**e4 技术白皮书**

John Arthorne, IBM Canada Inc.

Last modified July 29, 2009

译者： 天马

# 概述

Eclipse平台的初衷是构建一个可扩展的IDE组件框架，但它现在已经发展成为一个构建可扩展的任何软件应用的通用平台。目前，Eclipse应用出现在了各种部署环境中，比如Web服务器，Web浏览器，嵌入式客户端，以及传统的富桌面应用。

E4平台的设计是为了简化软件组件以及基于组件的应用的开发，以满足当前不断变化的计算场景的需求。本文主要介绍e4的架构和编程模型。

# 什么是e4

E4是一组相关技术的集合，这些技术主要用于构建可扩展的、基于组件的应用。E4将一组新的技术融入现有的eclipse平台，而不是对eclipse平台的大规模替换。目的就是使eclipse组件更易于编写，更可配置，并更易于在不同的运行时环境中复用。

第一代Eclipse平台（1.0到2.1）主要是一个集成平台，其主要功能是将不同作者开发的插件集成为一个一通用应用，并使最终用户有一个一致、内聚的体验。

第二代Eclipse平台（3.x）开始构建于OSGI运行时之上，这使Ecliplse成为一个更加强大的、通用的、基于组件的应用框架。第二代平台可用于很小的嵌入式应用、RCP应用、以及Web服务器。然而，每个组件（插件）通常难以在特定环境（组件的设计和测试环境）以外复用。为系统添加或删除组件很容易，但将一个为某应用或运行时环境设计的组件复用到一个完全不同的应用或环境中，将非常困难。

E4的愿景是使开发在多种应用和环境中都可复用和可定制的组件变得更容易。有两种途径达到这个目的：减少组件的外部依赖；增加可无缝集成到Eclipse运行时中的语言和技术。这两种方法都被Eclipse采用了：

1. 面向服务的编程模型，基于OSGI，把软件组件更好地从外围环境中分离；
2. GUI被描述为统一的模型，此模型可被查询、操作、做工具、扩展，还可在很少或无需编码的情况下快速的设计和定制用户界面；
3. 使用Web样式技术（CSS），无需修改应用代码，即可任意修改UI展现；
4. 把Eclipse运行时技术融入到JavaScipt世界，允许JavaScript写的软件在Eclipse运行时上执行；
5. 声明式地定义SWT应用的设计和结构，摒除了重复的公式化的SWT代码，这降低了开发成本，提高了UI一致性，并且允许定制应用发布；
6. SWT新的移植，称为“浏览器版本”，允许现有的SWT应用在像ActionScript/Flash 一样Web平台上运行；
7. 在开发工具方面，更灵活的资源模型为复杂的工程结构提供了更好的支持。

本文的后续章节将更详细的介绍这些新技术。

# 编程模型

E4编程模型开始于现有的Eclipse编程原则：

1. 应用由模块化的，被称为插件或bundle的松耦合的组件组成。Bundle可用Java或其它语言来编写，还可以包括文档和源代码等资源；
2. Bundle可以通过扩展点声明它可被定制或扩展的地方，还可以通过定义扩展来定制或扩展其它bundle；
3. 一次可以安装大量bundle，但只有实际使用的bundle才被加载并消耗系统资源。

E4区别于这种传统Eclipse编程模型的地方在于，bundle怎样在扩展注册表机制以外彼此交互。Bundle经常以不适于Eclipse扩展注册表的方式为其它bundle提供数据和服务。最常用的实现方式是，bundle通过直接引用API中定义的方法和常量来与其它bundle通信。为了获得服务的单实例，bundle一般会定义入口点（比如Platform,IDE, ResourcesPlugin, JavaCoret等这些类）。这种行为会导致使用服务的bundle和提供服务的bundle间存在紧密偶合，这种单实例访问的流行，将使其它供选择的服务实现很难或不可能被采用，多个实现也很难或不可能在同一时间存在。结果bundle将难以在不同的环境中重新装配或复用，而这些环境可能需要不同的或多个服务实现。

