

模板技术在内容管理系统中的应用

南 轶, 李先国

(西北工业大学 计算机学院, 陕西 西安 710129)

摘 要: 论述了运用模板技术的相关理论、系统的设计思想、系统开发中用到的相关技术及一个基于 .NET 的网上购物网站中模板技术的设计与实现过程, 自主开发了一套模板引擎, 利用正则表达式和缓存技术, 提高了标签的解析速度和系统的执行效率, 详细介绍了该系统中模板引擎的设计和实现。

关键词: 内容管理系统; 模板技术; 模板引擎; 模板; 标签

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2012)06-0180-05

Research and Application of Template Technology in Content Management System

NAN Yi, LI Xian-guo

(School of Computer, North-Western Polytechnical University, Xi'an 710129, China)

Abstract: This paper introduces the background related to content management, discusses the relevant theory and technology of template technology, the design method and a detailed design and implementation of development process of template technology in an online shopping website based on .Net technology, using regular expressions and cache technology developed a template engine independently, the dependent template engine improves the speed of label interpretation and system execution efficiency. On how to design and implement the template engine are mainly discussed.

Key words: CMS; template technology; template engine; template; label

1 引言

内容管理系统 CMS (Content Management System) 是一种位于 WEB 和业务应用系统、办公系统之间的软件系统, 它以信息共享为基础, 是一个面向海量信息处理, 集信息数字化、分布存储、管理、传播、查询为一体的管理平台^[1]. CMS 是基于强健的多层体系架构, 遵从开放标准, 易于与其他应用集成, 实现功能扩展和快速部署^[2-3]. 内容管理系统由内容采集模块、内容管理模块、内容发布模块组成, 其中内容发布模块直接关系到系统的执行效率, 而模板技术高度体现了 MVC (Module-View-Controller) 的设计模式思想, 它实现了内容的管理和设计的分离, 允许页面设计人员只需要关注内容的展现方式即可, 无需考虑逻辑程序的设计问题, 从而帮助

一般用户方便快捷地建站并对网站进行维护. 将模板技术应用于内容管理系统中, 可以方便地实现所管理内容的热插拔, 提高系统的灵活性和可用性.

2 模板技术概述

模板技术, 是一个特定格式的字符串变量动态替换技术, 我们将这个特定格式的字符串变量称为标签. 完整的模板技术需要包含四要素: 模板语言、包含模板语言的模板文件、拥有动态数据的数据对象和模板引擎^[4].

(1) 模板语言: 主要包含变量标识和表达式语句.

(2) 模板文件: 模板文件是指包含了模板语言的文本文件或数据库数据.

(3) 拥有动态数据的数据对象: 模板文件包含

的是扩展内容,那么其所需的动态数据就需要另外提供。

(4) 模板引擎

工作可以分为三步:

① 取得模板文件并确认其中的模板语言符合规范。

② 取得数据对象。

模板引擎根据请求参数查找数据库中的模板信息,找到与请求页面对应的模板,解析模板中的静态标签,获取数据对象。

③ 合并模板文件(扩展内容)和数据对象(动态内容),并生成最终页面。

合并的机制是,模板引擎通过正则表达式匹配出模板文件中的所有标签,从数据对象中取得相应的值,并用最终的标签内容替换原始标签。

3 相关算法与技术

模板引擎的工作原理主要是采用模式匹配的一些算法来替换模板文件中的静态标签,相关的算法有 BF(Brute Force)算法, KMP 算法, BM(Boyer-Moore)算法, 正则表达式等。在本文设计的这套模板引擎中主要采用的是正则表达式技术。

3.1 正则表达式

正则表达式提供了一种从字符集中搜寻特定字符串的机制。用户通过使用一系列的特殊字符构建匹配模式,然后把匹配模式与数据文件、程序输入等目标对象进行比较,根据目标文件中是否包含匹配模式,执行相应的程序^[5]。

正则表达式的主要功能包括:数据有效性验证功能,替换文本功能和提取符合规则的子字符串。

正则表达式的匹配过程如图 1 所示。

为提高匹配效率,通过 Regexp. Options. Compiled 选项将正则表达式预编译后保存在类的扩展变量中,防止每次解析时都重新生成和编译正则表达式。

3.2 工厂设计模式

工厂设计模式是我们在系统开发中广泛使用到的模式,它专门负责将有共同接口的类实例化,把对象的创建过程封装起来,使得创建对象的责任与使用对象的责任分离开来,符合面向对象的设计原则。

