[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/30 (2006.01)

[12] 发明专利说明书

专利号 ZL 200510128896.2

[45] 授权公告日 2009 年 5 月 20 日

[11] 授权公告号 CN 100489849C

[22] 申请日 2005.12.6

[21] 申请号 200510128896.2

[30] 优先权

[32] 2005. 1. 6 [33] US [31] 11/030,423

[32] 2005. 2.25 [33] US [31] 11/066,083

[73] 专利权人 微软公司

地址 美国华盛顿州

[72] 发明人 T·A·戴维斯 B·M·琼斯

A·塔勒吉哈尼 R·A·利特尔

M·萨维斯基 M·森德兰德

[56] 参考文献

US20040021679A1 2004.2.5

WO0108033A2 2001.2.1

US20030163603A1 2003.8.28

US20020010716A1 2002.1.24

审查员 李 楠

[74] 专利代理机构 上海专利商标事务所有限公司

代理人 张政权

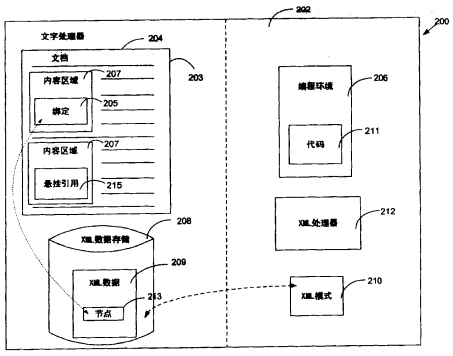
权利要求书 2 页 说明书 21 页 附图 4 页

[54] 发明名称

用于绑定数据的可编程性

[57] 摘要

一种对象模型允许使用编程环境来开发代码以访问例如文字处理应用程序等应用程序中的功能。该对象模型可被用来操纵应用程序中的一个或多个数据绑定。 还可使用编程环境来开发代码，用于反应于任一方向上的对与一个或多个数据绑定相关联的内容区域或 XML 数据中的节点的改变。 可开发代码以定义文件内容与数据存储中的数据内的内容之间的数据绑定。 此外，可开发响应于文件的绑定区域内或数据存储内的改变的代码，以捕获或截取诸如编辑、添加、删除等事件。



1．一种使用数据绑定的方法，所述数据绑定链接数据存储和内容区域，所述方法包括：

展示数据绑定接口，

使用与所述数据绑定接口相关联的方法、属性或事件，其中，所述方法、属性或事件返回允许对所述数据绑定、数据或与所述数据绑定相关联的其它对象的操纵的对象，

为所述数据绑定提供 XPath 和 XML 数据，

确定一特定数据绑定是否为悬挂引用，并且如果是，则等着看期望的数据是否出现在所述数据存储中。

2．如权利要求 1 所述的方法，其特征在于，还包括使用属性来定位与特定数据绑定相关联的 XML 数据。

3．如权利要求 2 所述的方法，其特征在于，还包括使用代码来操纵所述 XML 数据。

4．如权利要求 1 所述的方法，其特征在于，还包括使用属性来定位 XML 数据内的目标节点。

5．如权利要求 4 所述的方法，其特征在于，还包括使用代码来操纵相关联的数据绑定的目标节点。

6．如权利要求 1 所述的方法，其特征在于，还包括使用返回用于移除特定数据绑定的对象的方法。

7．如权利要求 1 所述的方法，其特征在于，还包括使用返回用于创建或改变内容区域上的数据绑定的对象的方法。

8．如权利要求 7 所述的方法，其特征在于，还包括为特定链接提供目标 XML 节点。

9．如权利要求 1 所述的方法，其特征在于，还包括使用返回表示用来创建内容区域上的数据绑定的信息的对象的方法。

10．一种用于提供对数据存储和内容区域之间的链接的访问的系统，所述系统包括：

数据绑定接口，用于展示与所述数据绑定接口相关联的方法、属性或事件，

其中，所述方法、属性或事件返回允许对所述数据绑定、数据、或与所述数据绑定相关联的至少一个对象的操纵的对象；

用于为所述数据绑定提供 XPath 和 XML 数据的装置；以及

用于确定一特定数据绑定是否为悬挂引用，并且如果是，则等着看期望的数据是否出现在所述数据存储中的装置。

11．如权利要求 10 所述的系统，其特征在于，所述数据绑定接口用于返回用于定位 XML 数据内的目标节点的对象。

12．如权利要求 10 所述的系统，其特征在于，所述数据绑定接口用于返回用于创建或改变内容区域上的数据绑定的对象。

13．如权利要求 10 所述的系统，其特征在于，所述系统还用于提供用于开发与相关联的数据绑定联合工作的代码的接口。

用于绑定数据的可编程性

相关申请的参照

本申请是微软公司所拥有的、于2005年1月6日向美国专利商标局提交的，题为“Data Binding In A Word-Processing Application”（文字处理应用程序中的数据绑定），代理人档案号60001.0459US01/MS310225.1的专利申请的部分继续申请，其全部内容通过引用包含于本文中。

本申请涉及题为“METHOD AND APPARATUS FOR UTILIZING AN OBJECT MODEL FOR MANAGING CONTENT REGIONS IN AN ELECTRONIC DOCUMENT”（利用对象模型管理电子文档中的内容区域的方法和装置），代理人档案号14917.0059USI1/MS310649.01，以及题为“MOTHOD AND APPARATUS FOR UTILIZING AN EXTENSIBLE MARKUP LANGUAGE SCHEMA FOR MANAGING SPECIFIC TYPES OF CONTENT IN AN ELECTRONIC DOCUMENT”（利用可扩展标记语言模式管理电子文档中特定类型的内容的方法和装置），代理人档案号14917.0060USI1/MS310650.01的专利申请，上述两个专利申请均为微软公司所拥有，并与本申请同时提交，且其全部内容通过引用包含于此。

背景技术

近年来标记语言大行其道。一种类型的标记语言，即可扩展标记语言（XML），是一种提供标识、交换和处理各种类型的数据的方法的通用语言。例如，XML被用来创建可由各种应用程序使用的文档。XML文件的元素通常具有相关联的名字空间和模式。

名字空间是XML文档中用来定义元素/属性名字和类型的名字集合的唯一标识。名字空间的名字通常被用来唯一地标识每个XML文档类。该唯一的名字空间区分来自不同的源但恰巧具有相同名字的标记元素。

XML模式提供一种在XML环境中描述和验证数据的方法。模式陈述什么元素和属性被用来描述XML文档中的内容，每个元素在哪些地方被允许，什么类型

的内容在其内被允许，以及哪些元素可在哪些其它元素内出现。模式的使用确保文档以一致和可预测的方式被结构化。模式可由用户创建，并且一般受到诸如 XML 等相关联的标记语言的支持。通过使用 XML 编辑器，用户可操纵 XML 文件，并生成符合该用户所创建的模式的 XML 文档。在先前的文字处理器应用程序中，对自定义的 XML 模式的支持被添加到应用程序中，以使用户能够用自定义的 XML 标记（例如，<title>）来‘标记’文档的内容，这本质上给了先前未归类的一连串文本语义意义。这意味着，先前只是具有格式编排、但没有意义供其它应用程序处理的文本如今可以是包含特定 XML 标记片段的结构化的 XML 文档，这些 XML 标记片段来自任何用户定义的 XML 模式，且任何其它知道 XML 的应用程序都可定位和理解该 XML 模式。

在一基本示例中，文档顶部的文本可用来自用户定义的 XML 模式的 <title> XML 元素‘标记’为标题，这意味着其它知道 XML 的应用程序如今可容易地理解此范围的文本包含一“标题”并可正确地提取该标题。这使得后端过程能够用适当的语义和上下文智能地提取文档的部分（例如，此文本是 <title> ）。

但是，与先前的文字处理器应用程序相关联的缺点源自以下事实，即自定义的 XML 标记的添加和持久化依赖于文档的呈现。即，在现有实现中，在文字处理器文档的 XML 标记（例如，以 XML 格式表达的顾客发票的细节）和它在文档表面上的呈现（例如，三页明文接下来是 4 行 5 列的 w/a 特定表格样式的表格）之间有不可避免的关联。因此，先前的文字处理器应用程序中所表示的 XML 数据（因为它依赖于呈现）必需与文档的内容严格一致。例如，如果发票所用的 XML 模式陈述<date>出现在<address>之前，而<address>出现在<phoneNumber>之前，则这三个 XML 元素必需严格按照文档中所呈现的次序出现。这意味着呈现格式的改变（例如，移动包含<date>的表格行）还会引起该文档中所包含的 XML 数据的结构的改变，这要求解决方案开发者一方额外的步骤来确保此数据符合相关联的 XML 模式的结构。由此，文档的终端用户没有只有操纵呈现的自由，因为这样做可能总是改变数据的语义，从而潜在地违反该数据的 XML 模式。