服务编程模型定义了三个不同的参与者：服务提供者、服务消费者和一个代理或容器（绑定服务提供者到消费者）。此编程模型的基本实现，比如OSGI服务API或Eclipse扩展注册表，出色地将服务提供者与消费者解耦，但经常需要服务提供者和消费者显式地知道特定容器和服务代理。

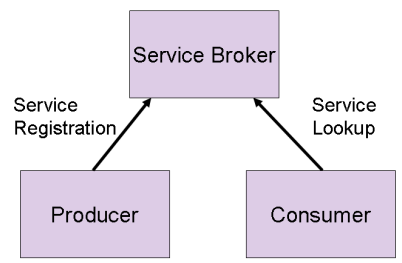


图1 简单服务架构

E4编程模型通过减少或消除服务提供者与消费者显式的对特定服务代理技术的依赖，进一步将这些参与者解耦。这种新的灵活性将在下面的介绍中给出详细解释。

## 上下文（context）：服务代理（the service broker）

E4引入“上下文”的概念，作为存储查找服务并将这些服务提供给消费者的通用机制。从这个基本功能来说，e4上下文看起来很像存储键值对的Java Map，客户可以向此Map里存值或取值。这个Map还可以存储，在客户请求上下文值时，可懒计算这些值的上下文函数。当客户请求了一个上下文中不存在的值时，请求将被委托给父上下文。这允许服务和数据存储在一个中心位置，被多个客户消费。上下文还有一个可插拔的查找策略，来允许外部参与者教上下文怎样获取特定种类的值。此查找策略使e4上下文和其它服务代理（如OSGI服务注册表）间的互操作性成为可能。这种灵活性意味着，各种不同的服务查找方法和代理系统都可以集成到e4上下文机制中。创建上下文层次结构以及插入服务查找策略的能力，可使上下文适用于从非常简单的类似map的服务注册与查找，到高度复杂的动态服务仲裁机制。

## 注入（Injection）：服务消费者（service consumers）

目前服务编程模型的最佳实践，是消费者通过依赖注入来得到依赖的内容。 这理论上可使应用代码完全消除对特定容器技术的依赖，从而保证了更大的复用。E4直接支持并鼓励把依赖注入作为向客户提供服务的手段，并支持构造函数注入、方法注入、以及字段注入。在客户代码中标识注入点的方式，可以是命名规范或Java注解。对于不太理解控制反转，而且喜欢代码清晰胜于框架独立（prefer code clarity over framework independence）的客户，还可以直接使用e4上下文（service locator设计模式）。

## 服务提供者

Bundle提供的服务和数据多种多样。有些服务非常轻量，可被使用或丢弃成千上万次，然而其它一些重量级服务可能要“活”很长一段时间。这些服务的生命周期也各不相同，有些依赖于特定UI控件的存在而存在，而有些可能贯穿于某bundle的整个生命周期。正是由于这些多样性，所以没有哪种服务发布方法可以适合于所有的服务提供者。

通常，e4中发布的服务使用OSGI服务机制。服务的注册和删除，可以通过编程，还可以通过OSGI声明式服务。声明式服务允许服务直到被请求时才实例化。当然，有很多种辅助框架可用于发布OSGI服务，比如，Spring DM, iPOJO, Peaberry等等。这些模型使用OSGI服务作为服务发布的基础，可以与e4无缝集成。

大多数客户代码将使用服务，创建上下文的框架可以直接简单地向上下文中添加服务。比如，一个关联到某控件的服务，可能要注册到控件构造函数中的上下文，并在控件销毁时撤消服务。手工注册可使服务仅在特定上下文中才变得可用，而通过OSGI服务，可使服务一直可用。最后要提到的是，上下文支持通过字段或访问方法注出（outjection，反注入）服务回至上下文。

最终，我们有了一个完整的模型：服务消费者不知道谁提供服务，也不知道用什么代理获取服务。通过分离服务配置数据和服务实现，服务提供者可与服务代理大大解耦。这可以使用声明式机制，如DS、注出，或者简单地从服务实现中分离框架相关的注册代码。下图说明了这个模型。

