代码的可维护性、可扩展性、各种主流浏览器的兼容性成为了新的挑战, 如何在保证用户体验的同时, 增强代码的可维护性和扩展性, 屏蔽主流浏览器的差异, 亟需一种忽略脚本语言本身的差异性, 让开发人员更好地专注于业务逻辑编码的柔性展现框架. 文献[1]提出了一种以“环境信息显式化、互动方式层次化、体系结构可演化”为特征的环境驱动模型, 本文工作以此为基础, 给出了一种柔性展现框架模型 FWF(Flexible Weblet Framework), 采用功能分层架构对客户端组件进行分类封装, 在组件组装层面引入抽象的“配置变化-适应动作”的策略层, 主要做出了如下一些贡献.

(1) 在框架中定义了上下文组件、展现组件、行为组件、策略适配器. 支持支持软件适应, 上下文组件通过策略连接器的事件驱动控制软件适应过程; 支持软件适应能力的在线调整, 通过 OSGI 的模块生命周期管理器[2]和 FWF 的类继承机制, 提供了运行时在线调整应用组件功能的能力.

(2) 采用柔性管理的脚本组织结构以及面向对象的客户端组件封装, 强调了客户端组件的复用性, 复用范围扩展到功能结构级别, 增强了代码的可扩展性和可维护性. 屏蔽了主流浏览器对脚本与样式解析的差异, 提供统一的应用程序接口. 在同一页面中采用画布分层的方式展现数据, 页面间的数据传递锐减, 降低了因页面跳转对服务器内存及 CPU 资源的消耗, 改良了用户体验. View 层更加专注于界面展现, Controller 层专注于界面元素之间的交互及 AJAX 通信, 更能保证业务操作的原始性; Model 层集中精力处理核心业务逻辑; 减少了服务端页面的频繁跳转.

1　FWF模型原理概述

软件适应性是指软件感知环境变化并据此调整自身行为的过程[3], 环境变化可以是用户预期操作触发的, 也可以是用户非预设的适应动作. 预期操作需要在开发阶段指定, 并提供可配置的接口, 非预设操作需要对软件进行修改, 并提供扩展接口机制保证扩展点可以在线更新.

1.1 FWF 模型组成

FWF 模型从柔性架构层面定义了三种组件类型(如图 1), 上下文组件、展现组件、行为组件. 上下文组件负责定义展现组件和行为组件的上下文环境, 捕捉环境上下文变化并输出相应的上下文事件, 以建立上下文环境模型和触发适应性动作. 展现组件负责接收展现上下文参数并结合 HTML DOM 渲染到浏览器中. 行为组件执行上下文组件发出的环境或配置改变命令或执行展现组件之间的交互行为.

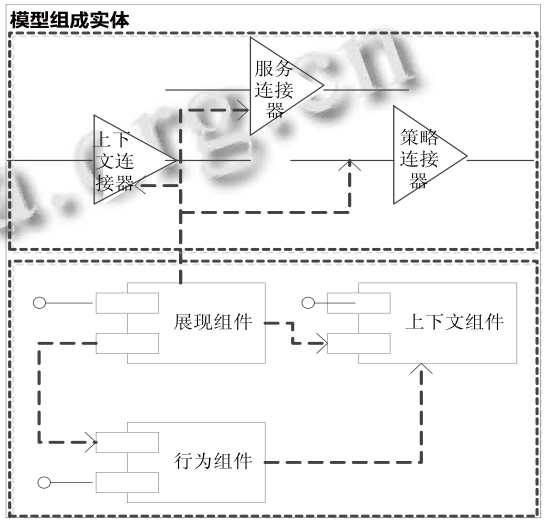

图 1　FWF 模型组成实体

在组件组装定义层面, 采用了 ZIP 格式的压缩包, 其中通过 XML 格式的 FWF 界面描述语言 FWFDL 定义了上下文组件、展现组件, 以及组件之间的关联关系, 并描述了行为组件对应的脚本文件, 脚本文件中定义了行为组件需要执行的行为逻辑.

1.2 FWF 运行时在线调整

在 FWF 展现框架模型中, 上下文组件负责维护运行时上下文模型及其与实际系统的关联, 结合 OSGI 技术及 WEB 容器, FWF 可以(如图 2 所示).

