(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 1811765 B
(45)授权公告日 2010.10.13

(21)申请号 200510128895.8

(22)申请日 2005.12.06

(30)优先权数据
11/030,423 2005.01.06 US

(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 M·萨维斯基
B·M·琼斯 R·A·利特尔
M·森德兰德

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 李玲

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

(56)对比文件
US 2003007009 A1,说明书第64段,第68段第1-4行,第69段第1-3行,第70段的第1-5行,第74段的第1-6行.
US 2004217985 A9,2004.11.04,说明书第88段的第12-15行,第19-21行.
US 2004243938 A1,2004.12.02,说明书第16段,第57段,第130段,第151段,第156段,第174段,第215段、附图18,附图25.

审查员 李劲娴

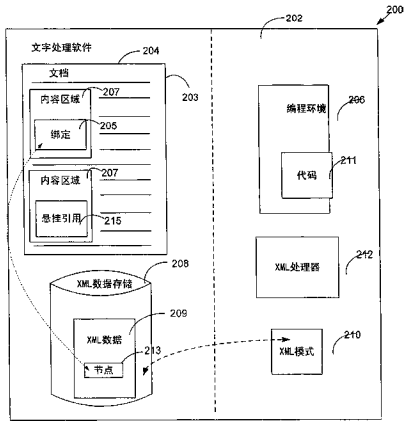
权利要求书 2 页 说明书 13 页 附图 3 页

(54)发明名称

文字处理应用程序中的数据绑定

(57)摘要

提供了用于创建文字处理软件文档的文字处理软件应用程序,其中数据和呈现可以被分开。数据可以被存储在与文字处理软件文档的呈现表面分开的位置中。文字处理软件应用程序的用户可以建立数据内容和呈现表面之间的链接(或绑定)。用户可以通过直接改变所链接的XML数据来修改文字处理软件文档的内容,而不必处理可能不断改变的呈现格式的复杂性。用户可以通过编辑数据存储来修改文字处理软件文档的内容,而无需对呈现表面的当前布局的了解。用户可以通过简单的文档编辑来修改复杂的后台数据,而不受数据结构的复杂性的影响。



CN 1811765 B

1. 在文字处理软件应用程序中，一种用于提供文字处理软件文档的方法，包括：

创建用于显示所述文档的内容的呈现表面，

创建用于存储含有相关联的结构的文档数据的数据存储，

将模式文件与存储在所述数据存储内的所述文档数据相关联，

在所述文字处理软件文档中创建一个或多个内容区域，其中所述一个或多个内容区域具有不同的呈现格式，以及

创建与所述一个或多个内容区域相关联的一个或多个绑定，所述一个或多个绑定用于使所述一个或多个内容区域与所述文档数据内的一个或多个节点相链接，并且当所述一个或多个内容区域被链接到所述文档数据内的同一节点时，所述一个或多个内容区域的内容完全相同。

2. 如权利要求 1 所述的方法，其特征在于，所述方法还包括将所述相关联的结构与所述呈现表面分开。

3. 如权利要求 1 所述的方法，其特征在于，所述方法还包括提供对所述数据存储中的文档数据的访问。

4. 如权利要求 3 所述的方法，其特征在于，所述方法还包括直接编辑在所述数据存储内的所述文档数据。

5. 如权利要求 4 所述的方法，其特征在于，所述方法还包括更新与编辑后的文档数据相关联的一个或多个内容区域。

6. 如权利要求 1 所述的方法，其特征在于，所述方法还包括基于对一个或多个内容区域的编辑来更新所述数据存储中的相关联的文档数据。

7. 如权利要求 1 所述的方法，其特征在于，所述方法还包括当数据绑定引用所述数据存储中不存在的位置时，创建悬挂引用。

8. 如权利要求 1 所述的方法，其特征在于，所述方法还包括向所述文字处理应用程序展示编程接口，并使用所述编程接口来开发用于引起所述数据存储中或所述呈现表面的一个或多个内容区域中的改变 / 对该改变作出反应的代码。

9. 如权利要求 1 所述的方法，其特征在于，所述方法还包括基于与数据存储中的至少一个节点相关联的修改来更新多个内容区域中的内容。

10. 一种用于提供文字处理软件文档的设备，包括：

用于创建存储的装置，所述存储用于存储数据，

用于将 XML 数据添加至所述存储的装置，

用于将模式文件与所述存储内的 XML 数据相关联的装置，其中，所述模式定义了所述数据的结构，以及

用于创建与所述文档的一个或多个内容区域相关联的一个或多个数据绑定的装置，所述一个或多个数据绑定用于使所述一个或多个内容区域与所述 XML 数据内的一个或多个节点相链接，其中所述一个或多个内容区域具有不同的呈现格式，并且当所述一个或多个内容区域被连接到所述 XML 数据内的同一节点时，所述一个或多个内容区域的内容完全相同。

11. 如权利要求 10 所述的设备，其特征在于，还包括用于开发监视一个或多个链接的代码的编程装置。

12. 如权利要求 10 所述的设备，其特征在于，所述设备可操作来编辑所述文档中的多个所链接的区域，从而引起所述存储中的相关联的改变。

13. 如权利要求 10 所述的设备，其特征在于，所述设备可操作来编辑所述存储中的节点，从而引起所述文档的一个或多个内容区域中的相关联的编辑。

14. 如权利要求 10 所述的设备，其特征在于，所述用于创建链接的装置还可用于创建悬挂或绑定链接。

文字处理应用程序中的数据绑定

背景技术

[0001] 近年来标记语言受到广泛的欢迎。一种标记语言，可扩展标记语言 (XML)，是提供标识、交换和处理各种数据的方法的通用语言。例如，XML 用于创建可以由各种应用程序利用的文档。XML 文件的元素一般拥有相关联的名字空间和模式。

[0002] 名字空间是 XML 文档中用于定义元素 / 属性名称和类型的名字集合的唯一标识符。名字空间的名字通常用于唯一地标识每一类 XML 文档。唯一的名字空间区分来自不同源且碰巧拥有相同名称的标记元素。