此外，在先前的文字处理器应用程序之上开发的解决方案在试图为后端应用程序读/写来自文档的数据时，需要更仔细地考虑呈现的蕴含。因此，如果一段粗体文本被标记为标题，则先前的文字处理器应用程序所保存的结果 XML 看起来将会是以下形式：

<w:p>

```
<Title>
    <w:r>
        <w:rPr>
            <w:b/>
        </w:rPr>
        <w:t>This is the title.</w:t>
    </w:r>
</Title>
<w:p>
```

如上所示，自定义的 XML 标记两边被非常专属于先前的文字处理器应用程序的 XML 标记（在此例中为 w:p、w:r 等）所包围。这意味着正在处理此数据的知道 XML 的解决方案不仅必须理解其自身的数据格式（包含<Title>元素），还必须理解先前的文字处理器应用程序格式编排的确切细节，从而当它搜索自己的数据时知道遍历并忽略该信息。由此，此类实现仍然对用户强加一些要求，因为文档中的文本的外观的小小改变（例如，将<Title>元素的内容拖到表格单元中，等等）可能导致该自定义 XML 标签的位置在周围环境的文字处理器的本机标签内部的显著改变。因此，程序员/代码开发者常常需要编写附加代码，以预期和理解先前的文字处理器应用程序将基于呈现而把自定义的 XML 元素放在何处，以及处理所有这些各种变换。这意味着最终的解决方案可能仍需包含大量逻辑代码以处理特定的先前的文字处理器应用程序的需求。

处理现有的文字处理器应用程序的程序员/代码开发者在考虑读和写操作时，还需考虑文档布局格式的蕴含。例如，用户可能试图攫取<StockSymbol>元素的值，并使用该值以将公司的全名放到同一文档的<CompanyName>元素中，来作为用户写公司报告的简单增强。为维护文档的完整性，用户在能够写功能代码以执行这些动作以前，需要同时对来自文档的期望数据的读和写两方面考虑文档的当前布局格式。例如，用户可能需要知道他们所写的值是否在表格单元中、着重列表中、等等，来构造现有文字处理器应用程序的格式编排信息，从而当这些信息被插入到文档中时将产生期望的结果。这是进行附加编码以理解文字处理器应用程序的呈现语义的另一个潜在的原因。

现有文字处理器应用程序的又一个局限性是有时会发现 XML 元素的编辑行为是“脆弱的”。这部分地是因为，如上所述，它们受到以下事实的限制，即标记在文档表面上的定位基于用户定义的模式来确定 XML 数据的结构。由此，可能产

生若干问题。首先，根据相关联的 XML 模式，典型的用户操作（例如，从一个部分复制/粘贴到另一个部分）可能会改变 XML 结构并使文档变为无效。第二，在此类文字处理器实现中，用户定义的 XML 模式所要求的所有元素都需要以某种形式被包括到文档表面上。这意味着开发者要创建相关联的 XML 数据以作为携带关于文档的附加信息（不在文档的表面上显示，而是用作元数据）十分困难。第三，文档表面上的语义上非必要的元素（例如，并不标记混合内容的非叶元素）也需要被包括到此类文字处理器实现中，这进一步增加了普通用户操作修改 XML 数据的能力。

在许多情形中，定义 XML 数据（例如，包括备忘录文档的数据）的模式趋向于由单个标准体严格定义，以便于此数据在多个异类处理系统之间的通信。但是，在如此便于后端通信的同时，文档数据的人类可阅读性和可编辑性往往被牺牲，这使得用户难以理解此数据及对其进行语法分析。例如，XML 标准可定义日期的标准格式，诸如：dd-mm-yyyyThh:mm:ss.ssss。所有日期都被要求以此格式表示以由知道 XML 的应用程序进行语法分析。显然，用户很难正确输入此格式，并且它常常与用户通常输入日期的方式（例如，许多场合通常使用 mm-dd-yyyy 而不是 dd-mm-yyyy，等等）冲突。

由此，需要有一种方法使开发者能够在诸如文字处理器应用程序等应用程序中将 XML 数据与此类数据的呈现分开。

发明内容

本发明的实施例提供一种包括模式的文字处理器应用程序，用于创建文字处理器文档，其中数据和呈现可被分开。更具体地，数据可被输入到文字处理器文档中，并可从其中被提取，且该文字处理器文档与其呈现格式存储在分开的位置。根据本发明的实施例，该文字处理器应用程序的用户可为文字处理器文档数据创建单独的存储位置，并建立该数据的内容与呈现表面之间的链接（或绑定）。

根据本发明的实施例，一种对象模型允许使用编程环境来开发代码。该代码可被用来操纵应用程序中的一个或多个数据绑定。还可使用编程环境来开发代码，以对 XML 数据中的内容区域或节点任一方向的改变作出反应。还可开发代码以定义文件内容和数据存储内的内容之间的关系。此外，可开发对文件的绑定区域内或数据存储内的改变作出反应的代码，以捕获或截取诸如编辑、添加、删除等事件。本发明的实施例使用户能够用相关联的 XML 单次编写代码，且如今该代码可移植

到支持使用 XML 构造的文件类型，而无需担心目标应用程序的确切语义，由此大大简化并提高了应用程序的开发的效率。

附图说明

图 1 示出可在本发明的一个示例性实施例中使用的示例性计算设备；

图 2 是示出用于实施本发明的示例性环境的框图；

图 3 是根据本发明的一个实施例的流程图；以及

图 4 是示出用于实施本发明的另一个示例性环境的框图。

具体实施方式

贯穿整个说明书和所附权利要求书，以下术语采取本文中明确相关联的含义，除非上下文中明显另有所指。

术语“数据”是指随文字处理器文档携带的、文字处理器文档本身、或由其使用的任何补充信息。此信息通常很大，并且很可能并不是在文档的呈现层上完整地展现。

术语“标记语言”或“ML”是指文档内的特殊代码的语言，它指定文档的各个部分要如何由应用程序来解释。在文字处理器文件中，标记语言指定文本要如何被格式化或布局。

术语“元素”是指 XML 文档的基本单元。元素可包含 XML 文档的属性、其它元素、文本、以及其它内容区域。

术语“呈现”是指文档的可视化部分——即在文档被打印的情况下将会显示出来的文本和布局。

术语“标记”是指被插入到文档中的、在 XML 文档内界定元素的字符。每个元素仅可有不超过两个的标记：开始标记和结束标记。有空元素（无内容）是可能的，在此情形中允许仅有一个标记。

标记之间的 XML 内容被视为该元素的“子元素”（或后代）。由此，嵌入到该元素的内容中的其它元素被称为该元素的“子元素”或“子节点”。直接嵌入到元素的内容中的文本被称为该元素的“子文本节点”。概括而言，元素内的子元素和文本构成该元素的“内容”。

术语“属性”是指设为特定值并与元素相关联的附加属性。元素可具有与其相关联的任意个数的属性设置，包括无属性设置的情形。属性被用来将附加信息与

不包含附加元素、或被视为文本节点的元素相关联。

术语“内容区域”是指文档被界定的和/或任选地被标记的区域，它起到用户所输入的类型的内容的容器的作用。详见由本发明的受让人于 2004 年 9 月 30 日提交的题为“Methods, System, and Computer-Readable Medium For Managing Specific Types of Content In An Electronic Document”（用于管理电子文档中的特定类型的内容的方法、系统和计算机可读介质）的已转让的美国专利申请第 10/955,612 号，其全部内容通过引用包含于此。

“XPath”是一个算子，它使用模式表达式来标识 XML 文档中的节点。XPath 模式是用斜杠分开的子元素名列表，它描述贯穿 XML 文档的路径。模式“选择”匹配该路径的元素。

术语“XML 数据存储”是指文字处理器文档内的一个容器，它提供访问，以供在文件打开时存储和修改存储在文字处理器文档中的数据（例如，XML 格式）。

术语“数据绑定”是指内容区域上的一个属性，它确定文字处理器文档内的一个或多个 XML 数据片段中可存储内容区域的内容的 XPath 位置。如本文中所使用：

“ref” – 指引用由个体绑定所使用（在打开/保存文档到文件时使用）的以下属性集合的中央存储的唯一整数；

“ID” – 指指定 XML 数据存储内的特定 XML 数据的唯一 ID；

“selectionNamespaces” – 指 XML 数据存储中的相关联 XML 文档的前缀映射关系（它将名字空间与简短的缩写相关联）；以及

“rootURI” – 指 XML 数据存储中的相关联的 XML 文档的根名字空间。

示例性操作环境

本发明的实施例提供一种用于创建文字处理器文档的文字处理器应用程序，其中 XML 数据的存储和呈现可被分开。更具体地，可被输入到文字处理器文档中、并可从其中被提取的数据与该文字处理文档的呈现格式存储在分开的位置。由此，该文字处理器应用程序的用户可为文字处理器文档内所包含的XML数据创建单独的存储位置，并建立该数据的内容和呈现表面之间的链接（或绑定），以使用户能够通过编辑该呈现的内容来编辑相关联的 XML 数据，而以相同方式防止用户改变相关联的 XML 数据的结构。例如，发票所用的数据可以文字处理器文件格式单独存储为 XML，从而移动链接在文档中的位置不会改变该单独的数据存储的结构。由此，令此结构化数据的后端处理变得更加简单，因为该数据如今有已知的结构，