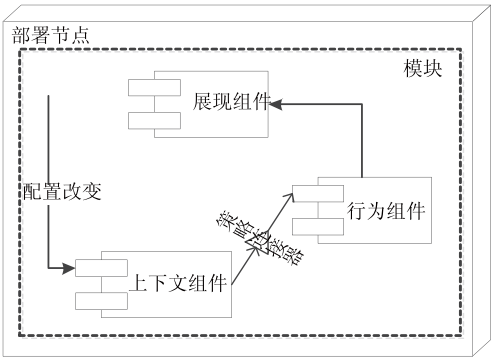

图 2　FWF 模型运行时在线调整

(1) 支持软件适应. 通过获取上下文组件的输出, 上下位组件的逻辑环境模型维护一个当前相关环境状态的镜像[4]. 在此基础上, 上下文组件通过策略连接器的事件驱动控制软件适应过程: 当策略连接器中指定的上下文组件产生上下文事件, 且当前环境状态符合一定条件时, 行为组件方法修改操作将被执行.

(2) 支持软件适应能力的在线调整. 第三方可以对运行时软件结构模型进行修改, 可以单独指定对运行时模块结构的修改动作, 如增删替换展现组件、行为组件、封装决策逻辑的策略连接器等. 模块可以接受修改规约, 修改运行时软件结构模型, 进而通过 OSGI 的模块更新作用到实际运行系统上.

2 FWF展现框架设计

FWF 展现框架从架构层面保证了易用性、扩展性、可配置型、交互响应体验、界面友好. 具备基础的面向对象编程、常用工具类、异常处理(包括超时、语法解析错误、服务端异常、客户端脚本错误)机制, 其中视图组件要具有布局能力、数据渲染能力、事件传播机制、辅助编辑器选择和展现、窗体、提示框及对话框等功能; 上下文组件 Weblet 具备上下文环境配置注册、加载、渲染、扩展、注销等机制; 行为组件根据策略连接器的要求执行上下文环境的命令及视图组件的交互等操作.

图 3 给出了 FWF 的运行时切面, 其中的组件及实体定义如下.

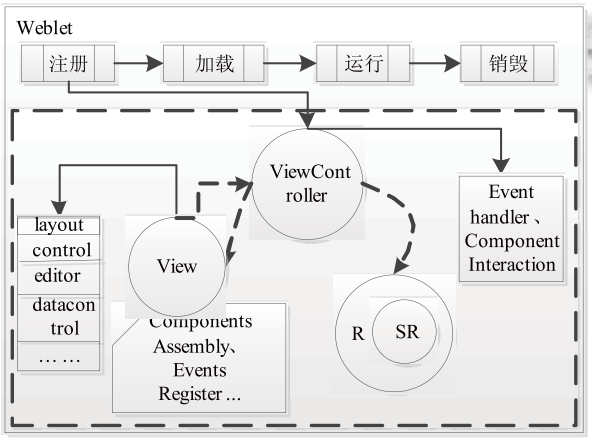

图 3 FWF 运行时切面

定义 1. (上下文组件 Weblet). Weblet 可定义为一个三元组(name, viewport, requires), 其中 name 定义了 Weblet 上下文的唯一名称, viewport 定义了 Weblet 的宿主环境, requires 定义了 Weblet 运行所需的内外部 Weblet 资源, 通过 requires 实现了资源的按需加载能力.

定义 2. (视图组件). 视图组件可定义为一个四元组(name, controls, behaviors, Weblet), 其中 name 定义了视图组件的唯一名称, controls 定义了视图组件的组成部分, 表示视图是由所需控件组装的一个小场景, 视图组件中的 controls 采用面向对象的封装, 支持继承和多态, 其中定义了“$import”关键字引入外部类, “$extend”关键字实现类之间的继承关系等面向对象封装, controls 的显示采用图层分层的样式显示, behaviors 定义了 controls 的行为, Weblet 代表了视图的上下文环境.