[0003] XML 模式提供描述和确认 XML 环境中的数据的方法。模式规定了什么元素和属性被用来描述 XML 文档中的内容、哪里允许每一元素、该模式中允许何种类型的内容以及在哪些其它元素内可以出现哪些元素。模式的使用确保文档是以一致且可预测的方式来结构化的。模式可以由用户创建，且一般由诸如 XML 等相关联的标记语言支持。通过使用 XML 编辑器，用户可以操纵 XML 文件，并生成遵循用户所创建的模式的 XML 文档。在以往的文字处理软件应用程序中，向该应用程序添加对自定义 XML 模式的支持，使得用户能够使用自定义 XML 标记（例如，<title>（标题））来对文档的内容“加标签”，这本质上向之前为未分类的文本运行给出语义意义。这意味着之前仅是具有格式的文本但对其它应用程序处理没有任何意义的文档现在可以是含有 XML 标记的特定片段的结构化 XML 文档，该 XML 标记来自任何其它知道 XML 的应用程序可以定位和理解的任何用户定义的 XML 模式。

[0004] 在一个基本示例中，文档顶端的文本可以使用来自用户定义的 XML 模式的 <title>XML 元素被“加标签”为标题，这意味着其它知道 XML 的应用程序现在可以容易地理解，该文本范围包含“标题”并适当地提取它。这使得后端进程能够智能地使用适当的语义和上下文来提取文档的各部分（例如，该文本是 <title>）。

[0005] 然而，与现有的文字处理应用程序相关联的缺点源于自定义 XML 标记的添加和持久性被束缚于文档的呈现这一事实。即，在现有的实现中，在文字处理软件文档的 XML 标记（例如，以 XML 格式表示的顾客发票的细节）与其在文档表面上的呈现（例如，三段纯文本及其后具有特定表格样式的 5 列 4 行的表格）之间存在不可动摇的链接。从而，在现有的文字处理软件应用程序中表示的 XML 数据（因为它被束缚于呈现）必须与文档的内容精确一致。例如，如果发票的 XML 模式规定 <date>（日期）在 <address>（地址）之前，而 <address> 在 <phoneNumber>（电话号码）之前，那么这三个 XML 元素必须以文档中呈现的完全一样的顺序出现。这意味着对呈现格式的改变（例如，来回移动含有 <date> 的表格行）也会引起对该文档中包含的 XML 数据的结构的改变，这需要解决方案开发器部分上的额外步骤来确保该数据遵循相关联的 XML 模式的结构。从而，不向文档的最终用户提供自由操纵呈现的自由，因为这样做可能一定会改变数据的语义，从而潜在地违反该数据的 XML 模式。

[0006] 另外，在现有的文字处理软件应用程序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含义。这样，如果粗体文本的段落被标记为标题，那么由现有的文字处理软件应用程序保存的结果 XML 将会如下：

[0007] <w:p>

[0008] <Title>

[0009] <w:r>

[0010] <w:rPr>

[0011] <w:b/>

[0012] </w:rPr>

[0013] <w:t>This is the title.</w:t>

[0014] </w:r>

[0015] </Title>

[0016] <w:p>

[0017] 如上所示，对现有的文字处理软件应用程序非常特有的 XML 标签（在该示例中，w:p、w:r 等）在两边围绕自定义 XML 标签。这意味着正在处理该数据的知道 XML 的解决方案必须不仅理解其自身的数据格式（包含 <Title> 元素），也必须理解现有文字处理软件应用程序格式的精确细节，所以当其搜索其自己的数据时知道遍历和忽略该信息。从而，这种类型的实现仍旧向用户强加某些要求，因为对文档中文本外观的小的改变（例如，将 <Title> 元素的内容拖入表格单元中）会导致围绕文字处理软件原有标签内的自定义 XML 标签的位置的显著改变。从而，程序员 / 代码开发员经常需要编写额外的代码，以预期和理解现有的文字处理软件应用程序将基于呈现会在何处放置自定义 XML 元素，并处理所有各种排列。这意味着所得的解决方案可能仍旧需要包含用于处理特定的现有文字处理软件应用程序需要的重要的逻辑代码。

[0018] 使用现有的文字处理软件应用程序工作的程序员 / 代码开发员在考虑读和写操作时也需要考虑文档布局格式的含义。例如，用户可能试图抓取 <StockSymbol> 元素的值，并使用它来将公司的全称放置在同一文档中的 <CompanyName> 元素中，作为对编写公司报告的用户的简单的促进。为了维护文档的完整性，用户需要在他们可以编写功能性代码来执行读写动作之前，对来自文档的所需数据的读和写来考虑文档的当前布局格式。例如，用户可能需要知道他们写入的值是否位于表格单元中、项目符号列表中等，以构造当被插入文档中时会产生所需结果的现有文字处理软件应用程序的格式化信息。这是额外编码来理解文字处理软件应用程序的呈现语义的另一个潜在原因。

[0019] 现有文字处理软件应用程序的又一限制是 XML 元素的编辑行为有时可能被察觉到为“脆弱的”。如上所述，这部分是因为它们受这一事实的限制，即基于用户定义的模式，文档表面标签的定位确定 XML 数据的结构。从而，引发大量问题。首先，一般用户操作（例如，从一部分复制 / 粘贴至另一部分）可能会更改 XML 结构并使得该文档依据相关联的 XML 模式是无效的。第二，在这样的文字处理软件实现中，顾客定义的 XML 模式所需的所有元素需要被包括在文档表面上的某种形式中。这意味着开发员可能难以创建相关联 XML 数据，作为用于携带没有在文档表面显示而更多地作为元数据的关于文件的额外信息的方法。并且，第三，在这样的文字处理软件实现中，也需要包括文档表面上语义上不必要的元素（例如，不标记混合内容的非叶子元素），这进一步提高了普通用户操作修改 XML 数据的能力。

[0020] 在多种情况下，定义 XML 数据（例如，构成备忘录文档的数据）的模式往往由单个标准体来严格定义，以便于多个不同种类处理系统之间的该数据的通信。然而，通过这样便

于后端通信，通常牺牲了文档的人可读性和可编辑性，这使得用户难以理解和剖析该数据。例如，XML 标准可以定义日期的标准格式，诸如 dd-mm-yyyyThh:mm:ss.ssss（日 - 月 - 年 T 时 : 分 : 秒 . 毫秒）。所有的数据必须以这种格式来表示，以便由知道 XML 的应用程序剖析。显然，用户难以正确地输入该格式，且该格式通常与用户一般输入日期的方式冲突（例如，在许多场所一般使用 mm-dd-yyyy（月 - 日 - 年）来代替 dd-mm-yyyy 等）。

