一种支持声明式表示层集成的组件模型①

李 辉[1]，杨 燕[2]，王 帅[2]，白 琳[2]，钟 华[2]

[1](中国科学与技术大学 计算机科学与技术学院，合肥 230027)
[2](中国科学院软件研究所 软件工程中心，北京 100190)

摘 要：从可重用组件或者模块中构造应用是软件工程中的一项重要技术，其中在应用集成相关的领域中，研究工作多集中在数据和业务层上。然而在表示层进行集成可以重用子模块的数据、业务逻辑和界面等多个层面，故而有利于进一步降低开发代价，具有重要研究意义。针对当前表示集成技术的不足，提出了一个新的面向表示层集成的组件模型。该组件模型支持层次化、声明式的组件组合，并设计了一个领域特定语言（DSL）PIDL用来描述组件及其组合。

关键词：表示层集成；组件模型；声明式；组合子；领域特定语言

A Component Model Support Declarative Presentation Layer Integration

LI Hui[1], YANG Yan[1], WANG Shuai[1], BAI Lin[2], ZHONG Hua[2]

[1](School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)
[2](Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Create application from reusable components or modules is an important technique in software engineering. In application integration area, current work mainly focuses on data and business logic level, the research on presentation level integration is insufficient. Because presentation level integration which covers all of the data, business and user interface, it can reduce development cost further more. This paper points out the weakness of current presentation level integration technology and proposes a new presentation integration oriented component model which supports hierarchy and declarative composition. It also designs a DSL named PIDL to describe components and its composition.

Key words: presentation layer integration; component model; declarative; combinator; DSL

从可重用组件或者模块中构造应用是软件工程中的一项重要的技术，能够有效地降低开发代价。在应用集成相关的领域中，如企业应用集成(EAI)，企业信息集成(EII)，服务组合等，存在大量的研究和开发工作，这些工作绝大多数都集中在数据层或者业务层上。然而工程实践和研究表明，用户界面的开发工作是应用开发中最耗时间的部分之一[1]，业界已经有很多在表示层进行粗粒度集成的需求：如 internet 上 web mashups 就是一种需求日增的复合应用构造方式，web mashups 可以分别在数据层、业务层和表示层进行集成。在企业内部，表示层集成技术 Portlet[2]已经被广泛地使用着。

然而在现有的研究和工程实践中[2-4]，对表示层集成的理解因循了以往业务逻辑层组件组合的思路，主要关注组件间的消息交换，在界面组合上策略相对简单，没有考虑界面组合与子组件内容的相关性[5]；组件模型不支持层次化组合，组件间的协作并未考虑到协作范围的问题。导致表示层集成技术的应用范围非常受限。

为了更好地支持以表示层集成的方式来构造应用，本文给出了一个声明式的面向表示层集成的组件模型。该组件模型在保持易用性的同时，允许组

① 基金项目:核高基(2010ZX01045-001-010-4,2009ZX01043-003-001);科技支撑计划(2009BAG18B01)
收稿时间:2011-04-14;收到修改稿时间:2011-06-07

合组件的界面显示与子组件内容联系起来，并支持基于域[6]协作控制机制，以满足复杂的组合需求。

1 框架概述

在详细描述组件模型之前，我们先简要描述该组件模型的运行支撑框架，如图 1 所示。

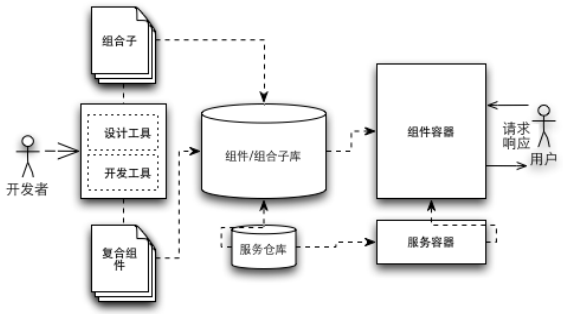

图 1 支撑框架图

在运行支撑框架中，应用就是一个组件，除了某些基本组件外，组件通常是一个能够与用户进行独立交互的模块。组件可以由多个子组件组合而成。

组件组合子是封装了某个组合策略的模块。组件/组合子组织成包，代表一个组件/组合子集合，存储在在组件/组合子库中，通过路径来定位。组件容器和设计工具存取库中的组件/组合子以完成相应的工作。