而该结构不受用户编辑文档的方式所影响。用户可在文字处理器文档中编辑数据、格式化数据（包括富呈现的格式化）、等等，仅对文本内容的改变被‘下推’回该文本背后所存储的 XML 数据。但是，根据本发明，经由用户与文字处理器文档的交互进行更新的所有数据都在原始的本机 XML 流中可用。因此本发明还实现通过直接改变所链接的 XML 数据来修改文字处理器文档的内容，而无需处理该数据呈现格式（同样可能不断改变）的复杂问题。这样做大大简化了文字处理器文档中的结构化数据的添加、编辑和提取。

参考图 1，一种用于实现本发明的示例性系统包括诸如计算设备 100 等计算设备。在非常基本的配置中，计算设备 100 通常包括至少一个处理单元 102 和系统存储器 104。取决于计算设备的确切配置和类型，系统存储器 104 可以是易失性的（诸如 RAM）、非易失性的（诸如 ROM、闪存、等等）、或这两者的某种组合。系统存储器 104 通常包括操作系统 105、一个或多个应用程序 106，并可包括程序数据 107。在一个实施例中，应用程序 106 可包括文字处理器应用程序 120。此基本配置在图 1 中由虚线 108 内的组件示出。

计算设备 100 可具有其它特征或功能。例如，计算设备 100 还可包括诸如磁盘、光盘、或磁带等其它数据存储设备（可移动和/或不可移动的）。此类其它存储在图 1 中由可移动存储 109 和不可移动存储 110 示出。计算机存储介质可包括以用于存储诸如计算机可读指令、数据结构、程序模块、或其它数据等信息的任何方法或技术实现的易失性和非易失性、可移动和不可移动的介质。系统存储器 104、可移动存储 109 和不可移动存储 110 都是计算机存储介质的示例。计算机存储介质包括，但不限于，RAM、ROM、EEPROM、闪存或其它存储器技术，CD-ROM、数字多功能盘（DVD）或其它光存储，磁带盒、磁带、磁盘存储或其它磁存储设备，或可用来存储所需信息并可由计算设备 100 访问的任何其它介质。任何此类计算机存储介质都可以是设备 100 的部件。计算设备 100 还可具有诸如键盘、鼠标、笔、语音输入设备、触摸输入设备等输入设备。还可包括诸如显示器、扬声器、打印机等输出。在本领域中，这些设备是公知的，因此无需在本文中详细讨论。

计算设备 100 还可包含通信连接 116，它允许该设备诸如通过网络等与其它计算设备 118 通信。通信连接 116 是通信介质的一个示例。通信介质通常可具体化为诸如载波或其它传输机制等已调制数据信号中的计算机可读指令、数据结构、程序模块或其它数据，并且包括任何信息传递介质。术语“已调制数据信号”是指以在信号中将信息编码的方式设置或改变其一个或多个特征的信号。作为示例，而非限

制，通信介质包括诸如有线网络或直接连线连接等有线介质，以及诸如声学、RF、红外及其它无线介质等无线介质。如本文中所使用的术语计算机可读介质既包括存储介质又包括通信介质。

计算设备 100 的系统存储器 104 中可存储若干程序模块和数据文件，包括适用于控制联网的个人计算机的操作的操作系统 105，诸如来自华盛顿州雷蒙德市微软公司的 WINDOWS XP 操作系统等。系统存储器 104 还可存储诸如文字处理器应用程序 120 以及下述的其它程序模块等一个或多个程序模块。文字处理器应用程序 120 用于提供创建、编辑和处理电子文档的功能。

根据本发明的一个实施例，文字处理器应用程序 120 包括来自微软公司的 WORD 程序。但是，应当认识到，可使用来自其它制造商的文字处理器应用程序来具体化本发明的各个方面。还应认识到，本发明的各个方面不仅局限于文字处理器应用程序，而是还可使用能够处理各种形式的内容（例如，文本、图像、图片、等等）其它应用程序 106，诸如数据表应用程序、数据库应用程序、演示应用程序、绘图或计算机辅助应用程序、等等。

本发明的实施例可以实现为计算机过程、计算系统、或实现为诸如计算机程序产品或计算机可读介质等制造品。计算机程序产品可以是计算机系统可读的、并且将用于执行酒商过程得计指令的计算机程序编码的计算机存储介质。计算机程序产品还可以是载波上所传播的计算系统可读的、并且将用于执行计算机过程的指令的计算机程序编码的信号。

在文字处理应用程序中绑定数据

图 2 是示出用于实施本发明的实施例的示例性环境的框图。图 2 中所示的示例性环境是文字处理器环境 200，它包括文字处理器应用程序 202、文字处理器文档 204、编程环境 206、数据存储 208、模式文件 210、以及 XML 处理模块 212。但是，如上所述，本发明还可适用于能够处理各种形式的内容（例如，文本、图像、图片、等等）的其它应用程序 106，诸如数据表应用程序、数据库应用程序、演示应用程序、绘图或计算机辅助应用程序、等等。编程模块 206 可为 XML 处理模块 212 提供简单的应用程序编程接口（API），它允许开发修改文档 204 或者 XML 数据存储 208 的内容的代码。可以认识到，并不试图以本文中所描述的任何特定实施例或示例来限制本发明。例如，文字处理器环境可包括多个文字处理器文档 204、数据存储 208、和/或模式文件 210。根据本发明的一个实施例，文字处理器应用程序 202 使用 XML 处理模块 212 以处理根据可扩展标记语言格式化的数据。一种合

适的 XML 处理模块 212 是华盛顿州雷蒙德市微软公司所制造并销售的 MSXML。

文字处理器应用程序 202 包括其自身的一个或多个名字空间、以及模式 210 或模式集合，定义这些是为配合于文字处理器应用程序 202 相关联的文档 204 使用的。模式 210 为文字处理器应用程序 202 定义的标记和属性的集合定义了文档 204 的格式。如以下将描述，并根据本发明的实施例，数据存储 208 可包括数据 209。较佳的是，模式 210 依附于数据存储 208 内的数据 209。文字处理器文档 204 还包括如下所述的由用户创建的内容区域 207。可包括一个以上的数据存储 208、相关联的 XML 数据 209 和模式 210，以作为文字处理器应用程序 202 的部件。为了提供带有管理可能被包括在给定 XML 数据 209 中的数据的类型和结构的语法和数据类型规则的集合，可将一个或多个 XML 模式 210 与 XML 数据 209 相关联，以提供管理用户可用来注释给定 XML 数据 209 的每一个 XML 元素和标签的规则。模式 210 包括管理将那些元素应用于 XML 数据 209 的次序的规则、以及与应用于 XML 数据 209 的个体元素相关联的特定规则。

本发明的实施例提供一种用于创建文字处理器文档 204 的文字处理器应用程序 202，其中数据和呈现可经由随文字处理器文档存储的单独的数据存储 208 的存在而被分开。更具体地，可输入到文字处理器文档 204 中或从其中被提取的数据被存储在文档的数据存储 208 内的一个或多个 XML 数据 209 文件中，由此将数据与文字处理器文档 204 的呈现格式分开。由此，文字处理器应用程序 202 的用户可为文字处理器文档 204 的数据创建单独的存储位置，并在该数据的内容与呈现表面 203 之间建立与一个或多个内容区域 207 相关联的链接（或绑定）205，这使得用户能够通过编辑呈现的内容来编辑数据，而以相同方式防止用户改变数据 209 的结构。在文档 204 中移动内容区域 207 的位置不会改变单独数据存储 208 中的 XML 数据 209 的结构。此外，对数据的呈现（如粗体、斜体、对齐等）所作的改变不影响数据结构。由此，结构化数据的后端处理被简化，因为 XML 数据 209 对应于不受用户编辑文档 204 的方式影响的已知结构。

本发明的实施例通过直接改变所链接的数据来实现文字处理器文档 204 的内容的修改，而无需处理可能不断改变的呈现格式的复杂问题。这样做大大简化了文字处理器文档 204 中的结构化数据的添加、编辑和提取。此外，绑定到结构化 XML 数据 209 的数据绑定 205 可在文档 204 中到处移动，而不会影响数据的结构。较佳的是使用 XPath 表达式来实现内容区域 207 上的数据绑定 205，该表达式可经由用户接口或编程窗口 206 来定义。