[0021] 从而，需要一种使得开发员能够分离 XML 数据和这类数据在诸如文字处理软件应用程序等应用程序中的呈现的方法。

发明内容

[0022] 本发明的实施例提供一种用于创建文字处理软件文档的文字处理软件应用程序，其中数据和呈现可被分开。更具体地，数据可被输入至存储在与其呈现格式分开的位置中的文字处理软件文档中，并可从中提取。根据本发明的实施例，文字处理软件应用程序的用户可为文字处理软件文档数据创建分离的存储位置，并建立该数据的内容与呈现表面之间的链接（或绑定）。文字处理软件应用程序的用户可以通过直接改变所链接的 XML 数据来修改文字处理软件文档的内容，而不必处理可能不断改变的呈现格式的复杂性。将视图 / 呈现与数据分离允许用户继续自由地编辑文档，同时允许数据以单个已知的结构格式在持久文件格式中表示。当使用外部进程来修改文字处理软件文档的内容时，用户可以编辑存储在该文字处理软件文件内的分离的 XML 数据片段，而无需对呈现表面或格式的当前布局的了解。这些链接（或绑定）也被创建来使得对呈现表面的结构的改变不会确定 XML 数据的结构，反之，对 XML 数据结构的改变不会确定文字处理软件文档的呈现表面的布局——可以创建或断开这些链接而没有对文档内容的不利影响。本发明的实施例倾向于简化文字处理软件文档中对结构化数据的添加、编辑或提取。

[0023] 根据本发明的实施例，可以使用文本的特定区域的语义来标记文字处理软件文档，而相关联的数据现在存储在该文档内的分开的 XML 数据文件中。使用数据绑定将数据链接至文档中的文本范围，即标记的区域。从而，数据现在可以被存储在其自己的 XML 数据文件中的一致位置（“数据存储”）中，而不论数据绑定将该内容放至文档呈现表面中的何处或者如何呈现该数据。这样，用户不必担心在文档中对数据进行移动 / 改变 / 删除等，因为该数据的 XML 结构位于一致位置中的自定义 XML 内，且不受对呈现表面改变的影响。而且，本发明的实施例为文字处理软件应用程序提供编程接口。用户可以使用该编程接口来开发用于对数据存储中或文档中一个或多个区域中的改变作出反应的代码。

附图说明

[0024] 图 1 示出了可用于本发明的一个示例性实施例中的示例性计算设备。

[0025] 图 2 是示出用于实现本发明的示例性环境的框图 ;以及

[0026] 图 3 是根据本发明的实施例的流程图。

具体实施方式

[0027] 贯穿说明书和权利要求书，以下术语采用此处明确相关联的含义，除非上下文清楚地另外指明。

[0028] 术语“数据”指的是文字处理文档带有的、引用的或使用的任何补充信息。该信息通常是巨大的，且不太可能全部在文档的呈现层上展示。

[0029] 术语“标记语言”或“ML”指的是文档内指定应用程序将如何解释文档的各部分的专用代码的语言。在文字处理软件文件中，标记语言指定如何格式化或布置文本。

[0030] 术语“元素”指的是 XML 文档的基本单元。元素可以包括属性、其它元素、文本和 XML 文档的其它内容区域。

[0031] 术语“呈现”指的是文档的可视部分——如果打印文档则会出现的文本和布局。

[0032] 术语“标签”指的是描绘 XML 文档内的元素的插入到文档中的字符。每一元素可以含有仅仅两个标签：开始标签和结束标签。可能有空元素（没有内容），在这种情况下，允许一个标签。

[0033] 标签之间的 XML 内容被认为是元素的“子元素”（或后代）。因此，嵌入在元素内容中的其它元素被称为“子元素”或“子节点”或元素。直接嵌入在元素内容中的文本被认为是该元素的“子文本节点”。子元素和元素内的文本一起构成了元素的“内容”。

[0034] 术语“属性”指的是被置为特定值并与元素相关联的附加性质。元素可以含有与之相关联的任意数量的属性设置，包括没有属性设置。属性被用来将附加信息与不含有附加元素的元素相关联，或被作为文本节点来对待。

[0035] 术语“内容区域”指的是作为用户输入的一种类型的内容的容器的文档的有界和 / 或可任选标注的区域。见于 2004 年 9 月 30 日提交的，转让给本发明的受让人的名为“Methods，System，and Computer-Readable Medium For ManagingSpecific Types of Content In An Electronic Document（用于管理电子文档中特定类型内容的方法、系统和计算机可读介质）”的美国申请第 10/955,612 号，该申请通过整体引用包含在此。

[0036] “XPath”是使用模式表达式来标识 XML 文档中的节点的操作符。XPath 模式以斜杠分割的子元素名称列表，它描述通过 XML 文档的路径。模式“选取”匹配路径的元素。

[0037] 术语“XML 数据存储”指的是文字处理文档内的容器，当文件打开时它提供对存储在文字处理软件文档中的数据（例如，以 XML 格式）的存储和修改的访问。

[0038] 术语“数据绑定”指的是内容区域上的属性，它确定文字处理软件文档内其中可以存储内容区域的内容的一个或多个 XML 数据片断中的 XPath 位置。如此处所用的：

[0039] “ref”- 指的是由个别绑定使用的每一绑定的 XML 文档的唯一整数；

[0040] “ID”- 指的是 XML 数据存储中特定 XML 文档的唯一 ID

[0041] “selectionNamespaces”- 指的是 XML 数据存储中相关联 XML 文档的前缀映射（将名字空间与简短缩写相关联）；以及

[0042] “rootURI”- 指的是 XML 数据存储中相关联 XML 文档的根名字空间

[0043] 说明性操作环境

[0044] 本发明的实施例提供了用于创建其中可以分离 XML 数据的存储和呈现的文字处理软件文档的文字处理软件应用程序。更具体地，可以被输入到文字处理软件应用程序中并从中提取的数据被存储在与文字处理软件文档的呈现格式分离的位置中。从而，文字处理软件应用程序的用户可以为包含在文字处理软件文档中的 XML 数据创建分离的存储位置，并建立该数据内容和呈现表面之间的链接（或绑定），使得用户能够通过编辑该呈现的内容来编辑相关联的 XML 数据，而且防止用户改变相关联的 XML 数据的结构。例如，发票的