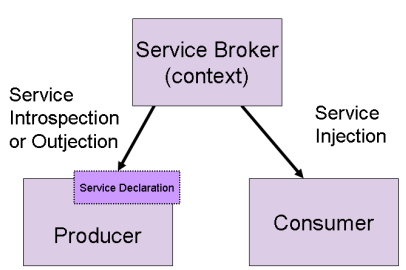


图2 e4服务架构

## Eclipse应用服务

为支持这种解耦的服务编程模型，Eclipse API要变成可用的服务，而不是以前那种单实例访问器模型（Platform、ResourcesPlugin等）。然而，多年来Eclipse平台已经积累了大量的API，但很多组件仅使用这些API中的一小部分。仅这些API的广度就使基于Eclipse进行应用开发成为一个挑战，陡峭的学习曲线增加了学习成本，阻碍了新手的开发。e4将通过定义一组核心服务获取大量有用的平台功能来解决这些问题。目的是开发人员应该可以仅使用这些核心bundle，就能很好的构建完整的e4 bundle。这些核心服务还为把其它编程语言集成到eclipse平台定义了好的起点，后面JavaScript集成相关章节将进一步讨论这个话题。

# 基于模型的UI

上一代Eclipse平台UI（称为workbench）复杂而且难以维护。虽然多年来它已经变得稍更灵活，但它仍然坚持着workbench结构和布局的严格、硬编码的模型：一个workbench包含多个workbench window，一个workbench window包含一个或多个workbench page，每个page又包括一个editor area、一组view stack，以及一些硬编码的修剪元素（透视图切换按钮，进度指示器等）。应用设计者在定制eclipse应用的结构时有一些严格的限制。

E4 workbench为应用设计者大大提高了灵活性，因为它提供了一个正式的workbench组成元素的模型。应用可以重新装配或扩展这个模型来实现各种不同的应用展现，而无需编码。将workbench结构标准化为一个定义明确的模型有一个额外的好处，就是使workbench代码更加简单，而且更少有潜在的错误。最重要的是，这使一些非常不同的workbench UI布局成为可能，比如视图或编辑器脱离透视图，对话框中放视图或编辑器，以及其它一些老eclipse平台不允许的设计。有了模型，还可以为应用设计者提供更加高级的工具，比如可视化设计工具。可以在运行时操作e4 workbench模型，模型将立即反应到UI上。这开始了脚本式操作workbench结构和状态的历史，就像JavaScript操作web浏览器中的dom一样。下面图3中，我们可以看到定制正在运行的应用的模型编辑器，这个例子中，正在改变传统eclipse 问题视图的name和tootip。

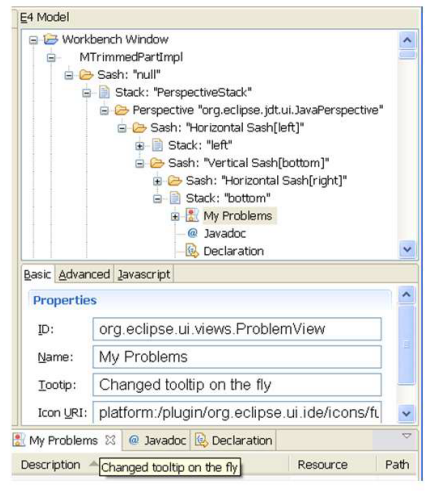


图3 e4 模型编辑器

E4 workbench模型被翻译器翻译成小部件。Workbench自带一个默认的将模型翻译成SWT小部件的翻译器，其它翻译器也可以为模型元素提供不同的翻译。这可用于为某部件提供细微变化，甚至翻译成完全不同的部件库或类似web浏览器的运行时环境。翻译器是在单个模型元素的级别提供的，而不是为整个应用提供一个整体的翻译器。单个翻译器可以为一个或多个模型元素提供翻译，或者相反，多个翻译器也可以为单个模型元素提供翻译。这种精细的可扩展性允许客户用他们自己的元素扩展workbench模型，然后提供自定义的翻译器来以一种特别的方式翻译这些元素。