3.3 模板设计模式

本文将按照模板方法(Template Method)的设计模式思想来设计开发模板引擎的解析功能。

Template Method 是一种简单模式。假设一个

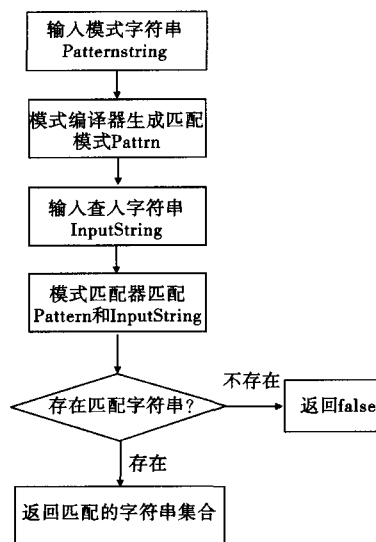


图 1 正则表达式的匹配流程图

抽象类是其他继承类的基类,在抽象类中实现一些通用方法,继承的所有对象都可以使用这些方法。子类还可以实现自己的其他方法。

模板方法的基本思想是:模板方法模式在操作中定义算法的框架,至于具体的步骤由子类决定。模板方法模式允许子类重新定义算法的某些步骤,而不用改变算法的结构^[6]。

4 模板技术的设计与实现

4.1 开发的技术框架

本系统是在 .NET 框架下使用 C# 语言对整个系统进行设计和开发,采用 MVC 的架构模式。

.NET Framework 是企业体系结构依赖的最终工具箱。它是一个计算平台,目标是简化应用程序的开发。

MVC 模式是一种网络应用的设计和开发模式,其主要思想是将应用程序中的用户界面、业务模型和业务逻辑控制实现分离,这样一方面可以在开发中有效地实现分工与合作,提高了开发的工作效率;另一方面,可以使整个系统更趋于模块化和结构化,易于扩展和维护,提高了代码的复用性^[7]。

4.2 模板技术的分析与设计

一个完整的模板的设计与实现包含三个部分:模板引擎的设计、模板管理的设计和标签管理的设计,其中模板引擎是模板技术中最核心的部分,它主要负责接收访问者的请求,并从相应的模板中解析具体内容并将最终结果展现给访问者,在我们所开发的模板引擎中主要由 3 个部分组成,分别是请求接收模块、模板解析模块和内容展现模块。模板引擎

的工作原理如图 2 所示。

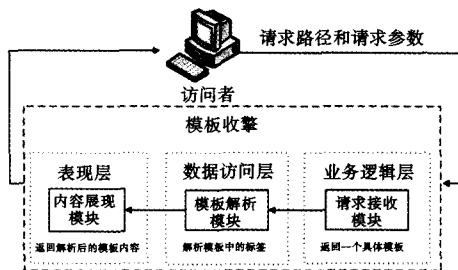


图 2 模板引擎的工作原理图

4.3 系统缓存设计

目前常见的内容管理系统的缓存策略分为静态缓存和动态缓存。静态缓存是在内容发布的同时就立刻生成静态页面并将其缓存起来，它具有比较复杂的触发机制^[6]，对于大型的复杂的内容管理系统来说并不合适，因此我们选用动态缓存的方法来解决系统的缓存问题。

采用动态缓存，在新页面发布以后，并不预先生成相应的静态页面，而在对该页面发出请求时，如果前台缓存服务器中没有缓存该页面时，再向后台内容管理服务器发出请求并将其缓存起来，这样用户第一次访问页面可能会有点慢，以后就是直接访问内存中的扩展页面了，速度会提升很多。

在模板解析的过程中，由于相同的标签可能会出现在其他页面，因此我们对标签也进行缓存。采用标签名称加所带参数的组合作为标识符将标签内容存入缓存中，这样，当两个或多个模板中包含了相同的标签时，除解析第一个模板的该标签需要访问数据库以外，其他模板的标签可以直接从缓存中获取标签内容而无需再访问数据库，降低了数据查询所带来的效率损耗，提升了模板的解析效率。

5 模板引擎的设计与实现

5.1 请求接收模块

5.1.1 功能概述

我们将一个网站中各种不同的页面类型设计成一个模板，本系统共有 84 个模板类型，其中每一种模板类型又对应有三种不同风格的模板，用户初次使用时选用系统提供的默认模板，当访问者客户端访问某一个页面时，系统会根据请求的页面类型及参数查找对应的模板，模板解析模块对该模板进行解析。

5.1.2 类关联图

下面以请求接收模块中的模板信息类 Mould-

Info 的类关联图为例来说明本系统中各个层之间的调用关系，如图 3 所示。

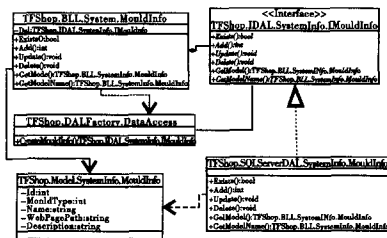


图 3 MouldInfo 类的类关联图

其中，类及其所在的命名空间的功能描述如下：

(1) TFShop.DAL.Factory: 数据工厂，调用工厂方法生产 IDAL 对象；

主要的成员方法：

public TFShop.IDAL.SystemInfo.IMouldInfo CreateMouldInfo();