数据可以按文字处理软件文件格式来分离地存储为 XML，使得在文档中来回移动链接的位置不会改变分离的数据存储的结构。这样，使得对该结构化数据的后端处理更容易，因为数据现在具有不受用户编辑文档的方式影响的已知结构。用户可以在文字处理软件文档中编辑数据、格式化数据，包括丰富呈现格式化等，且仅有对文本内容的改变被‘推’回到存储在该文档后方的 XML 数据。然而，根据本发明，通过与文字处理软件文档的用户交互来更新的所有数据可用于 XML 的原始本地流。从而本发明也使得能够通过直接改变链接的 XML 数据来修改文字处理软件文档的内容，而无需处理可能不断改变的该数据的呈现格式的复杂性。这样做，极大地简化了对文字处理软件文档中的结构化数据的添加、编辑和提取。

[0045] 参考图 1，用于实现本发明的一个示例性系统包括计算设备，诸如计算设备 100。在最基本的配置中，计算设备 100 一般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）或是两者的某种组合。系统存储器 104 一般包括操作系统 105、一个或多个应用程序 106，且可以包括程序数据 107。在一个实施例中，应用程序 106 可以包括文字处理软件应用程序 120。该基本配置在图 1 中由虚线 108 内的组件示出。

[0046] 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，例如磁盘、光盘、或磁带等附加数据存储设备（可移动和 / 或不可移动）。这样的附加存储在图 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、磁盘存储或其它磁性存储设备，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。任何这样的计算机存储介质都可以是设备 100 的一部分。计算设备 100 也可以具有诸如键盘、鼠标、笔、语音输入设备、触摸输入设备等输入设备。也可以包括诸如显示器、扬声器、打印机等的输出设备。在本领域中，这些设备是公知的，无需在此处详细讨论。

[0047] 计算设备 100 也可以包含允许设备诸如通过网络等与其它计算设备 118 通信的通信连接 116。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线连接的有线介质，以及诸如声学、RF、红外及其它无线介质的无线介质。如此处所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。

[0048] 可以在计算设备 100 的系统存储器 104 中存储多个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。系统存储器 104 也可以存储一个或多个程序模块，诸如文字处理软件应用程序 120 以及以下所述的其它程序模块。文字处理软件应用程序 120 可用于提供用于创建、编辑和处理电子文档的功能。

[0049] 根据本发明的一个实施例，文字处理软件应用程序 120 包括来自微软公司的 WORD 程序。然而，应该理解，可以使用来自其它制造商的文字处理软件应用程序来实施本发明的

各个方面。还应理解，本发明的各个方面不限于文字处理软件应用程序，而也可以利用能够处理各种形式的内容（例如，文本、图像、图片等）的其它应用程序 106，诸如电子表格应用程序、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。

[0050] 本发明的实施例可以被实现为计算机进程、计算系统或者诸如计算机程序产品或计算机可读介质等制品。计算机程序产品可以是计算机系统可读的且编码用于执行计算机进程的指令的计算机程序的计算机存储介质。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且编码用于执行计算机进程的指令的计算机程序的信号。

[0051] 在文字处理软件应用程序中绑定数据

[0052] 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软件环境 200，包括文字处理软件应用程序 202、文字处理软件文档 204、编程环境 206、数据存储 208、模式文件 210 以及 XML 处理模块 212。然而，如上所述，本发明也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，诸如电子表格应用程序、数据库应用程序、演示应用程序、绘图或计算机辅助的应用程序等等。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。可以理解，本发明不旨在由此处所述的任何特定实施例或示例来限制。例如，文字处理软件环境可包括多个文字处理软件文档 204、数据存储 208 和 / 或模式文件 210。根据本发明的实施例，文字处理软件应用程序 202 使用 XML 处理模块 212 来处理根据可扩展标记语言格式化的数据。一个合适的 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。

[0053] 文字处理软件应用程序 202 包括其自己的一个或多个名字空间和一个或一组模式 210，它们被定义以用于与文字处理软件应用程序 202 相关联的文档 204。由模式 210 为文字处理软件应用程序 202 定义的该组标签和属性定义了文档 204 的格式。如下所述，并根据本发明的实施例，数据存储 208 可以包括数据 209。较佳地，模式 210 被附加至数据存储 208 内的数据 209。文字处理软件文档 204 也包括如下所述的由用户创建的内容区域 207。可以作为文字处理软件应用程序 202 的一部分包括一个以上数据存储 208、相关联的 XML 数据 209 和模式 210。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一组语法和数据类型规则，可以将一个或多个 XML 模式 210 与 XML 数据 209 相关联，用于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 的顺序的规则以及与应用于 XML 数据 209 的各个元素相关联的特定规则。

[0054] 本发明的实施例提供可用于创建文字处理软件文档 204 的文字处理软件应用程序 202，文档中的数据和呈现可通过与文字处理软件文档一起存储的分离的数据存储 208 的存在而被分离。更具体地，可输入到文字处理软件文档 204 中或从中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软件文档 204 的呈现格式分离。从而，文字处理软件应用程序 202 的用户可以为文字处理软件文档 204 的数据创建分离的存储位置，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，使得用户能够通过编辑呈现的内容来编辑数据，但是通过同一标记防止用户改变数据 209 的结构。在文档 204 中来回移动内容区域 207 的位置不会改变分离的数据存储 208 中的 XML 数据 209 的结构。而且，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，由于 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，从而简化了结构化数据的后端处理。

[0055] 本发明的实施例使得能够通过直接改变所链接的数据来修改文字处理软件文档 204 的内容，而无需处理可能不断改变的呈现格式的复杂性。这样做极大地简化了对文字处理软件文档 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 中来回移动，而不会影响数据的结构。较佳地，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界面或编程窗口 206 来定义。

[0056] 用户使用 XPath 表达式（用于标识 XML 树中的节点 213 的标准 XML 方法）来唯一地标识文档内容区域应被绑定到的期望 XML 节点 213。文字处理软件应用程序 202 用于自动剖析和使用 XPath 以定位数据绑定区域的期望目标。这也意味着熟悉 XPath 标准的开发者可充分利用 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，假设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。如果此信息位于与文档 204 相关联的 XML 数据 209 中，则用户可创建链接至具有最高的已完成事项的个人的名字的 XPath 表达式，并且随数据改变该链接自动转移到适当的位置（节点 213）。链接也可以通过使用代码 211、用户界面、或编程环境 206 在节点 213 之间改变。