# 声明式样式

Workbench模型到小部件的基本翻译是由翻译器完成的，可插拔的样式引擎则用于定制部件的字体、颜色、以及等其他方面。我们知道，web展现技术中分离文档结构（html）和样式（CSS），保证了多个文档可以有一致的用户体验，而且也很容易在单个地方变化样式。基于模型的UI对E4样式引擎是透明的，事实上这个引擎可以完全独立的运行到早期版本的eclipse中。具体的部件和样式数据做为引擎的输入，将样式应用于部件实例上就生成了样式结果。下面图4显示了一个完成流程：模型通过一个多个翻译器变成了部件，然后通过样式引擎和声明式样式数据（CSS文件）产生了最终结果。

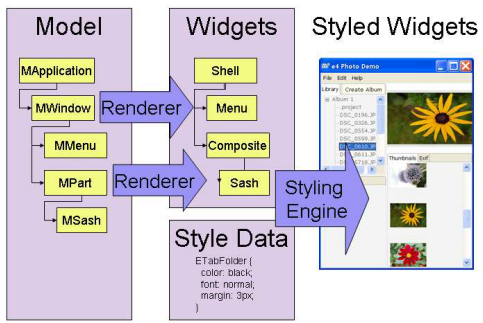


图4 翻译器和样式数据流

e4中的声明式样式是基于标准CSS语法的，支持HTML CSS中的大部分属性名和类型格式，比如字体、边距、颜色等。此外，e4还有自定义的属性以及eclipse部件特有的伪选择器（pseudo selectors）。CSS中你可以使用类型选择器，e4中你可以使用部件类名做为选择器。E4还支持CSS class选择器和ID，所以同一个部件可以有不同的样式。同样，e4用CSS 伪类（pseudo-classes）来基于部件状态选择样式规则，这让你可以为被选的tab设置不同的字体颜色。比如，下面的片段为普通的tab folder指定一种颜色，为被选择的编辑器指定了一个不同颜色：

CTabFolder {

background-color: rgb(241, 240, 245);

font: normal;

}

CTabFolder.editors:selected {

background-color: rgb(255, 255, 255) rgb(255, 247, 229);

}

CSS类和伪类的使用，使workbench部件（视图，编辑器）以及特定状态部件的样式高度可定制。比如，正在运行后台任务的视图可以标记为忙，样式则可以描述怎样展现这个状态：不同的字体，改变的边距，如甚至没有样式。最终将使应用更容易被用户理解和使用。

# Web到桌面

过去的几年中，浏览器应用和桌面应用的界限已经变得模糊。应用中的浏览器、部件、和语言框架的复杂性也在接近桌面操作系统。与此同时，桌面应用正变得web化，也在采用一些web应用的用户交互方式和样式标记。一段时间内，桌面应用和浏览器应用的需求都将存在，而且使软件组件能同时用在两种环境的需求正越来越多。在面向组件的软件（如eclipse）中，我们不再选择目标平台，然后从头到尾的构建整个软件。我们基于大量的可用组件，然后将他们组织一起来构建一个符合我们需求的应用。如果我们能在多个运行时环境中复用同一组件，将大大减少开发和维护成本。

E4正在以多种方式探索在多个目标平台和技术间复用组件。其中一种是用JavaScript写Eclipse组件。

作为客户端浏览器编程语言的事实标准，JavaScript对于那些寻求编写可在多种运行时环境中运行的组件的人，是一个好的选择。为此，e4正在研究将eclipse的优势（模块化、可扩展性、制作工具）带入到JavaScript世界，并将JavaScript组件融入到eclipse桌面环境中。虽然目前e4的焦点在JavaScript上，但这项工作的意图更加广泛：把用多种不同语言编写eclipse组件变得更简单。

虽然JavaScript被长期用于浏览器脚本语言，但很少用于构建类似于我们在Java世界中看到的大型应用。它的一个弱点是缺少模块化机制。没有定义和访问命名空间以及表示JavaScript代码段依赖的机制，也没有以一种可扩展的方式查询和消费其它组件定义的服务的机制。在java和eclipse中，我们有强大和成熟的OSGI模块化系统，可以满足这些需求。

为了解决这些限制，e4在OSGI基础上引入一个模块化框架，用于JavaScript应用。它允许创建模块，创建大规模纯JavaScript应用，甚至把JavaScript bundle集成到传统的基于Java的OSGI运行时中。图5 说明了e4 JavaScript框架构架，以及它和纯JavaScript bundle、标准java bundle间的关系。