组件库中的组合组件和组合子使用我们所设计的表示层集成描述语言 PIDL（Presentation Layer Integration Description Language）开发。在组件库中还有一类外部组件，使用第三方组件技术如 Portlet 来编写的组件，通过适配器集成到运行支撑框架中来。

为了方便组合组件的构造，可以把常用的组合策略封装成组合子。允许终端用户在组件设计器的帮助下，基于已有的组件和组合子以可视化的方式构造个性化应用（组件）。

组件运行在组件容器中，组件容器解析组件规范生成运行时实体，接受用户的请求，根据组件的定义执行绘制和事件分发处理逻辑，生成响应结果。在这个过程中，需要调用相关的业务服务。业务服务由组件通过一个服务名来引用。服务仓库提供了服务的注册、发现和引用的设施。

下面我们主要描述组件模型及其组合方法，限于篇幅的关系，本文不对 PIDL 语言进行详细描述，仅在示例中予以展现。

2 组件模型及其组合

2.1 组件模型

组件具有如下特征集：{descriptor, parameters, publish, subscribe, render, execute}，其中：

descriptor 是组件的描述信息，是一个属性集，包含了该组件相关的属性信息。如功能描述，组件名称等，属性（name="PDFReader"）表明该组件名称为 PDFReader。

parameters 是组件的绘制参数集合，通过给定任意一组参数值，组件的绘制接口可以返回其界面表示，绘制参数的值决定了组件当前的导航状态。

publish 定义了组件发布的事件集合，由一个事件属性约束集来给出。

subscribe 定义了组件订阅的事件集合，由一个事件属性约束来给出。

render 是组件界面的绘制接口，绘制接口根据组件容器提供的绘制参数，产生与绘制参数相对应的组件界面表示。界面表示用字符串表示。组件模型要求 Render 接口的处理过程是幂等的，以满足 http 协议中的 Get 操作特性。

execute 是组件的事件处理接口，事件处理接口接受一个事件，可以产生若干个事件，其中可能包含通知容器的事件，如通知容器更改当前会话中的组件导航状态，即绘制参数。

2.2 组件层次结构

图 2 是组件层次结构图。

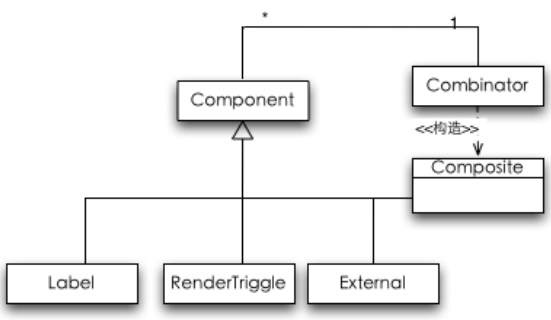

图 2 组件层次结构

在几种主要的面向表示层集成的组件模型中，如 Portlet 和 Widget，组合方法相对简单，主要考虑易用性，不支持层次化的组件构造，而根据现有的基于组件的软件工程方法经验，层次化的组件构造是一种关键特征。我们认为在面向表示层集成的组件模型中也需要支持这一特征，这将使得组件组合结果的可重用性得到增强。

组件分组合组件和基本组件，基本组件是组件容器提供的开箱即用的组件，包含集成进来的外部组件，组合组件由基本组件或者其他的组合组件通过组合子构造而成。下面我们分别描述基本组件、组合组件及其构造方法。

2.3 组件类别

2.3.1 触发器组件

触发器组件表示一个能够触发事件的界面部件，对于 web 应用，它一般是超链接或者按钮，关联着该动作相关的事件描述。

其中 RenderTrigger 组件触发导航绘制请求事件，该事件请求组件容器更改当前组件的绘制参数，这将导致组件的导航状态的变更。组件的构造由相关的绘制参数集构成。如 RenderTrigger（id:32）。

ActionTrigger 组件触发引发业务状态更改的操作。组件的构造则由事件的属性集合构成。如 ActionTrigger(action:"save",id，content)。

2.3.2 事件处理器组件

事件处理器组件接受用户操作界面所触发的动作事件（ActionEvent），或者其他组件所产生的协作事件。事件处理器组件在进行业务逻辑处理的过程中，又可以产生若干个事件。事件处理器组件由 Processor(sub, pub, op)来定义，其中 sub 表示事件订阅，pub 表示该事件处理器在执行的过程中所产生的事件类，op 是业务服务。