规约

```
package.subpackage.CustomClass = function(){
    var me = $extend(MXComponent);
    me.on(eventname) = null;
    me.customMethod= function(){
        var eventParam = {cancel: false }
        me.trigger(eventname, eventParam);
      … …
    };
      … …
      return me.endOfClass(arguments);
};
```

示例

```
mx.customs.datacontrolsBizDataTree = function(){
    var me = $extend(mx.datacontrols.DataTree);
    me.onexpanding = null;

    me.customMethod= function(){
        me.trigger(‘expanding’);
      … …
    };
      … …
      return me.endOfClass(arguments);
};
```

图 4 组件扩展规范和示例

定义 3. (视图控制器组件). 视图控制器组件可定义为一个三元组(name, actions, Weblet), 其中 name 定义了视图控制器组件的唯一名称, actions 定义了视图组件定义的控件行为及组件之间的交互行为, Weblet 代表了视图控制器组件的上下文环境.

定义 4. (资源). 资源分为静态和动态资源两类, 静态资源 Rs 定义了界面渲染所需的脚本、样式、图片等文件, 动态资源 Rd 定义了数据资源模型, 为视图组件中的展现控件提供输入.

FWF 提供了组件类的混淆压缩功能, 将 FWF 框架及二次开发组件发布为独立的混淆压缩脚本文件, 支持在运行环境配置组件为生产模式, 提供更快捷的资源下载方式. 支持按需加载. 通过定义 3 描述的“$import”导入机制以及 Weblet 的类注册机制, 实现资源的按需加载, 缩短了资源下载时间.

3 原型验证与实验

本节给出我们实现的 FWF 模型的容器原型, 并结合在其上开发的实例验证 FWF 模型框架的有效性和性能.

3.1 支持 FWF 模型的容器原型

如图 5 所示, FWF 模型框架运行于 OSGI 服务平台之上, 作为基础模块并对外暴露服务, 支持灵活扩展, 可以支持适应和适应能力的在线调整, 其中适应引擎负责依据策略连接子驱动软件适应过程: 事件服务将上下文事件推给适应引擎, 适应引擎从环境模型读取当前环境状态, 检查指定的条件是否满足, 执行上下文结构模型修改动作或是调用具体的组件方法. 对于软件适应能力的在线调整, 第三方在确定调整的时机和内容后, 书写如图 4 所示规约形式的扩展代码, FWF 模型修改接口可以接受规约, 修改上下文结构模型本地视图, 并在界面中作出快速调整和展现.

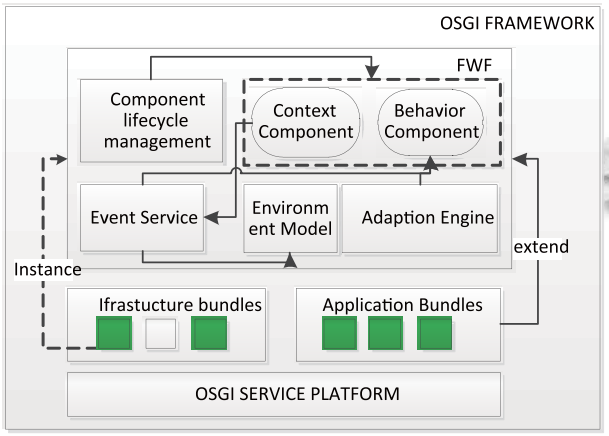

图 5　支持 FWF 模型的容器原型

3.2 实例分析

我们在 OSGI 服务平台之上通过 FWF 展现框架实现了当前主流表现形式的 Web 桌面, 包含了用户认证, 桌面菜单, 桌面快捷方式, 桌面 Widget, 桌面任务栏, 桌面主题设置, 桌面国际化设置等一些基本组件, 如图 6 所示. 该应用具体背景如下: 用户登录进入 Web 桌面后, 根据自身偏好设置桌面主题、菜单快捷方式及桌面 Widget; 当发现所需 Widget 尚未提供时, 要求允许二次开发实现相应 Widget, 在线更新 Widget 组件并支持该组件配置到 Web 桌面.

图 6　Web 桌面

我们通过如下场景来验证 FWF 构件模型对软件适应和适应能力在线调整的支持:

(1) 软件适应. Web 桌面可以在主流浏览器(IE、FIREFOX、CHROME 等)上无差异显示, 支持主题、国际化的动态调整和展现, 支持根据配置调整界面布局. 当登入用户进行主题或国际化等个性化设置时, 通过事件服务将上下文事件推送给适应引擎, 适应引擎从环境模型中读取当前环境状态和配置信息, 检查用户个性化配置是否和当前环境状态相冲突, 在无冲突时读取配置主题或国际化文件, 并通知上下文组件, 触发改变事件以执行环境调整行为.