(2) TFShop.IDAL: 数据接口层，抽象出操作数据库相应表的方法，放在相应的接口中；

主要的成员方法：

public TFShop.Model.SystemInfo.MouldInfo GetModel(int Id);

// 根据传入的模板 ID 获取对应的模板

public TFShop.Model.SystemInfo.MouldInfo GetModelName(string Name);

// 根据传入的模板名称获取对应的模板

bool Exists(int Id); // 是否存在记录

int Add(TFShop.Model.SystemInfo.MouldInfo model);

// 增加一条数据

void Update(TFShop.Model.SystemInfo.MouldInfo model);

// 更新一条数据

void Delete(int Id); // 删除一条数据

(3) TFShop.SQLServerDAL: 数据库访问层，针对 SQL Server 数据库，实现 TFShop.IDAL 接口中的相应方法。

(4) TFShop.BLL: 业务逻辑层，再次封装 IDAL 中定义的方法，方法实现中调用 IDAL 中对应的方法。

主要成员变量

private readonly IMouldInfo dal = DataAccess.CreateMouldInfo();

主要成员方法

```
public TFShop. Model. SystemInfo. MouldInfo
GetModel(int Id);
//返回 dal. GetModel(Id)
public TFShop. Model. SystemInfo. MouldInfo
GetModelName(string Name);
//返回 dal. GetModelName(Name)
public bool Exists(int Id);
//返回 dal. Exists(Id);
public int Add(TFShop. Model. SystemInfo.
MouldInfo model);
//返回 dal. Add(model);
public void Update(TFShop. Model. System-
Info. MouldInfo model);
//返回 dal. Update(model);
public void Delete(int Id);
//返回 dal. Delete(Id);
```

(5) TFShop. Model 数据模型,针对数据库中的表,提取相应的字段,并将字段封装成属性;

```
主要成员变量
public int Id;//模板 id
private int MouldType;//模板类型
private string Name;//模板名称
private string WebPagePath;//模板路径
private string Description;//模板描述
```

5.1.3 关键实现代码

```
TFShop. BLL. SystemInfo. MouldInfo bll =
new TFShop. BLL. SystemInfo. MouldInfo();
TFShop. Model. SystemInfo. MouldInfo model
=bll. GetModelName(FormerName);
```

5.2 模板解析模块

5.2.1 功能概述

模板解析模块主要完成的任务是解析模板接收模块获取的模板中的扩展标签,并用解析后的逻辑内容替换原始模板文件中的标签。

模板解析模块首先根据模板接收模块返回的模板信息,提取其中的模板内容,并根据正则表达式匹配出模板文件中所有的扩展标签,然后根据标签所带的参数信息判断标签的类型,由该类型标签所属解析类解析该标签,将解析后获得的数据以一定的方式组织起来替换模板中的扩展标签,并将解析获得的动态数据存入标签缓存中。

5.2.2 类关联图

图 4 为模板类及模板解析类之间的类关联图。

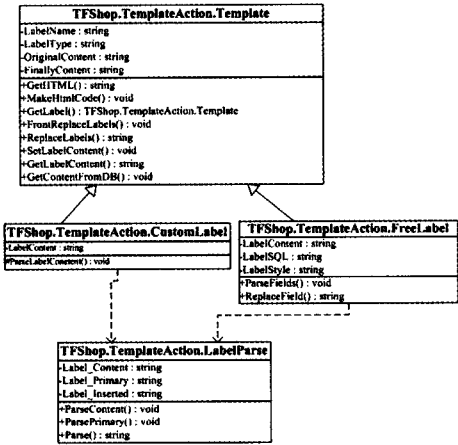


图 4 模板类及模板解析类之间的类关联图

其中,命名空间、类、成员变量、成员方法的描述如下:

TFShop. TemplateAction. Template; 动作层模板标签类

成员变量

protected stringLabelName// 标签名称

protected stringLabelType// 标签类型

protected stringOriginalContent// 原始的标签内容

protected stringFinallyContent// 最终的标签 HTML 代码

成员方法

public void FrontReplaceLabels()

// 解析模板所有的标签

public void GetHTML()

// 读取模板文件中的 html 字符串

public static Template GetLabel(string labelname, string labelType)

// 根据标签名称及标签种类并返回标签实例

public static string ReplaceLabels(string LabelContent) // 替换标签内容

public virtualvoid GetContentFromDB()

// 从数据库取得标签所包含的动态数据

public virtual void SetLabelContent(string labelContent) // 设置标签内容

TFShop. TemplateAction. CustomLabel; 普通标签类

成员变量

private string LabelContent; // 普通标签的内容

成员方法

protected void ParseLabelConetent()