2.3.3 外部组件

外部组件是由其他技术实现的组件，如 Portlet，Widget，这些组件需要通过适配器集成到组件的运行环境中来，适配器负责读取外部组件的部署描述信息，为组件容器提供组件规范所要求的信息，如组件描述，组件参数，组件的发布订阅事件，并为组件的绘制和事件处理提供接口。外部组件通过 External(refId)来引用，其中 refId 是外部组件的唯一标识。

2.3.4 组合组件

组合组件是本文的所叙述的方案的核心部分，是表示集成的结果。组合组件由多个子组件构成，在既有组件的基础上，为用户提供新的功能。子组件依据特定的界面布局策略和协作实现组合组件的界面和功能。组合组件同样具备一般的组件模型特征，能够被重用，并且可以用来组成更高一级的组合组件。

3 组合组件的构造

组件的组合分为界面组合与组件间协作两部分内容。

3.1 界面组合

在界面组合方面有两类组合方式，其中 AND 组合方式将所有的子组件的界面全部呈现出来，OR 组合仅仅呈现其中一个子组件。

3.1.1 AND 型组合

AND 型组合将子组件布局在二维空间上。由于本文假定界面描述的形式是高层的声明式描述代码，因此可以使用模版技术来定义 AND 型组合子，完成对子组件进行布局和装饰。本文采用的的模板技术严格遵循 MVC 分离的原则，与 mustache、StringTemplate[7] 类似，但针对组件参数进行扩展。

3.1.2 OR 型组合

本文以 OR 型组合来实现组合组件的界面展示与子组件内容相关联的功能。现有的面向表示层集成的组件模型如 Widget，Portlet 均是静态的界面组合，即组件是否展示取决于用户是否在组件树中导航到目标组件。本文通过组件 OR 组合来提供动态展示模型，满足子组件内容相关的组件展示需求。

OR 组合用一个匹配约束作用在组件集上，将满足匹配的第一个子组件的内容将作为组合组件的显示内容。如果没有任何组件满足匹配约束，则显示一个空的组件。

如下面的 PIDL 代码示例用 OR 组合子构造了一个组合组件，其界面显示内容是支持 pdf 格式的阅读器

组件。

```
Reader=OR {"pdf" in @MimeTypes}
        [PDFReader,HTMLReader]
```

通过这种方式，我们可以实现组合组件的界面与子组件内容相关，从而解决文献[5]中提出的组合组件的导航与子组件内容无关的问题。

3.2 组件协作

构成组合组件的子组件之间不仅仅是界面组合，还需要通过协作以实现单个组件所不能提供的功能，协作可以发生在组件的绘制阶段和事件处理阶段,分别称之为基于参数绑定的协作和基于事件的协作。

3.2.1 基于参数绑定的协作

绘制阶段的协作在用户的导航绘制请求处理阶段产生，该阶段组不执行更改性事务，仅仅绘制组件，参与协作的组件中，可能有组件的显示内容要依赖于其他组件当前的显示内容，比如问题场景中所示，显示相关文档列表的组件其显示内容依赖于文档显示组件的当前内容。通过绘制参数绑定机制来实现这种需求。

基于参数绑定关系的绘制阶段的协作，相当于 Portlet 组件的共享绘制参数，Portlet 组件的参数共享只能发生在全局范围内，本文的组件模型允许有选择地进行共享，参数通过双向绑定关系构成了一个闭集，该闭集构成了一个有限的共享参数协作域。

3.2.2 基于事件的协作

事件处理阶段协作在用户的动作请求处理阶段产生，该阶段执行更改型业务操作，并可能发送协作事件将业务更改通知其他组件。事件先在兄弟组件间派发，然后派发到父组件，这个派发过程递归执行。在组件边界上施加事件过滤器，能控制事件的传播范围。

4 组件组合子

我们构造了一些组合子，这些组合子封装了一些常用的组合策略，如布局，事件过滤等策略。通过利用这些组合子，用户能够很方便地组合出他们所需要的组合组件。下面给出两个最典型的组合子示例。

4.1 **TabPanel** 组合子

TabPanel 组合子的作用时将组件集合以 tab 页布局方式显示给用户，如下列代码所示：

```
TabBox(select)[...]=VBox[HBox [titles],
    OR[...]{select ==@id}]where titles=map{
        {@id=select}=>Focused[
                RenderTrigger(name=@name,
                select=@id)],
        {}=>RenderTrigger(name=@name,
                        selected=@id)
        }[...]
```