[0057] 或者，用户可创建唯一标识文档内容区域 207 应被绑定至的数据 209 中对象表示节点 213 的数据绑定。文字处理软件应用程序 202 用于自动确定 XPath，以为数据绑定区域定位期望目标。然而，这意味着在该情况下，数据绑定 205 将自动更新其 XPath 以确保与同一 XPath 表达式相比，数据绑定 205 指向同一对象。

[0058] 如上文所简要所述的编程代码 211 可以被开发成使用 XML 处理模块 212，并对在任一方向上移动（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，或相反）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，代码 211 可以被开发成对文档 204 的绑定区域内或数据存储 208 内的改变作出反应，以俘获或截取诸如编辑、添加、删除等的事件。例如，假设用户想要确保至多一个文档可以使用一个特定标题。基于输入到标题节点中的内容，代码 211 可以对照中央数据库检查该标题是否已被使用。如果该标题已被采用，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。本发明的实施例使得用户能够使用相关联的 XML 一次性编写代码 211，且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，从而极大地简化并流线化应用程序开发。

[0059] 根据本发明的实施例，可用表示文本的特定区域（例如，标题、正文等）的语义的内容区域 207 来对文字处理软件文档 204 添加标签，并且通过添加数据绑定 205，该内容区域 207 内的相关联文本现在被存储在文档 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，被标签的区域。从而，如今数据 209 可被存储在其自己的 XML 流中的一致位置（“数据存储”），而无论数据绑定 205 在文档 204 的呈现 203（即，编辑特定文档时用户与之交互的数字表示，例如描绘特定用户文档的 WORD 窗口）中的何处定位相关联的内容，或者如何呈现数据 209。这样，用户不必担心在文档 204 中来回移动数据，因为 XML 数据 209 如今被定

位在一个一致的位置中的数据存储 208 内。而且，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。诸如元数据等的这些“额外”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。

[0060] 根据本发明的实施例，数据的结构被保存在分离的位置中，即文档的数据存储 208 内的一个或多个 XML 数据片断 209 中，从而使得用户能够在呈现 203 中来回移动链接（即，数据绑定 205），而不影响数据结构。从而，XML 数据 209 的结构不会改变，仅有与文字处理软件文档 204 相关联的 XML 数据 209 的呈现改变。这样，对文档 204 中的数据呈现的格式的改变不影响数据存储 208 的结构。通过操纵文档表面 203 用户并不移动实际数据 209——这样用户对呈现拥有完全的控制，而无需考虑在存储 208 中分开维护的数据 209 的衍生物。因此，本发明的实施例允许用户访问与呈现信息分离的自定义 XML 信息。

[0061] 可以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，此处称为 XML 数据 209）“绑定”到文档 204 中的位置。如此处所使用的，XML 数据 209 包括诸如文本（纯文本或丰富格式的文本）、图像、特定类型（日期）的内容等任何类型的内容。数据绑定 205 的结构可以被描述为 XPath 链接，它允许文字处理软件应用程序 202 连接至 / 与之同步化 / 持久链接至与文档 204 相关联的 XML 数据存储 208 中的 XML 节点 213。XML 数据存储 208 较佳地是文档 204 的一部分（即，存储 208 是与文字处理软件文件一起保存的，并随特定文档 204 行进或与之相关联）。数据绑定 205 也可以包含用于控制应如何在呈现（例如，文字处理软件的文档表面 203）和数据存储 208 之间变换数据的信息。文字处理软件应用程序 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）以标准格式存储，但在文字处理软件文档 204 中以较友好的格式呈现同一信息（例如，用户看到的是 1 月 29 日，2004 年，但该数据是按照 29-01-004T12:00:00.0000 来以 dateTime 格式存储的）。作为另一个示例，数据绑定信息可包含图像——对数据 209 而言，该图像被表示为看起来无意义的字符串，但上述同一变换原理意味着用户将在文字处理软件文档 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，以使任何后端进程可以存储 / 操纵该信息。

[0062] 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 表达式来提供所关心的链接的 XML 数据 209（例如，标识了一个或多个节点 213）。一旦被绑定，该内容区域 207 的内容将被链接或绑定至由该 XPath 返回的节点 213 的内容 (XML 数据）。因此，这意味着如果 XML 节点 213 是以由 XPath 返回的 XML 节点 213 改变的方式被添加 / 移除 / 改变的，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 进入如下所述的‘悬挂引用’状态。

[0063] 例如，假定文档 204 包括以下段落，其中“Microsoft Corporation”对应于纯文本内容区域 207（以斜体示出），该内容区域 207 被绑定到该文本的数据存储 208 内的某个链接的 XML 数据 209 的 XPath/contract(1)/header(1)company(1)。显示在呈现 203 上的段落是：

[0064] “Microsoff Corporation is located at One Microsoft Way.”

[0065] 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：

[0066] Document.ContentRegion.Add0.DataBinding.Add(“/contract(1)/header(1)/

company(1)”)

[0067] 相应的链接的 XML 数据 209 可能看起来是（以单引号示出至节点 213 的链接）：

[0068] <contract>

[0069] <header>

[0070] ‘<company>Microsoft Corporation</company>’

[0071] <company>Contoso Corporation</company>

[0072] </header>

[0073] </contract>

[0074] 假设现在用户使用数据存储 208API 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）：

[0075] <contract>

[0076] <header>

[0077] ‘<company>Fabrikam Corporation</company>’

[0078] <company>Microsoft Corporation</company>

[0079] <company>Contoso Corporation</company>

[0080] </header>

[0081] </contract>

[0082] 内容区域 207 上所得的绑定 205 仍旧被绑定到同一 XPath

[0083] (“/contract(1)/header(1)/company(1)”)，因此文档内容可以立即更新来显示该节点 213 的新内容：

[0084] “Fabrikam Corporation is located at One Microsoft Way.”