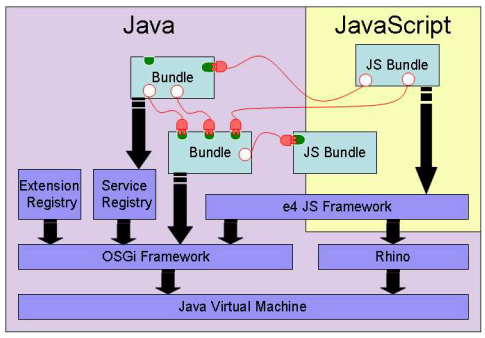


图5 e4 JavaScript框架

e4 JavaScript框架是用java写的，运行为一个纯OSGI bundle。它负责解析JavaScript bundle的描述文件（manifests），并分析JavaScript bundle间的依赖。JavaScript bundle以及它们的依赖关系对OSGI框架是透明的：JavaScript框架本质上完全是一个OSGI实例中嵌套的框架。JavaScript bundle间的依赖关系可表示在bundle级别（就像OSGI 头信息中的Require-Bundle），或者在单个脚本文件级别（类似OSGI 头信息中的Import-Package）。这些依赖表示在JavaScript manifest中（一个JSON文件）。下面是一个简单JavaScript bundle manifest例子：

{

"Bundle-SymbolicName":"sample.jsbundle",

"Bundle-Version":"1.0",

"Bundle-ScriptPath":"script.js",

"Import-Package":"a.resource;version=[1.0.0,2.0.0)",

"Export-Package":"sample.resource;version=1.0.0",

"Require-Bundle:"some.other.bundle",

}

JavaScript bundle可用编程的方式定义或安装到框架中，还可在标准OSGI bundle中，用额外的manifest header声明JavaScript bundle的存在：

JavaScript-Bundle: scripts/manifest.json

虽然JavaScript bundle和javabundle可通过直接调用来交互，但推荐的方式是通过OSGI服务注册表，或是eclipse扩展注册表。比如，JavaScript bundle可以声明为一个标准OSGI bundle，然后编程式地或使用类似OSGI DS的声明式机制向OSGI服务注册表贡献服务。标准OSGI bundle就可以消费这些服务，而根本不知道这些服务的实现是用JavaScript写的。

同样，e4 JavaScript框架提供eclipse扩展工厂来实例化纯JavaScript 写的eclipse扩展。JavaScript bundle可以简单的声明一个plugin.xml文件来记录向扩展注册表的扩展，而不需要写一行java代码。Java写的bundle就可以消费这些扩展而不知道他们是用另一种语言写的。相反，JavaScript bundle可以用一些相应的java API声明一个扩展点，这些API可被java bundle实现，然后加载并用JavaScript代码向这个扩展点提供扩展。这些JavaScript bundle和java bundle间的交互在图5中可以看到。E4 JavaScript框架本身用java和JavaScript定义API。这样纯javabundle可以查询或操作JavaScript框架（比如安装一个新的JavaScript bundle），JavaScript写的bundle可以通过框架的 JavaScript API与框架交互。图5中e4 JavaScript框架块同时扩展java和JavaScript说明了这一点。

E4编辑模型还简化了集成用其它语言实现的组件。通过组件间基于服务的交互、依赖注入，组件不必知道服务用哪种语言实现，也不必知道用什么语言消费它们暴露的服务。类似地，消减后的eclipse应用服务可以暴露为其它语言的API，所以eclipse平台的大量功能可被其它语言实现的bundle使用。用于与这些核心e4服务交互的少量JavaScript API现在已经可用，它们定义在org.eclipse.e4.ui.web bundle中。作为概念的证明，e4包含了一个用纯JavaScript 实现的PDE update site编辑器（见图6）。这个编辑器可在web浏览器中运行，还可以无缝集成到eclipse平台UI中。



图6 e4 JavaScript site编辑器

# 桌面到Web

我们已经展示了web组件怎样集成到eclipse平台中。然而，用传统java语言实现的桌面应用也可以移植到web平台。E4中有两个研究方向与此有关：新的SWT移植版本、Eclipse Rich Ajax Platform (RAP)。