用户使用 XPath 表达式（一种用于标识 XML 树中的节点 213 的标准 XML 方法）唯一地标识文档内容区域应被绑定到的期望的 XML 节点 213。文字处理器应用程序 202 对 XPath 进行自动分析，并使用 XPath 来为数据绑定区域定位期望的目标。这也意味着熟悉 XPath 标准的开发者可控制 XML 的这一用途来创建本质上半动态的数据绑定 205。即，基于对数据 209 的其它改变或对呈现 203 的改变来标识不同的目标节点 213。例如，假定用户想要显示在给定时间段里完成最高销售额的雇员的名字。如果此信息在与文档 204 相关联的 XML 数据 209 中，则用户可创建链接到具有最高的已完成单数的个人的名字的 XPath 表达式，并且当数据改变时，该链接自动转移到适当的位置（节点 213）。还可通过使用代码 211、用户界面、或编程环境 206 在节点 213 之间改变链接。

或者，用户可创建唯一标识数据 209 中文档内容区域 207 应被绑定到的对象表示节点 213 的数据绑定。文字处理器应用程序 202 自动确定 XPath，来为数据绑定区域定位期望的目标。但是这意味着在此情形中，数据绑定 205 将自动更新其 XPath 以确保数据绑定 205 指向同一对象，这与同一 XPath 表达式相反。

可开发如上文所简述的编程代码 211 以使用 XML 处理模块 212，以及对向任一方向（即，从文档表面 203 上的内容区域 207 到数据存储 208 中的 XML 数据 209 中的节点 213，反之亦然）移动的改变作出反应。用户可开发定义文档表面 203 和数据存储 208 内的特定内容之间的关系的代码 211。此外，可开发对文档 204 的绑定区域内或数据存储 208 内的改变作出反应的代码，以捕获或截取诸如编辑、添加、删除等事件。例如，假定用户想要确保至多只有一个文档可使用某个特定的标题。代码 211 可基于输入到标题节点中的内容检查中央数据库，以确定该标题是否已被使用。如果该标题已被取用，则代码 211 可提示用户输入另一个标题和/或警告用户该标题不可用。本发明的实施例使用户能够用相关联的 XML 一次性编写代码 211，而该代码如今可被移植到支持 XML 构造的使用的所有文档类型，而无需担心目标应用程序的确切语义，由此大大简化并提高了应用程序开发的效率。

根据本发明的实施例，可用表示特定文本区域（例如，标题、正文、等等）的语义的内容区域 207，并通过添加数据绑定 205 来标记文字处理器文档 204，如今该内容区域 207 内的相关联文本被存储在文档 204 内的数据存储 208 中的某个 XML 数据 209 内的节点 213 中。使用一个或多个数据绑定 205，数据 209 被链接到文档中的内容区域 207，即，被标记的区域。由此，如今数据 209 可被存储在其自己的 XML 流中的一致的位置（"数据存储"），而无论数据绑定 205 将相关联

的内容定位到文档 204 的呈现 203 中的何处（即，用户编辑特定文档时与之交互的数字表示，例如描绘特定用户文档的 WORD 窗口等），或者如何呈现数据 209。由此，用户不必担心数据在文档 204 中被到处移动，因为 XML 数据 209 如今位于一个一致的位置中的数据存储 208 内。此外，数据存储 208 可包含未被文档 204 内的数据绑定 205 所引用的数据 209。诸如元数据等这些“额外”的数据向诸如解决方案开发者等用户提供可能与文档用户无关的附加信息。

根据本发明的实施例，数据的结构被保存在单独的位置（文档的数据存储 208 内的一个或多个 XML 数据片段 209）中，从而使用户能够将链接（例如，数据绑定 205）在呈现 203 中到处移动，而不会影响数据结构。由此，XML 数据 209 的结构不会改变，只是与文字处理器文档 204 相关联的 XML 数据 209 的呈现有所改变。因此，改变文档 204 中的数据呈现的格式不影响数据存储 208 的结构。用户操纵文档表面 203 并不移动实际数据 209——因此用户对呈现具有完全的控制，而无需考虑单独在存储 208 中维护的数据 209 的分支。由此，本发明的实施例允许用户与呈现信息分开访问自定义 XML 信息。

可使用一个或多个数据绑定 205 将数据源的内容（数据存储 208 中的 XML 数据 209，本文中称为 XML 数据 209）绑定到文档 204 中的位置。如本文中所使用，XML 数据 209 包括诸如文本（明文或富格式化的文本）、图像、特定类型（日期）的内容等任何类型的内容。可将数据绑定 205 的结构描述为 XPath 链接，它允许文字处理器应用程序 202 与和文档 204 相关联的 XML 数据存储 208 中的 XML 节点 213 连接/同步，或将到其的链接持久化。最佳的是，XML 数据存储 208 是文档 204 的部件（即，存储 208 是随文字处理器文件保存，并随特定文档 204 移动或与其相关联）。数据绑定 205 还可包含控制应如何在呈现（例如，文字处理器的文档表面 203）和数据存储 208 之间变换数据的信息。文字处理器应用程序 202 的用户可允许存储在数据存储 208 中的数据 209（由后端应用程序操纵的数据）被以标准格式存储，但在文字处理器文档 204 中以较友好的格式呈现相同的信息（例如，用户看到 1 月 29 日 2004 年，但该数据是以 29-01-004T12:00:00.0000 的 dateTime 格式存储的）。又如，数据绑定信息可包含图像——对数据 209 而言此图像被表示为看起来无意义的字符串，但上述的同一变换原理意味着用户将在文字处理器文档 204 的内容区域 207 中看到一个图像，用户可添加/改变图像，且 XML 编码的表示将被持久化为 XML 数据 209，以使任何后端过程可存储/操纵该信息。

根据一个实施例，当用户向内容区域 207 添加数据绑定信息 205 时，用户通

过指定 XPath 表达式来提供关注的所链接的 XML 数据 209（例如，标识一个或多个节点 213）。一经绑定，此内容区域 207 的内容即将被链接或绑定到该 XPath 所返回的节点 213 的内容（XML 数据）。因此，这意味着如果 XML 节点 213 是以 XPath 所返回的 XML 节点 213 改变这样一种方式被添加/移除/改变，则文档 204 中的内容区域 207 的内容自动更新。或者，如果发生导致特定数据绑定 205 不返回任何 XML 节点 213 的改变，则该数据绑定 215 进入以下所描述的‘悬挂引用’状态。

例如，假定文档 204 包括以下段落，其中“Microsoft Corporation”对应于明文内容区域 207（斜体所示），该内容区域 207 被绑定到该文本的数据存储 208 内的某个所链接 XML 数据 209 中的 XPath/contract(1)/header(1)/company(1)。显示在呈现 203 上的段落是：

“*Microsoft Corporation* is located at One Microsoft Way.”

根据一个实施例，可通过在编程环境 206（例如）中指定单行代码来建立链接：

Document.ContentRegion.Add().DataBinding.Add(“/contract(1)/header(1)/company(1)”)

对应的所链接的 XML 数据 209 看起来可能是（到节点 213 的链接以单引号示出）：

```
<contract>
    <header>
        ‘<company>Microsoft Corporation</company>’
        <company>Contoso Corporation</company>
    </header>
</contract>
```

假定现在用户使用数据存储 208 API 来添加新的<company>节点 213，作为<header>的第一个子节点（新节点 213 以单引号示出）：

```
<contract>
    <header>
        ‘<company>Fabrikam Corporation</company>’
        <company>Microsoft Corporation</company>
        <company>Contoso Corporation</company>
    </header>
```

</contract>

内容区域 207 上最终的绑定 205 仍然是绑定到同一 XPath（"/contract(1)/header(1)/company(1)"），因此文档内容可被即时更新以显示该节点 213 的新内容：

"*Fabrikam Corporation* is located at One Microsoft Way."

根据本发明，如果文字处理器文档 204 的一个或多个区域包含数据绑定的内容区域 207，则文档 204 反应于对任一所链接的内容源的改变。由此，如果文档 204 的一个范围是数据绑定的，则改变相关联的 XML 数据 209 中的 XML 节点 213 的内容将会导致内容区域 207 的文本的自动改变。由此，如果文档 204 的一个范围是数据绑定的，则改变文档 204 中该绑定的内容区域 207 会导致对应的 XML 数据 209 中的 XML 节点 213 的内容的自动改变。即，具有完全相等的绑定 205 的多个内容区域 207 可存在于文档 204 的多处。例如，具有到一名字的数据绑定 205 的内容区域 207 可被添加到文档的头部以及正文。改变这些位置中的任何一个会将该文本与 XML 数据存储 208 同步，后者将进而在文档 204 中存在具有到该节点 213 的数据绑定 205 的内容区域 207 的任何地方反映该改变。