[0085] 根据本发明，如果文字处理软件文档 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 对任一所链接的内容源的改变作出反应。从而，如果文档 204 的范围是数据绑定的，那么对相关联的 XML 数据 209 中的 XML 节点 213 的内容的改变将导致内容区域 207 的文本的自动改变。相应地，如果文档 204 的范围是数据绑定的，那么改变文档 204 中该绑定的内容区域 207 的文本会导致对相应的 XML 数据 209 中的 XML 节点 213 的内容的自动改变。即，具有相同绑定 205 的多个内容区域 207 可以存在于文档 204 中的多处。例如，具有至名字的数据绑定 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使该文本与 XML 数据存储 208 同步，XML 数据存储 208 又将在文档 204 中存在至该节点 213 的数据绑定 205 的内容区域 207 的任何地方反应该改变。

[0086] XML 数据 209 中的 XML 节点 213 可与文档 204 具有一对多的关系，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 引用。只要更新文档 204 中数据绑定的内容区域 207，即引起对 XML 数据 209 中适当的 XML 节点 213 的改变，这又使文档 204 中的其它内容区域 207 中的所有其它相关联的绑定 205 使用该新文本来更新。例如，假设文档 204 的头部中的内容区域 207 包含为某个 XML 数据 209 中的 <title/> 节点指定 XPath 表达式的数据绑定 205，而文档 204 的正文中的另一个内容区域 207 也包含至该同一元素的数据绑定 205。根据本发明，这两者会显示相同的内容，即使它们具有不同的格式。如果用户编辑文档 204 的正文中的内容区域 207 中的内容，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，引起同样指定该 XML 节点 213 的、文档 204

中与绑定 205 相关联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用于将文档中的多个位置绑定至数据存储 208 中的单个 XML 节点 213 的机制，当前将所有这三个位置的内容链接到单个数据源。从而，文档 204 中被链接到 XML 数据 209 中的同一节点 211 的的所有内容区域 207 的内容是完全相同的。

[0087] 这样的说明性示例是典型的报告文档，其中用户通常可以在若干位置显示标题：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）。正常地，用户将必须在每个位置打出标题，确保如果这三个位置中的任何一个的标题改变，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，总是太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，一旦数据存储 208 被置于包含用户想要在文档 204 中显示的 XML 数据 209 的位置中，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。

[0088] 因此，所有这三个位置的内容被链接或绑定至单个数据源 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，将自动引起用户的文本被推入至底层 XML 数据 209，然后又被推入至文档 204 中含有相应的数据绑定 205 的内容区域 207 的其它位置（在该情况下，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同。根据本发明的实施例，文档的各个区域可以用多种方式来呈现（大号粗体、小号斜体等），但数据存储 208 中的数据结构保持不变。

[0089] 悬挂引用

[0090] 根据本发明的实施例，用户也可以指定没有目标的 XPath 表达式——它们指定的目标 XML 节点 213 不存在于数据存储 208 中的 XML 数据 209 中。数据绑定 205 没有‘忘记’其期望目标节点 213，而是进入‘等待’状态，在该状态中，它不被连接至任何特定 XML 数据 209，而是等待来查看期望节点 213 是否出现在后台 XML 数据存储 208 中的 XML 数据 209 中。这对于文档组装情形特别有用，在该情形中，文档 204 的每一部分（例如，标准封面、封底、以及重复使用的条款）可包含数据绑定 205，数据绑定 205 仅应在这些部分被组装为单个最终文档 204 中时被填充。在该情形中，文档创建者指定每一文档‘部分’中的内容区域 207 内至不存在于该部分中的 XML 数据 209 中的 XML 节点 213 的数据绑定 205（例如，封面可能包含带有至 <Title/> 的 XML 元素和 <Date/> 的 XML 元素的绑定 205 的内容区域）。当该部分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，但只要该部份一被添加到包含期望数据 209 的文档中，数据绑定 205 立即连接（同步）至数据 209，并显示正确的内容——即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。

[0091] 当内容区域 207 上的数据绑定 205 没有被成功链接到已链接的 XML 流中的节点 213 时（即，内容区域中绑定的状态），发生一种类型的悬挂引用 215。当从已链接的 XML 流中替换 / 移除节点 213 时，作为结果，一个或多个数据绑定 205 可能变为悬挂引用 215。较佳地，如果数据绑定 205 由于其 XPath 而具有悬挂引用 215，那么文字处理软件应用程序 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 不再解析至任何节点 209 时可能发生这一情况。每当数据存储 208 向文字处理软件文档 204 发送某个 XML 数据 209 的更新的消息时，文字处理软件应用程序 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）。如果文字处理软件应用

程序 202 解析悬挂引用，那么数据存储 208 的内容较佳地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。较佳地，使用可通过一个或多个编程环境 206 访问的简单 API 层来展示悬挂引用。

[0092] 作为示例，假设文字处理软件 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) 的纯文本内容区域 207 数据：

[0093] “Microsoft Corporation is located at One Microsoft Way.”

[0094] 相应的 XML 数据 209 可能看起来是（以单引号示出至节点 213 的链接）：

[0095] <contract>

[0096] <header>

[0097] <company>Fabrikam Corporation</company>

[0098] <company>Contoso Corporation</company>

[0099] ‘<company>Microsoft Corporation</company>’

[0100] </header>

[0101] </contract>

[0102] 如果诸如开发员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）：

[0103] <contract>

[0104] <header>

[0105] ‘<company>Fabrikam Corporation</company>’

[0106] <company>Contoso Corporation</company>

[0107] <company>Microsoft Corporation</company>

[0108] </header>

[0109] </contract>

[0110] 所得的文档 204 中内容区域 207 上的数据绑定 205 维护至同一 XPath 的链接，因此数据绑定 205 变为至当前不存在的 \contract\header\company(3) 的悬挂引用 215：

[0111] <contract>

[0112] <header>

[0113] <company>Ford Corporation</company>

[0114] <company>Intel Corporation</company>

[0115] </header>

[0116] </contract>

[0117] 这意味着在内部有断裂的链接，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，

[0118] “Microsoft Corporation is located at One Microsoft Way.”