Eclipse SWT为桌面图形应用（可跨多个操作系统）提供了通用API。它允许开发者只写一次，即可得到多个目标平台上的高性能、具有本地化外观的应用。同样，web浏览器编程有多种语言和部件工具包。Web技术变化迅速，由于担心过时，应用开发者都不愿只掌握一种技术。E4中SWT的新移植版本（称为SWT浏览器版本，SWT/BE）旨在为web UI 编程提供一个通用平台，就像它已经为桌面应用编程所做的那样。这潜在的允许已有的SWT应用运行在Web上，并允许开发者构建web UI应用而不必锁定在某个web技术中。

SWT/BE现在支持 ActionScript/Flex 做为此项技术的原型样例，但移植到其它web平台，比如JavaScript/Dojo、 Silverlight、 Java FX等，也是可能的。图7显示了运行在Adobe Flex 上的SWT控件样例，它包括所有SWT主要部件。

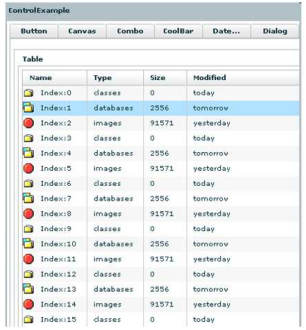


图7 Flex上的SWT 控件样例

Eclipse Rich Ajax Platform (RAP)提供了运行在Web上的SWT、JFace、Workbench UI实现。类似SWT/BE， RAP为同一个应用运行即运行在桌面又运行在web上提供了可能。然而，原始的RAP实现需要一些eclipse平台代码来支持web目标环境。特别地，Web应用通常必须支持在一个应用实例中存在多个并发的用户会话，由于大量使用单实例，eclipse workbench通常不支持这个。

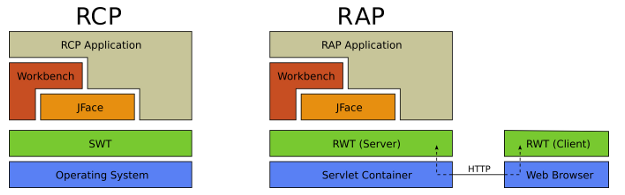


图8 RAP构架

E4编程模型由于它面向服务注入的编程方式，使得它更有适于多并发会话。早期实验中在RAP运行时上运行的e4应用已经预示，workbench仅需要较少的变化即可运行在RAP上。RAP集成工作为e4的目标（支持多种运行时环境是可以实现的）提供了验证。在类似RAP的目标平台上运行e4应用的工作，有助于保证e4架构和编程模型的持续灵活性和开放性。

# 声明式控件

基于模型的e4 workbench为workbench本身提供了一个抽象展现：windows、pages、 perspectives、 view等。此模型用翻译器可翻译成SWT，可用声明式样式引擎进行定制。然而，在单个workbench视图内，UI仍然用普通的SWT来创建。这些SWT代码通常重复，而且样式被硬编码。为了解决这些问题，e4正在探索将模型/翻译器概念应用于SWT的所有方面。目前这个探索有两个方向：基于XML的小部件和基于模型的小部件。

## XWT：基于XML的声明式控件

XML UI for SWT (XWT)，是一个框架，可将SWT小部件声明在xml中。XWT中，应用的所有结构、小部件的层次、小部件到特定底层应用模型的绑定，以及小部件到实现小部件行为的java回调的绑定都是声明式的。

XWT以声明式的UI数据作为输入，与应用模型一起生成小部件。XWT包含一个符合javabean规范（简单数据访问器和setter方法）的类的简单模型。其它方法可以通过扩展点向XWT扩展。XWT UI 生成器组合声明式UI数据和模型定义，生成运行时SWT/JFace控件。图9展示了XWT的架构。