(2) 软件适应能力的在线调整. 由于未预期的原因(如软件功能新需求或改进等)导致现有功能无法满足. 我们通过图 5 中的扩展继承结构修改规约以新增功能组件或覆写已有功能组件, 并在 OSGI 服务平台上以新增模块的形式动态安装到运行环境中, 不会影响运行环境中的其它模块.

3.3 性能评估

和采用动态服务器技术(JSP、ASPX、PHP)等动态页面相比, 由于需要加载 FWF 展现框架, 性能同比会损失 5%左右. 但是动态服务器技术仅能实现的界面交互效果, 其界面表现和交互能力不如 FWF 展现框架. 以简单门户新闻页面实现为例, 通过 JSP 方式和 FWF 方式分别实现后, 通过 loadrunner 进行压力测试, 性能

差异如表 1 所示.

表 1 JSP 方式与 FWF 方式性能差异比较

并发用户数	JSP 方式平均响应时间（秒）	FWF 方式平均响应时间（秒）	FWF 框架性能下降率（JSP 方式 TPS - FWF 平台 TPS）/ JSP 方式 TPS
50	1.449	1.467	1.23%
100	2.658	2.876	7.58%
200	5.047	5.195	2.85%
300	7.246	7.485	3.20%
FWF 平均性能下降率			3.72%

由于展现框架对数据渲染和编辑的数据控件对性能要求较高, 因此, 我们针对主流展现框架 ExtJS、JQuery easyUI 的树、表格、表单等数据组件, 通过前端性能测试工具 dynaTrace Ajax Edition[5]进行了性能比较, 统计结果以秒为单位, 如表 2、表 3、表 4 所示.

表 2 表单组件性能比较

数据量（字段数）	FWF	EasyUI	ExtJS
25	0.34	0.31	1.8
40	0.51	0.4	2
50	0.65	0.5	2.3

表 3 表格组件性能比较

数据量（列/行）	FWF	EasyUI	ExtJS
5/500	4.9	164.2	4.9
5/2500	23.6	>3000	18.8
20/2500	54.5	>6000	41.7

表 4 树组件性能比较

数据量（层/总节点数）	FWF	EasyUI	ExtJS
3/243	1.45	2.65	2
5/3125	50.5	198.5	108.5

综合 FWF 框架和展现框架之间的性能比较结果, JQuery easyUI 性能稍差, ExtJS 在表格组件上性能较优. FWF 在表单和树组件上略占优势, 综合性能较好.

4 结语

运行时的软件适应和调整是软件柔性的一个重要特征, FWF 模型提供了丰富的 RIA 展现组件在同时, 通过上下文组件 Weblet、视图组件、视图控制器组件、策略连接器组件实现了 Web 应用的柔性驱动, 可以快速客制出复杂美观的界面. 并通过实例验证了可以 FWF 的软件适应性以及软件适应的在线调整能力. 在实现简单界面功能时, 性能相比常规服务器技术(JSP、PHP、ASPX)略有损失, 但是和主流展现框架相比, 在支撑相同组件功能的同时, 在框架的柔性层面有优势, 性能综合评估较好, 已达到大规模应用性能要求.

参考文献

1 Lv J, Ma XX, Tao XP, Cao C, Huang Y, Yu P. On environment-driven software model for Internetware. Science in China (SeriesE), 2008, 38(6): 864–900.

2 梁小江.基于 OSGi 的构件库系统设计与实现[硕士学位论文].西安:西安电子科技大学,2010.

3 Saleehie M, Tahvildari L. Self-adaptive software: Landscape and research challenges. ACM Trans. on Autonomous and Adaptive Systems, 2009, 4(2): 1–42.

4 吴元立,丁博,史殿习,刘惠.普适计算环境下的构件模型映射机制的研究与实现.第 4 届和谐人机环境联合学术会议. 武汉

5 谢 菊 .http://www.ibm.com/developerworks/cn/web/1205_xieju_dtraceajax/.IBM 中国软件开发中心, 2012.