[0119] 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 立即变为指向已删除 XML 数据 209 的悬挂引用 215。如果数据绑定 205 包含悬挂引用 215，那么文字处理软件应用程序 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，文字处理软件应用程序 202 试图将这些链接重新附加至相关联的 XML 数据存储 208 中的任何其它可用的 XML 数据 209。如果任何数据绑定 205 的确解析到另一个 XML 数据 209 中的节点 213，那么所有的悬挂引用 215 都关联至该 XML 数据 209，从而更新数据绑定 205 当前连接的相关联内容区域 207。如果该 XML 数据 209 没有导致对任何悬挂引用 215 的有效数据绑定 205，那么文字处理软件应用程序 202 对数据存储 208 中的每一 XML 数据 209 等执行类似的检查。如果没有 XML 数据 209 可用于悬挂引用 215，那么该绑定保持为至原始 XML 数据 209 的悬挂引用 215。

[0120] 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。图 3 中所示的过程 300 在 302 处开始，那时用户使用文字处理软件应用程序 202 打开文字处理软件文档 204。在 304 处，文字处理软件应用程序创建数据存储 208，然后在 310 处使用存储在文字处理软件文档 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 填充数据存储 208。数据存储 208 较佳地作为文档 204 的一部分被包括在内，但在文档编辑表面 203 上不可见。可以理解，可在创建内容区域 207 和数据绑定 205 之前装载数据存储 208。同样地，可在数据存储 208 之前创建内容区域 207。换言之，图 3 中所示的各个操作不必以任何特定顺序来执行，并可根据特定用户的偏好来实现。

[0121] 在 306 处，用户创建存在于文档 204 的表面 203 上的一个或多个内容区域 207。注意，也可以从文档 204 的现有内容读取这些内容区域。在 308 处，用户可通过提供特定的链接的 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 成为绑定的或悬挂的。如上所述，每一节点 213 可被绑定至多个内容区域 207，内容区域 207 中的每一个指定至同一 XML 节点 213 的数据绑定 205。而且，数据绑定 205 可被链接到多个数据存储 208。在 309 处，用户可以在内容区域 207 或者数据存储 208 中创建 XML 数据。在 310 处，文字处理软件应用程序 202 将所有 XML 数据装载至数据存储 208 中。在 312 处，文字处理软件应用程序 202 从文档 204 或按照用户所请求的 (306) 装载内容区域 207，并且在 314 处，文字处理软件应用程序 202 从文档 204 或按照用户所请求的 (308) 装载数据绑定 205。在 316 处，文字处理软件应用程序 202 检查是否存在与特定数据绑定 205 所指定的节点 213 相关联的 XML 数据 209。

[0122] 如果 XML 数据 209 不存在，那么在 318 处，文字处理软件应用程序 202 确定同一 XML 名字空间内是否存在其它 XML 数据 209。在 316 处，如果 XML 数据 209 被定位在数据存储内，那么在 320 处，文字处理软件应用程序 202 确定是否存在所指定的 XPath 的相关联的 XML 节点 213。如果该 XPath 存在，那么在 322 处，文字处理软件应用程序 202 通过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至一个或多个相关联的 XML 节点 213。如果在 320 处没有定位到 XPath，那么在 324 处，文字处理软件应用程序 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 处，在数据存储 208 内的同一 XML 名字空间中找到其它 XML 数据 209，那么在 326 处，文字处理软件应用程序 202 再次检查在该数据内是否存在 XML 节点 213。

[0123] 如果 XML 数据 209 不存在，那么在 324 处，文字处理软件应用程序 202 将特定数据绑定 205 标记为悬挂引用（进入悬挂状态）。如果在 326 处发现 XML 数据 209 存在，那么在 328 处，文字处理软件应用程序 202 在 XML 数据 209 内搜索期望的 XPath。如果在 320 处找

到节点 213，那么在 322 处，文字处理软件应用程序 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至一个或多个相关联的 XML 节点 213。

[0124] 应该理解，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机器逻辑电路或电路模块。实现方式是取决于实现本发明的计算系统的性能要求进行选择的问题。从而，组成此处的本发明的实施例的逻辑操作被不同地称为操作、结构设备、动作或模块。本领域技术人员将认识到，这些操作、结构设备、动作以及模块能够用软件、固件、专用数字逻辑、及其任何组合实现，而不背离如所附权利要求书中所阐述的本发明的精神和范围。

[0125] 以上说明书、示例及数据提供了制造和使用本发明的组成部分的完整描述。因为可实现本发明的众多实施例而不背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。