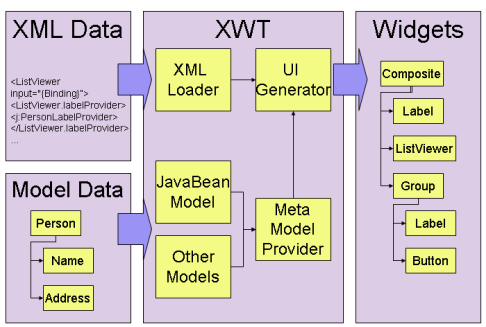


图9 XWT 架构

## TM：基于EMF的声明式控件

Toolkit Model（TM）是UI元素的抽象EMF模型。在应用运行时，此模型被绑定到一组具体的小部件(比如SWT)上，TM构建器在这个模型与小部件间双向传播变化，以保持它们的同步。应用在运行时通常与TM而不是具体的小部件交互，这让应用开发者面对一个更简单的抽象来工作。

这个更高级的部件抽象使改变具体部件实现成为可能，可以以精细的方式改变，也可以是根本改变，比如与运行在不同进程或不同物理机器上的小部件实例进行交互。举例来说，一个运行在web服务器上的应用可以操作此服务器上的toolkit模型，并且TM运行时可以透明的用运行在不同机器上的小部件实现这个模型，比如一个web浏览器客户端。

Toolkit模型中的事件由JavaScript实现的事件处理器来处理，也可以用java实现。JavaScript与toolkit模型交互的方式与JavaScript与操作html DOM类似。由于有图形工具可用于构建或操作EMF模型，这种EMF和JavaScript的组合可以进行应用的快速原型设计。

图9展示了TM架构。构建器负责创建小部件并把它们绑定到模型上，这是通过为每一种创建的小部件找到一个合适的绑定器实现的。绑定器创建具体的小部件，并在创建后管理小部件和模型元素间的同步。每种小部件有一个绑定器，而不是每个部件实例一个绑定器。JavaScript代码为模型提供动态行为，包括部件事件的回调和其它应用行为。

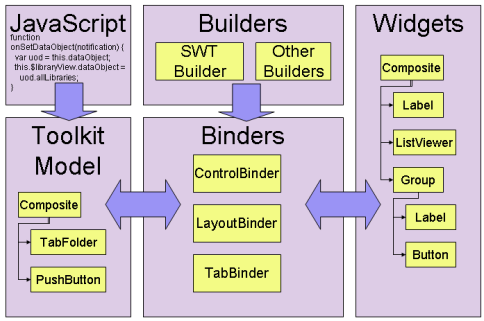


图10 TM构架

# 灵活的资源模型

开发工具仍在eclipse生态系统中保持着重要角色。许多IDE开发者都希望以前版本的eclipse支持更复杂的工程结构。由于eclipse严格的资源模型限制，一些开发工具的用户很难把已经规划好的源码及其它资源的结构应用在基于eclipse的IDE中。E4引入了新的增强的资源模型，以更好的支持在workspace导入和管理更复杂的工程结构。这些增强包括：



1. 可以在工程级别定义路径变量，并创建相对于这些变量的链接资源。这使得移动那些包含复杂链接结构的工程时，不会破坏链接。
2. 称为组的新的虚拟文件夹。组不存在于文件系统中，但它们可用于创建任意复杂的、包含资源链接的虚拟工程树。
3. 支持工程和文件夹的过滤，可从eclipse workspace树中排除特定资源或符合某模式的资源。过滤器允许eclipse工程包含持有大量元素的文件夹，但仅有一小部分元素在工程中，排除的资源将不占用内存。

1. 通过拖拽来创建和管理链接资源。从eclipse外拖拽一个文件树时，你可以把这个树作为包含链接和组的虚拟树放到eclipse workspace中，而不是作为一个具体的文件系统树。
2. 允许编辑已有链接资源的位置，或把链接在绝对路径和相对于变量的路径之间回来转换。

这些增强综合起来可以让用户在workspace中快速搭建复杂的工程结构，并与其他用户共享这些工程，而虚拟工程结构保持完好。

# 结论

E4工程引入了很多新技术使eclipse平台的架构和设计原则更加现代。这些变化使平台更加开放，支持更多的编程语言和目标运行时环境，从而使eclipse平台的未来更加美好。E4组件将更容易定制、配置并在不同的应用中复用，而无需修改。从应用逻辑中分离样式和展现逻辑，可使eclipse应用以一致的方式方便的更换皮肤。遵守e4设计原则的开发者，在构建组件和应用时，将与底层操作系统和web浏览器的技术变化更加隔离。