// 解析普通标签内容

TFShop. TemplateAction. FreeLabel; 自由标签类

成员变量

```
private string LabelContent; // 自由标签的内容
private string LabelSQL; // 自由标签的 SQL 语句
private string LabelStyle; // 自由标签的样式
```

成员方法

```
private IList<StParam> ParseFields(string Input)
// 解析 SQL 语句的字段位置
protected string ReplaceField(string Input, string Field-
Name, string FieldValue)
// 替换 SQL 语句的字段内容
```

5.2.3 标签

标签是模板技术中最重要的组成部分,一个模板文件包装了若干个待解析的标签,一个标签就表示模板中要显示的一个内容单元. 标签由两部分组成:标签内容和标签参数,标签内容代表了该标签最终要显示的内容,即需要从数据库或者配置参数查找的字段或内容;标签参数作为标签中的重要元素,指定了标签内容的获取条件及显示样式,这样根据标签内容及标签参数,系统程序将其翻译成可以直接在浏览器中显示的 html 语言.

首先标签解析类提取标签所携带的标签参数,一个标签参数由一个键值对组成,比如下面这个标签参数 TF: LableType=ProductClass(其中 TF 为系统中标签的公共前缀)表示这个标签的类型为商品分类. 然后将提取的标签参数中的值作为查询数据库的限制条件或者标签内容的显示样式.

如下是系统中的商品分类标签:

```
[TF: Loop, TF: LableType = ProductClass, TF: ShowMode =
NoneLimit, TF: ShowType = 0, TF: Showlevel = 2, TF: Composi-
torType=0, TF: CompositorField=cid, TF: StairShowRows=5,
TF: StairShowColumns = 1, TF: TowShowRows = 4, TF: Tow-
ShowColumns=5, TF: StairCharacter = 10, TF: TowCharacter =
4, TF: WhetherOmit = 1, TF: TempletLayout = 0, TF: Product-
ClassID=undefined]
<ul>
<h3><a
href="[ # URL]">[ # StairName]</a></h3>
{ # <span><a
href="[ # URL]">[ # SecondName]</a></span> &nbsp;|
&nbsp;| &nbsp;| # }
</ul>
[/TF: Loop]
```

其中粗体部分为标签内容,URL 及 StairName 为商品的一级分类链接地址和名称,URL 及 SecondName 为该一级分类下的二级分类的分类名和链接地址. 粗体部分以外的为标签参数.

5.3 内容展现模块

内容展现模块主要将模板解析模块解析后获得的最终模板内容展现给访问者,最终的模板内容是一个 HTML 格式的字符串,实现起来很简单,调用 Response. Write(string template content)方法就可以将最终内容输出到客户端. 无需数据库的支持. 有助于提高页面的访问速度,这些扩展的 HTML 页面,根据所属模板的类型,在模板初始化时就被配置到指定的服务器目录下,从而减少了服务器的配置,提高了系统的执行效率.

6 结束语

本文详细介绍了模板技术在内容管理系统中的应用,针对当前主流模板引擎在解析效率上的不足,根据所开发的内容管理系统的功能架构,自主开发了一套模板引擎,并使用了缓存技术来提高模板的解析效率和页面的访问速度. 实测表明,我们所开发的这套模板引擎结合缓存技术,提升了系统的执行效率,增强了系统的可用性.

参考文献:

- [1] 史玉慧,张晓防,麦中凡. 从数据管理到内容管理——企业门户核心技术研究[J]. 计算机工程与应用,2001,37(13):143-146.
- [2] 孔芳,周建美. 基于内容管理的合同管理方案的设计[J]. 微电子学与计算机,2007,24(6):211-213.
- [3] Nousath S, Hemantha Kumar G, Shiva kumara P. (2D)2 LDA: An efficient approach for face recognition[J]. Pattern Recognition, 2006, 39(7): 1396-1400.
- [4] 未知. JAVA 模板引擎. [EB/OL]. (2005-02-28)[2011-08-29]. <http://www.moon-soft.com/doc/41561.htm>.
- [5] 喻斌. 内容管理系统中模板技术的研究与应用[D]. 南昌:南昌大学,2007.
- [6] Joachim Rossberg, Rickard Redler. 设计可扩展的. NET 应用程序[M]. 北京:清华大学出版社,2005.
- [7] 施珺,顾勋梅. 基于 MVC 的网络化学习云平台架构[J]. 微电子学与计算机,2011,28(10):24-27.
- [8] 夏纯中. 轻量级企业内容管理系统的设计与实现[J]. 计算机工程与设计,2007,28(17):4235-4236.

作者简介:

南 轶 女,(1986—),硕士研究生. 研究方向为软件工程、网络软件.

李先国 男,(1960—),副教授,博士. 研究方向为软件工程、网络软件.