XML 数据 209 中的 XML 节点 213 可与文档 204 有一对多的关系，这意味着 XML 数据 209 中的同一 XML 节点 213 可被多个数据绑定 205 所引用。只要文档 204 中数据绑定的内容区域 207 一被更新，它即引起 XML 数据 209 中适当的 XML 节点 213 的改变，这进而使文档 204 中的其它内容区域 207 中的所有其它相关联的绑定 205 被以该新文本更新。例如，假定文档 204 的头部中的内容区域 207 包含指定某个 XML 数据 209 中的<title/>节点的 XPath 表达式的数据绑定 205，且文档 204 的正文中的另一个内容区域 207 也包含到同一元素的数据绑定 205。根据本发明，这两者都会显示相同的内容，即使它们具有不同的格式编排。如果用户编辑文档 204 的正文中的内容区域 207 中的内容，则该更新会被持久化到数据存储 208 中的适当的 XML 数据 209 中的适当的 XML 节点 213，从而引起文档 204 中也指定该 XML 节点 213 的、具有相关联的绑定 205 的所有其它内容区域 207（例如，页眉中，页脚中，等等）的更新。本发明的实施例提供一种用于将文档中的多个位置绑定到数据存储 208 中的单个 XML 节点 213 的机制，正如本例中将所有这三个位置的内容链接到单个数据源。由此，文档 204 中被链接到 XML 数据 209 中的同一节点 211 的所有内容区域 207 的内容是完全相等的。

对此的一个示例性例子是典型的报告文档，其中用户通常使标题在若干位置

显示：在封面上（以大号粗体文本），在页眉中（以较小的文本），以及在页脚中（以较小的斜体文本）。通常，用户将必须在每个位置打出标题，还要确保如果这三个位置中的任何一个的标题被改变，则他们要记得在其它两个位置改变标题（以保持内容一致）。但是，往往很容易忘记要使所有这三个位置保持同步。根据本发明的实施例，一旦包含用户想要在文档204中显示的XML数据209的数据存储208就位，文档中的多个位置（例如，上述的三个位置）全部都可以是绑定到数据存储208中的单个XML节点213的内容区域207数据。

因此，所有这三个位置的内容被链接或绑定到单个数据源209。这意味着用户通过改变这些区域中的任何一个（例如，封面）的内容，将自动使用户的文本被下推到底层的XML数据209，然后进而被下推到文档204中具有带对应的数据绑定205的内容区域207的其它位置（在此情形中为页眉和页脚）。这意味着如今用户在他们与文档的交互的范围内已链接或绑定了这三个内容范围以使它们完全相等。根据本发明的实施例，文档的各个区域能以多种方式呈现（大号粗体、小号斜体、等等），但数据存储208中的数据结构保持不变。

悬挂引用

根据本发明的一个实施例，用户还可指定没有目标的XPath表达式——即数据存储208中的XML数据209中不存在他们所指定的目标XML节点213。数据绑定205不是‘忘记’了其期望的目标节点213，而是进入一种‘等待’状态，在‘等待’状态中，它不被连接到任何特定XML数据209，而是等着看所期望的节点是否出现在213后台XML数据存储208中的XML数据209中。这对于文档汇编情形特别有用，其中文档204的每个部件（例如，标准封面、结束页、以及被重复使用的从句）可包含数据绑定205，而它们仅应在这些部件被汇编到单个最终文档204中时被填充。在此情形中，文档创建者指定每个文档‘部件’中的内容区域207内到XML数据209中的不存在于该部件中的XML节点213的数据绑定205（例如，封面可能包含带有到<Title/>XML元素和<Date/>XML元素的绑定205的内容区域）。当该部件在其目标文档外被查看时，那些绑定不被连接，因为不存在任何XML数据209，但只要该部件一被添加到包含期望数据209的文档中，该数据绑定205立即连接（同步）到数据209，并显示正确的内容——从而允许文档的创建者指定并保存绑定205，即使数据209尚未被生成。

一类悬挂引用215在内容区域207上的数据绑定205不能被成功链接到已链接的XML流中的节点213（即，绑定在内容区域中的状态）时发生。当节点213

从已链接的 XML 流中被替换/移除时，结果是一个或多个数据绑定 205 可能变为悬挂引用 215。较佳的是，如果数据绑定 205 由于其 XPath 而有悬挂引用 215，则文字处理器应用程序 202 继续将节点 213 的最后已知的 XPath 存储在数据绑定 205 上。当 XPath 不再解析到任何节点 209 时这可能发生。每当数据存储 208 通知文字处理器文档 204 某个 XML 数据 209 的更新时，文字处理器应用程序 202 就检查是否有任何悬挂引用 215 被最后的更新所解析（即，XPath 如今指向 XML 树中的有效节点 213）。如果文字处理器应用程序 202 解析一悬挂引用，则较佳的是数据存储 208 的内容优先于当前在数据绑定 205 中的内容——即，数据绑定 205 的内容被数据存储 208 中的节点 213 中的内容所取代。较佳的是使用通过一个或多个编程环境 206 可访问的简单 API 层来展示悬挂引用。

例如，假定文字处理器文档 204 包括以下段落，其中 Microsoft Corporation 对应于绑定到某个 XML 数据 209 中的 XPath /contract/header/company(3)的明文内容区域 207 数据：

"*Microsoft Corporation* is located at One Microsoft Way."

对应的 XML 数据 209 看起来可能是（到节点 213 的链接以单引号示出）：

```
<contract>
    <header>
        <company>Fabrikam Corporation</company>
        <company>Contoso Corporation</company>
        '<company>Microsoft Corporation</company>'
    </header>
</contract>
```

如果诸如开发者等用户使用数据存储 208 的 API 来移除<header>下的第一个<company>节点 213（单引号中的节点 213）：

```
<contract>
    <header>
        '<company>Fabrikam Corporation</company>'
        <company>Contoso Corporation</company>
        <company>Microsoft Corporation</company>
    </header>
</contract>
```

这导致文档 204 中的内容区域 207 上的数据绑定 205 维持到同一 XPath 的链接，因此数据绑定 205 变为到如今不存在的 \contract\header\company(3)的悬挂引用 215：

```
<contract>
    <header>
        <company>Ford Corporation</company>
        <company>Intel Corporation</company>
    </header>
</contract>
```

这意味着在内部我们有断裂的链接，但根据本发明，内容区域 207 的内容不会改变，也不会发生错误，即，

"*Microsoft Corporation* is located at One Microsoft Way."

当某个 XML 数据 209 被替换或移除时（或当一链接从一个文档被移到另一个文档时），则引用该 XML 数据 209 的所有数据绑定 205 立即变为指向已被删除的 XML 数据 209 的悬挂引用 215。如果数据绑定 205 包含悬挂引用 215，则文字处理器应用程序 202 继续存储与该数据绑定 205 相关联的最后已知的 XPath/名字空间链接。根据本发明的一个实施例，当一组数据绑定 205 变为悬挂引用 215 时，文字处理器应用程序 202 试图将这些链接重新附接到相关联的 XML 数据存储 208 中的任何其它可用的 XML 数据 209。如果其中任何数据绑定 205 的确解析到另一个 XML 数据 209 中的节点 213，则所有悬挂引用 215 都与此 XML 数据 209 相关联，从而更新了数据链接 205 如今所连接的相关联的内容区域 207。如果此 XML 数据 209 没有导致任何悬挂引用 215 的有效数据绑定，则文字处理器应用程序 202 对数据存储 208 中的每个 XML 数据 209 等执行类似的检查。如果没有一个 XML 数据 209 可被供悬挂引用 215 使用，则该绑定继续保持为到原始 XML 数据 209 的悬挂引用 215。

参考图 3 中所示的流程图，并继续参考图 2，描述了本发明的一个实施例。当用户使用文字处理器应用程序 202 打开文字处理器文档 204 时，图 3 中所示的过程 300 在 302 开始。在 304，文字处理器应用程序创建数据存储 208，然后在 310，用存储在文字处理器文档 204 中的、或通过使用用户界面或编程窗口 206 请求的任何 XML 数据 209 填充该数据存储 208。较佳的是，数据存储 208 作为文档 204 的部件被包括，但在文档编辑表面 203 上不可见。可以认识到，可在创建内容区域 207

和数据绑定 205 之前加载数据存储 208。类似地，可在数据存储 208 以前创建内容区域 207。换言之，图 3 中所示的各个操作不必按照任何特定顺序来执行，并可根据特定用户的偏好来实现。

在 306，用户创建一个或多个存在于文档 204 的表面 203 上的内容区域 207。注意，还可从文档 204 的现有内容读取这些内容区域。在 308，用户可通过提供特定的已连接 XML 数据 209 和指定目标节点 213 的 XPath 表达式来将数据绑定信息与内容区域 207 相关联。一个或多个数据绑定将数据存储 208 中的 XML 数据 209 的一个或多个节点 213 链接到一个或多个内容区域 207。数据绑定 205 变为绑定的或悬挂的。如上所述，每个节点 213 可被绑定到多个内容区域 207，后者中的每一个都指定到同一 XML 节点 213 的数据绑定 205。此外，数据绑定 205 可被链接到多个数据存储 208。在 309，用户可在内容区域 207 或数据存储 208 中创建 XML 数据。在 310，文字处理器应用程序 202 将所有 XML 数据加载到数据存储 208 中。在 312，文字处理器应用程序 202 从文档 204 或按照用户所要求 306 加载内容区域 207，并且在 314，文字处理器应用程序 202 从文档 204 或按照用户所要求 308 加载数据绑定 205。在 316，文字处理器应用程序 202 检查特定数据绑定 205 所指定的与节点 213 相关联的 XML 数据 209 是否存在。