VBox 是单列布局组合子，HBox 是单行布局组合子，Focused 组合子将一个组件修饰为获得焦点时的显亮样式，具体定义从略。select 是 TabBox 组合子引入的绘制参数。titles 是根据组件集生成的对应的标签页标题元素集合，该集合是 RenderTrigger 集合。组件过滤器{@id=select}选中当前正显示的组件，用 Focused 组合子特别修饰其标题的显示样式。

4.2 **NotAllowed** 组合子

该组合子属于协作域组合子，这类组合子的职责是控制事件的传播。其中最常用的是 NotAllowed，定义如下：

```
NotAllowed[C]=PubFilter({}=>())C
```

在这个定义中，"{}=>()"定义了一个事件过滤器，该事件过滤器将阻止任何事件传播到组件外部。

5 结语

本文提出了一种面向表示层集成的组件模型，支持层次化组合，允许组合组件的界面显示与子组件内容关联起来，与当前现有的支持声明式的表示层集成的组件模型相比，增强了组合的表达能力。

本文的组件组合语言是声明式的，利用组件组合子，便于开发人员构造复杂的组合组件，通过设计工具的支持，也便于终端用户以所见即所得的方式来构造应用。

参考文献

1 Daniel F, Yu J, Benatallah B, Casati F, Matera M, Saint-Paul R. Understanding ui integration: A survey of problems, technologies, and opportunities. IEEE Internet Computing,

（下转第 9 页）

南日报向广大群众公布。截止 2010 年 12 月 31 日，云南省网上信访系统累计开通部门 2525 个，页面累计浏览量 502247 次。

5 结语

云南省网上信访系统的投入应用极大的拓展了云南省的传统信访渠道，使公众能够更加便利的反应信访问题，拉近了政府和群众之间的距离，同时也使公众对信访工作有一个更加全面的认识和了解。对于政府部门，系统的运用有效提高了工作效率，增加了办事的透明度，规范了办事的流程，主管部门的监督管理工作更加准确有效。由于网上信访系统投入使用时间不长，系统还可进一步挖掘扩展，首先可以考虑利用工作中长期积累的数据通过使用数据挖掘技术对数据进行分析整理，从而为政府的决策提供充分的数据处理，若今后条件允许可将系统延伸至企事业单位，支持。其次是目前系统主要是用于政府部门进行信息提高系统的办理覆盖面。

参考文献

1 林军.基于安全分级的网上信访系统的设计实现.计算机安全杂志,2009,102:59–60.

2 张朝明.XML 开发典型应用.北京:电子工业出版社,2008.492–498.

3 周悦芝.Oracle J2EE 应用开发.北京:清华大学出版社,2005.223–262.

4 Price J. Oracle Database 10g SQL 开发指南.第 10 版.北京:清华大学出版社,2010.457–475.

5 Richardson C. POJOs in Action.中文版.用轻量级框架开发企业应用.北京:电子工业出版社,2007.267–358.

（上接第 49 页）

2007,11(3):59–66.

2 Bellas F. Standards for second-generation portals. IEEE Internet Computing, 2004,8(2):54–60.

3 Yu J, Benatallah B, Saint-Paul R, Casati F, Daniel F, Matera M. A framework for rapid integration of presentation components. Proc. of the 16th international conference on World Wide Web, New York,ACM, 2007:923–932.

4 Concolato C, Feuvre JL, Dufourd JC. Declarative interfaces for dynamic widgets communications. Proc. of the 9th ACM symposium on Document engineering, New York, ACM, 2009: 241–244.

5 Díaz O, Irastorza A, Azanza M, Villoria F. Modeling portlet aggregation through statecharts. Karl Aberer, Zhiyong Peng, Elke Rundensteiner, Yanchun Zhang, and Xuhui Li, Web Information Systems–WISE 2006, volume 4255 of Lecture Notes in Computer Science, Berlin, Springer, 2006:265–276.

6 Fiege L, Mezini M, Mühl G, Buchmann A. Engineering event-based systems with scopes. In Boris Magnusson, ECOOP 2002–Object- Oriented Programming, volume 2374 of Lecture Notes in Computer Science, Berlin, Springer, 2006: 257–268.

7 Parr TJ. Enforcing strict model-view separation in template engines. Proc. of the 13th international conference on World Wide Web, New York, ACM,2004:224–233.