如果 XML 数据 209 不存在，则在 318，文字处理器应用程序 202 确定同一 XML 名字空间内是否存在其它 XML 数据 209。在 316，如果 XML 数据 209 被定位在数据存储内，则在 320，文字处理器应用程序 202 确定对应于所指定的 XPath 的相关联的 XML 节点 213 是否存在。如果该 XPath 存在，则在 322，文字处理器应用程序 202 经由数据绑定将各个文档内容，即内容区域 207 以及任何其它所链接的内容，连接到一个或多个相关联的 XML 节点 213。如果在 320 没有定位到 XPath，则在 324，文字处理器应用程序 202 将该特定数据绑定 205 标记为悬挂引用 215（进入悬挂状态）。如果在 318，在数据存储 208 内的同一 XML 名字空间找到其它 XML 数据 209，则在 326，文字处理器应用程序 202 在此检查 XML 节点 213 是否存在于该数据内。

如果 XML 数据 209 不存在，则在 324 文字处理器应用程序 202 将该特定数据绑定 205 标记为悬挂引用（进入悬挂状态）。如果在 326 发现该 XML 数据 209 存在，则在 328，文字处理器应用程序 202 在该 XML 数据 209 内搜索期望的 XPath。如果在 320 找到节点 213，则在 322 文字处理器应用程序 202 将文档内容，即，内容区域 207 以及任何其它所链接的内容，连接到一个或多个相关联的 XML 节点

213。

用于绑定数据的对象模型

现在参考图4，描述本发明的另一个实施例。在许多软件应用程序中可使用对象模型来提供一种用于通过诸如应用程序 400 等应用程序的用户界面使用户可能执行的任务自动化的代码编写方法。对象模型允许编写执行以下任务的代码，例如：

- 为用户将重复性的任务自动化，
- 对用户的动作作出反应（以进行验证，等等），和/或
- 当文件410被打开/编辑/保存时执行自动的动作。

例如，应用程序（例如，WORD 2003）提供对象模型接口，它通过代码给出到文字处理器的用户界面的所有功能的访问。它是通过提供对文字处理器应用程序的所有范型/用户界面的等效部分的代码访问来实现此功能的。这些结构被组织为4个基本类别：

- 对象：对应于应用程序内的可作用的对象——例如，段落可被表示为对象模型中对应于文档表面中用户的段落的对象。
- 方法：可对对象执行的动作（例如，删除）。
- 属性：关于对象可设置的事物（例如，将段落加粗）。
- 事件：对应于响应于用户（或其它代码）与应用程序的交互而运行的子例程（例如，代码可注册以监听文件打开事件，只要应用程序中有文件被打开，无论是被用户还使其它代码打开，应用程序即可通知此代码（以允许用户执行任何动作）。

应用程序的对象模型内的任何对象都可展示允许代码操纵该对象的方法、属性、和/或事件。

描述包括根据本发明的一个实施例的编程环境402的应用程序400。编程环境402经由对象模型提供对诸如上述的文字处理器应用程序202等应用程序400中的对象的代码访问。可以认识到，本发明的实施例是在文字处理应用程序的上下文中描述的，但是如上所述，并不试图将本发明限制于任何特定的应用程序。编程环境402还可用于在利用稳健的方法/属性/事件集合来捕获对内容区域408的内容的改变的同时提供对内容区域408的行为的控制。此外，编程环境402使诸如程序员等用户能够程序地访问提供一个或多个内容区域408与数据存储414的一个或多个节点416之间的链接的一个或多个数据绑定406。

如上所述，可使用编程环境402来开发响应于任一方向的内容区域408或XML

数据 412 中的节点 416 的改变的代码 404。还可开发代码 404 以定义文件内容和数据存储 414 内的内容之间的关系。此外，还可开发反应于文件 410 的绑定区域内或数据存储 414 内的改变的代码，以捕获或截取诸如编辑、添加、删除等事件。本发明的实施例使用户能够用相关联的 XML 一次性编写代码 404，且该代码如今可被移植到支持 XML 构造的使用的所有文件类型，而无需担心目标应用程序的确切语义，从而大大简化并提高了应用程序开发的效率。

诸如应用程序编程接口（API）等编程环境 402 展示对象并提供使用代码 404 来操纵应用程序 400 内的一个或多个数据绑定 406 的能力。可将一个或多个数据绑定 406 与特定文件 410 的一个或多个内容区域 408 相关联。

较佳的是，内容区域属性返回提供用于操纵数据绑定的行为的两个事件的对象：

- **.BeforeStoreUpdate(ContentRegion As ContentRegion, Content As String)**

  每当内容区域的内容被修改时、并且在文本被发送到 XML 数据 412 中的适当节点 416 以前返回对象（串）的事件。这允许代码 404 在串到达 XML 数据的节点以前操纵该串，而无需改变内容区域 408 的内容。

- **.BeforeContentUpdate(ContentRegion As ContentRegion, Content As String)**

  每当有对 XML 数据 412 中被链接到相关联的内容区域 408 的节点 416 的内容的改变、并且在内容区域 408 被更新以前引发一个事件。这允许代码 404 在数据 412 被显示在内容区域 408 中以前操纵数据 412，而无需改变节点的数据。较佳的是，内容区域 408 包括属性，这些属性包括通过编程环境 402 可访问的“DataBinding”（数据绑定）属性。DataBinding 属性提供对内容区域 408 和与文件 410 相关联的数据存储 414 的数据 412 之间的关系的访问。较佳的是，DataBinding 返回 DataBinding 对象（属性、方法和事件），以允许对数据绑定 406 以及与数据绑定 406 相关联的其它数据的操作。代表性的方法和属性包括，但不限于：

- **.CustomDataStream As CustomDataStream**

返回表示与数据绑定 406 相关联的 XML 数据 412（用于进一步的操纵）的对象的属性。可开发代码 404 以使用对象模型来直接操纵 XML 数据。如果是悬挂引用则不返回任何对象。

- **.CustomDataXMLNode As CustomDataXMLNode**

返回表示 XML 数据 412 中是数据绑定 406 的目标的 XML 节点 416 的对象的属性。可开发代码 404 以操纵特定数据绑定 406 的目标节点 416。如果是悬挂引用则不返回任何对象。例如，假定用户想要移动节点 416 以改变数据绑定 406 的目标。又如，假定用户想要删除节点 416，因为文件 410 将被放到某个公共服务器上。由此，代码 404 可被用来使用此被返回的对象来寻找并删除节点 416，以防在文件 410 被放到公共服务器上时因疏忽而发送信息。

- **.Delete As Void**

返回用于将数据绑定 406 从内容区域 408 移除的对象的方法。例如，假定文件 410 包括具有被链接（或数据绑定）到 XML 数据 412 中的 name 元素的名字的内容区域 408。代码 404 可被用来删除绑定 406（即，“dataBinding.Delete”），因为用户想要释放数据 412，从而操作以将该链接从内容区域 408 移除。

- **.IsBound As Boolean**

返回用于确定数据绑定 406 当前是否为悬挂引用的对象的属性。代码 404 可被用来确定目标节点 416 是否存在。在各种情况中，为成功进行代码实现，较佳的是确定目标节点是否存在，即，数据绑定 406 是悬挂的还是有效的。如果目标节点存在，则该对象返回“.true”，如果是悬挂引用或不存在绑定，则返回“.false”。

- **.SetBindingByNode(CustomDataXMLNode As CustomDataXMLNode) As Boolean**

返回用于通过为特定链接提供目标 XML 节点 416 来创建/改变当前内容区域 408 上的数据绑定 406 的对象（CustomDataXMLNode）的方法。如果绑定存在，则现有绑定被替换，且新的绑定节点（若有）的存储内容取代现有内容。如果绑定是活节点则返回“.true”，如果不是绑定到活节点则返回“.false”。

- **.SetBindingByXPath(CustomDataStream As Variant, XPath as String, [PrefixMapping As String]) As Boolean**

返回表示用于通过为链接提供 XPath 和 XML 数据 412 来创建内容区域 408 上的数据绑定 406 的信息（诸如 XPath、到节点的指针（ID），和/或与名字空间有关的前缀映射关系）的对象（CustomDataStream、XPath、和/或 PrefixMapping）的方法。

- CustomDataStream：指定应被绑定到的存储项的变量，可以是以下任何一项：
  - 表示期望的流的 CustomDataStream 对象

- 表示期望的流的根名字空间的串——如果有一个以上具有该根名字空间的流，则攫取存储中所找到的第一个项
- 表示期望的存储项在存储的集合中的索引的整数

• XPath：指定要绑定到的 XPath 的串，试图确保其为有效 XPath 句法，如果不是则失败并给出错误消息。

• PrefixMapping：表示在查询所提供的 XPath 表达式时可使用的前缀映射关系的可任选的串，如果没有给出，则使用存储项本身所使用的前缀映射关系集合。

• **.XPath As String**

返回用于定位数据绑定 406 所使用的 XPath 表达式以指向 XML 数据 412 内的目标 XML 节点 416 的对象的属性。

可以认识到，使用以上所列出的属性、方法、以及事件的代码开发可根据特定应用程序来定制。

应当认识到，本发明的各个实施例的逻辑操作被实现为(1)计算机实现的动作序列或计算系统上运行的程序模块，和/或(2)计算系统内相互连接的及其逻辑电路或电路模块。实现是根据实现本发明的计算系统的性能要求进行选择的问题。由此，组成本文中所描述的本发明的实施例的逻辑操作被称为操作、结构设备、动作或模块等不同的术语。本领域技术人员将会认识到，这些操作、结构设备、动作以及模块可在软件、固件、专用数字电路、及其任何组合中实现，而不会偏离本发明的精神和范围，如所附权利要求书中所阐述。

以上说明书、示例及数据提供了制造和使用本发明的组成的完整描述。因为可作出本发明的许多实施例而不会偏离本发明的精神和范围，所以本发明驻留在所附权利要求书中。

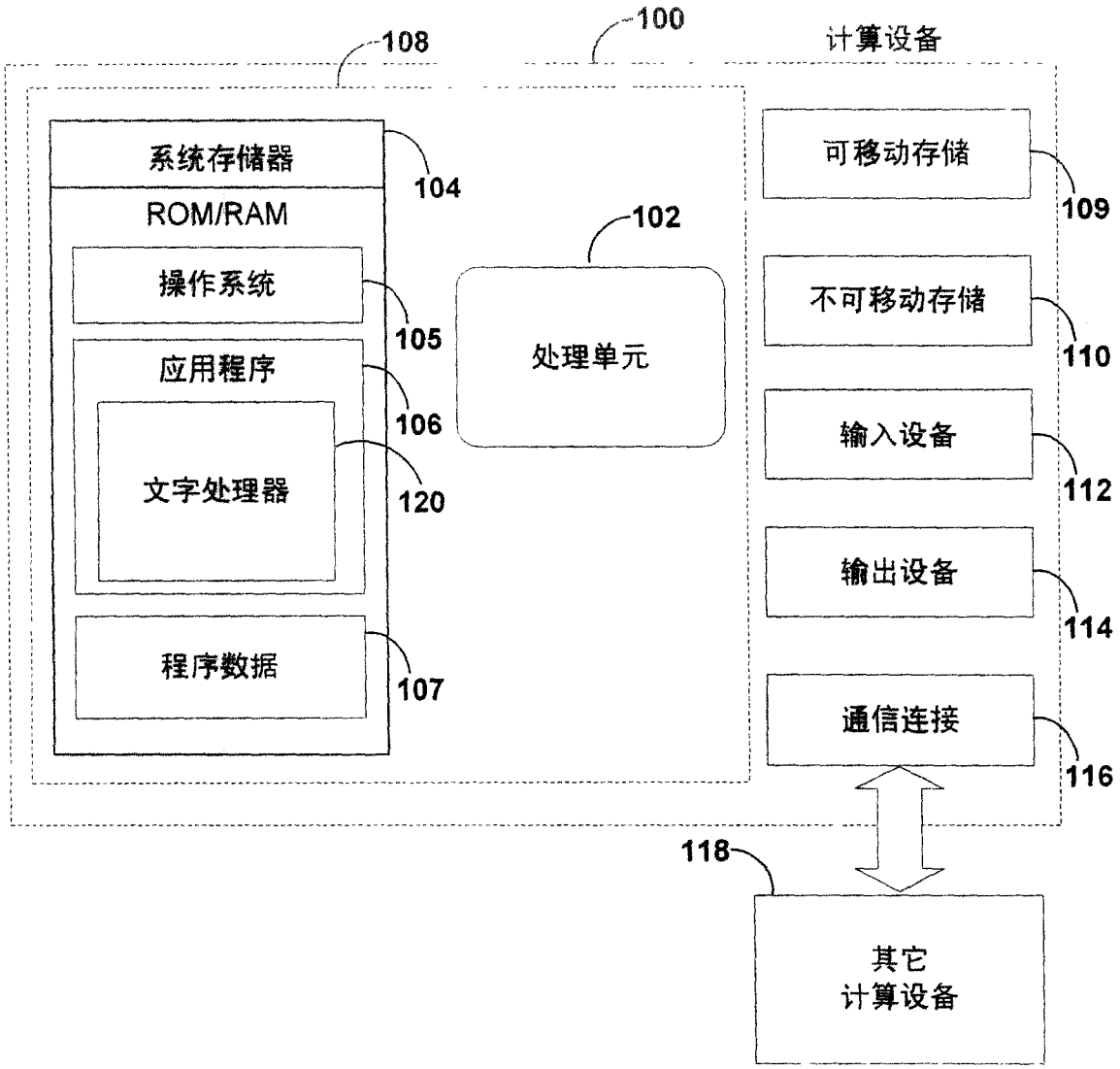
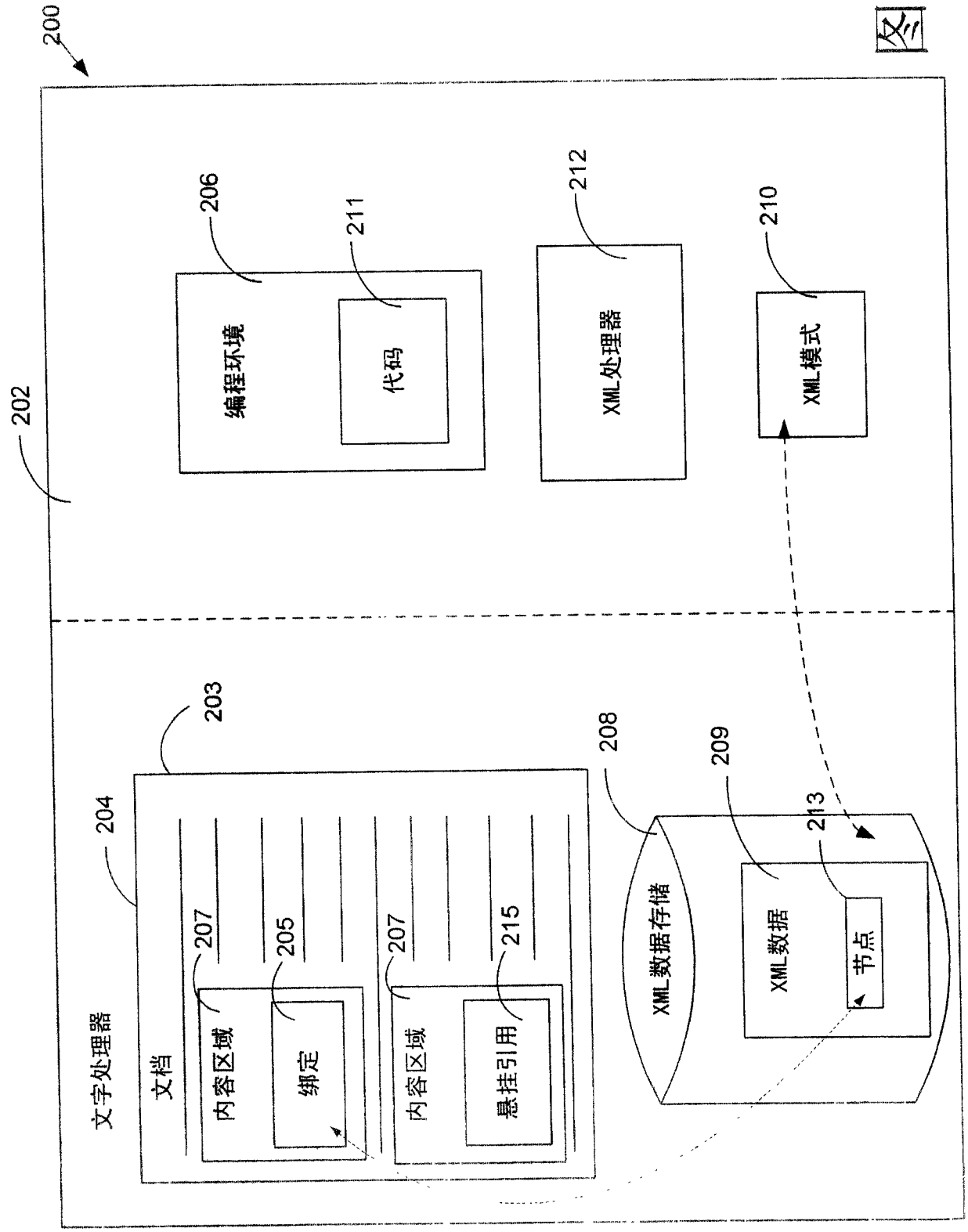

100
计算设备
108
系统存储器
104
ROM/RAM
操作系统
105
应用程序
106
文字处理器
120
程序数据
107
102
处理单元
可移动存储
109
不可移动存储
110
输入设备
112
输出设备
114
通信连接
116
118
其它
计算设备

图 1

图 2


200
202
编程环境
206
代码
211
XML处理器
212
XML模式
210
文字处理器
204
文档
203
内容区域
207
绑定
205
内容区域
207
悬挂引用
215
208
XML数据存储
209
XML数据
213
节点

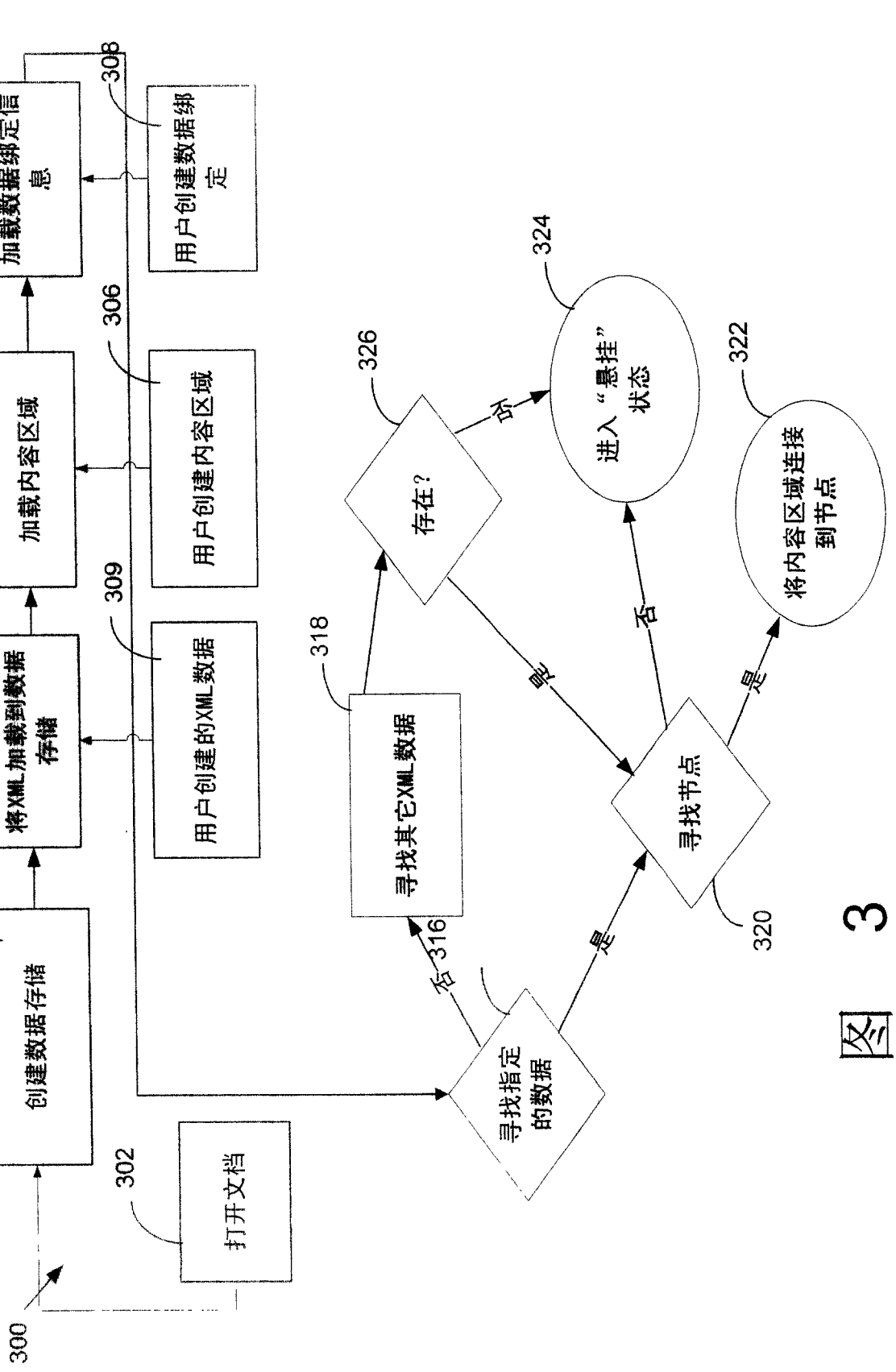

300
打开文档
302
创建数据存储
304
将XML加载到数据存储
310
加载内容区域
312
加载数据绑定信息
314
用户创建的XML数据
309
用户创建内容区域
306
用户创建数据绑定
308
寻找指定的数据
寻找其它XML数据
318
316
否
是
存在?
326
否
是
进入“悬挂”状态
324
寻找节点
320
否
是
将内容区域连接到节点
322

图 3

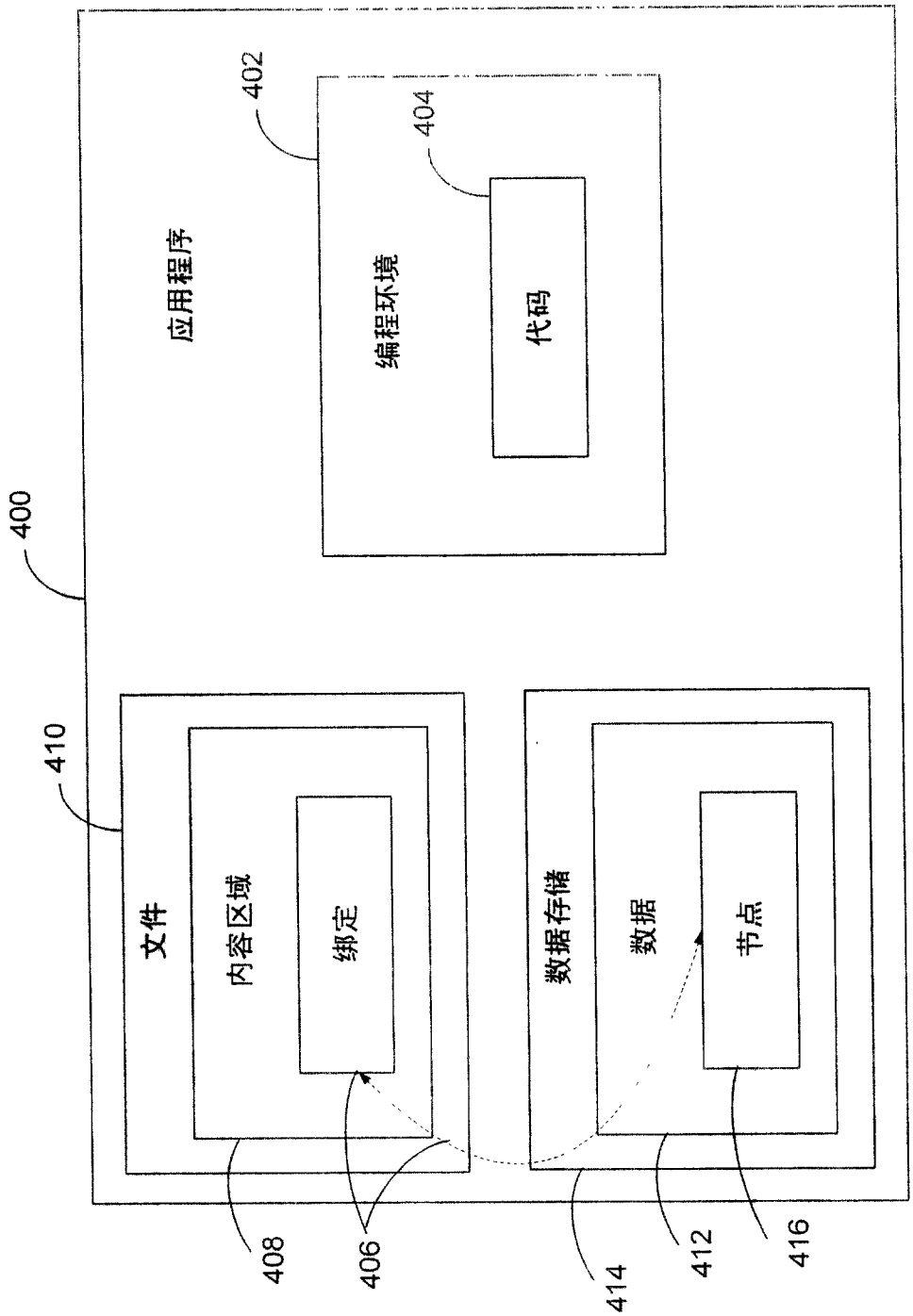

400
应用程序
402
编程环境
404
代码
410
文件
内容区域
绑定
数据存储
数据
节点
408
406
414
412
416

图 4