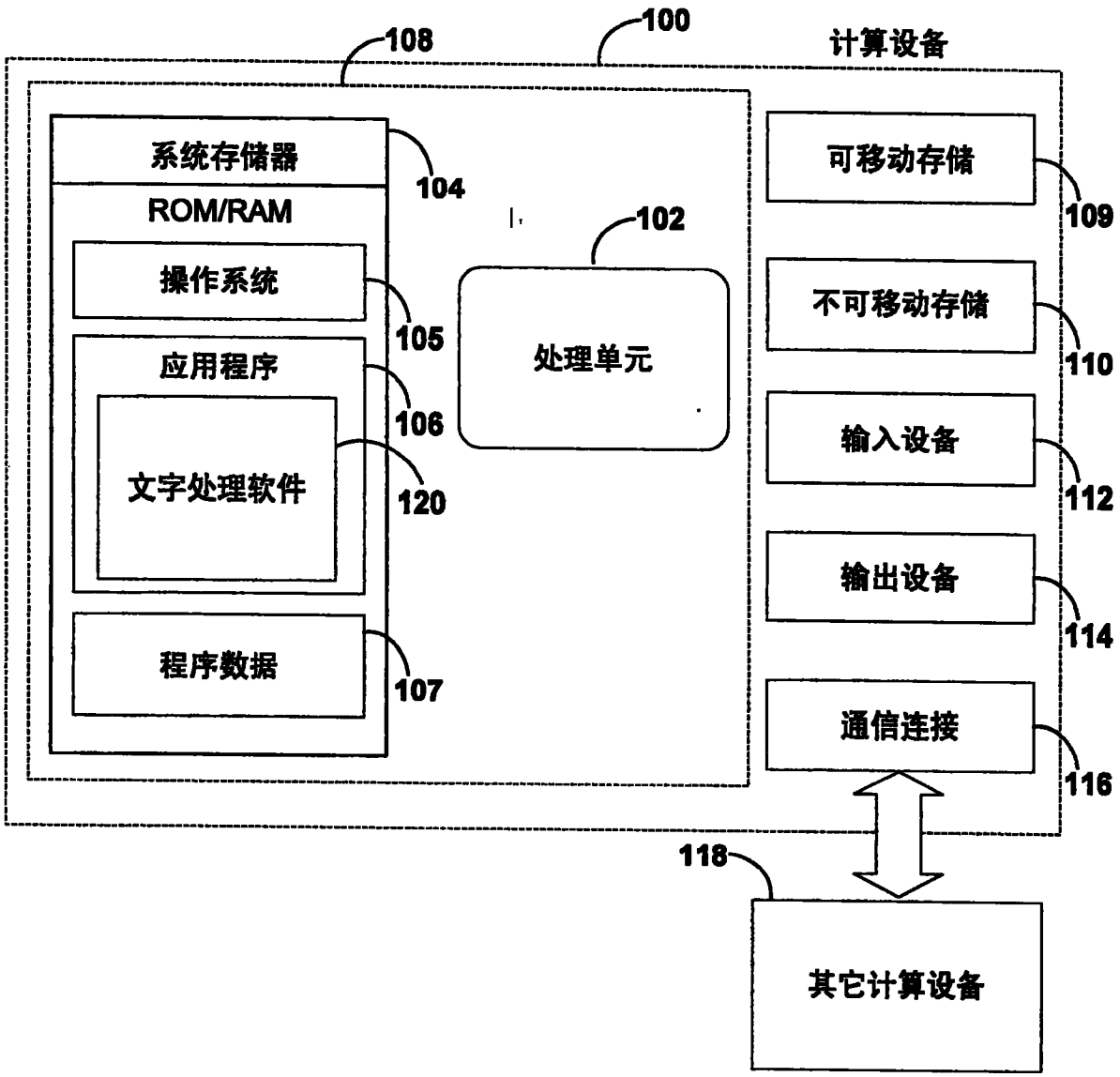

100
计算设备
108
系统存储器
104
ROM/RAM
操作系统
105
应用程序
106
文字处理软件
120
程序数据
107
102
处理单元
可移动存储
109
不可移动存储
110
输入设备
112
输出设备
114
通信连接
116
118
其它计算设备

图 1

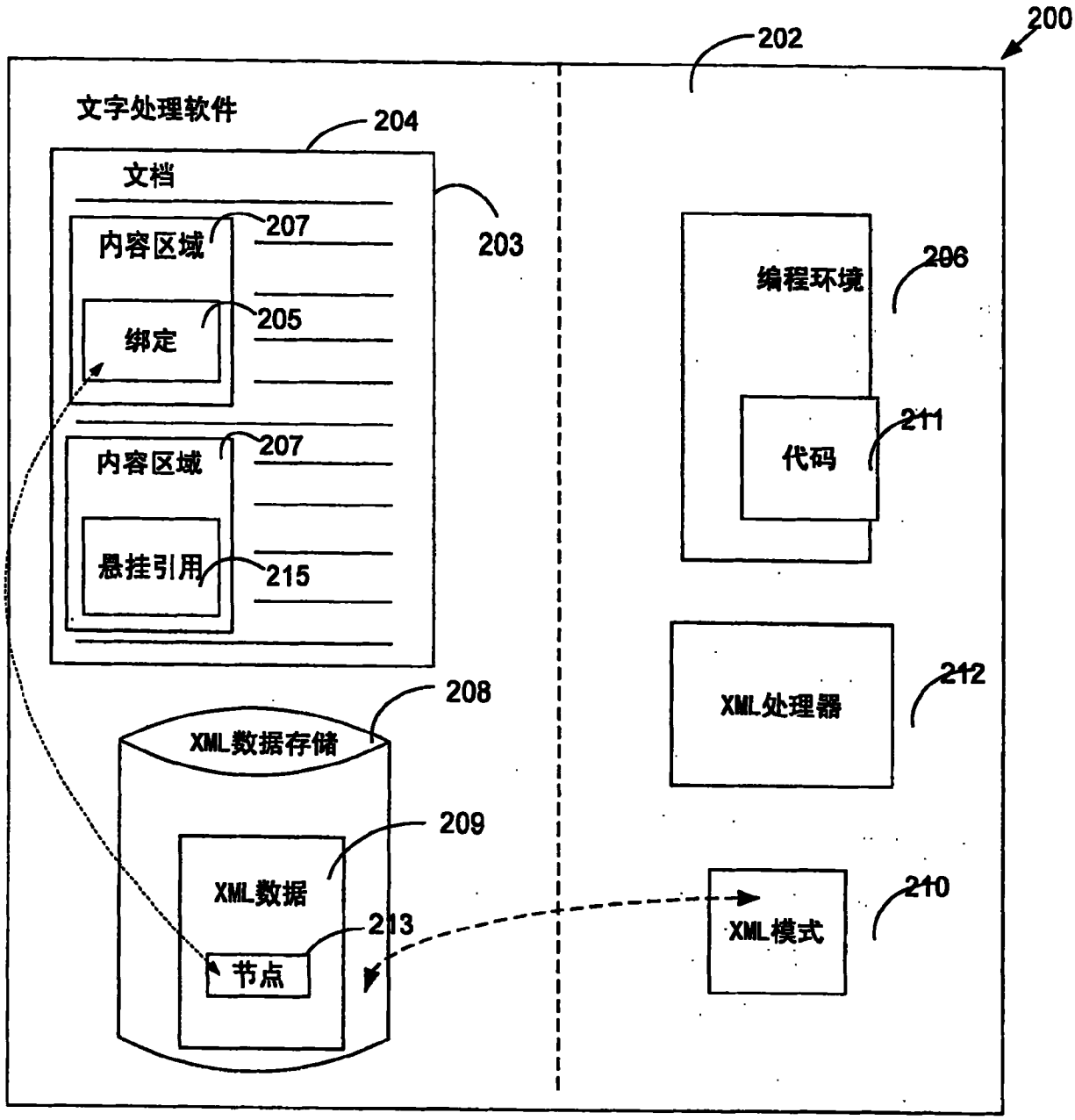

200
202
文字处理软件
204
文档
203
内容区域
207
绑定
205
内容区域
207
悬挂引用
215
编程环境
206
代码
211
XML处理器
212
208
XML数据存储
209
XML数据
213
节点
XML模式
210

图 2

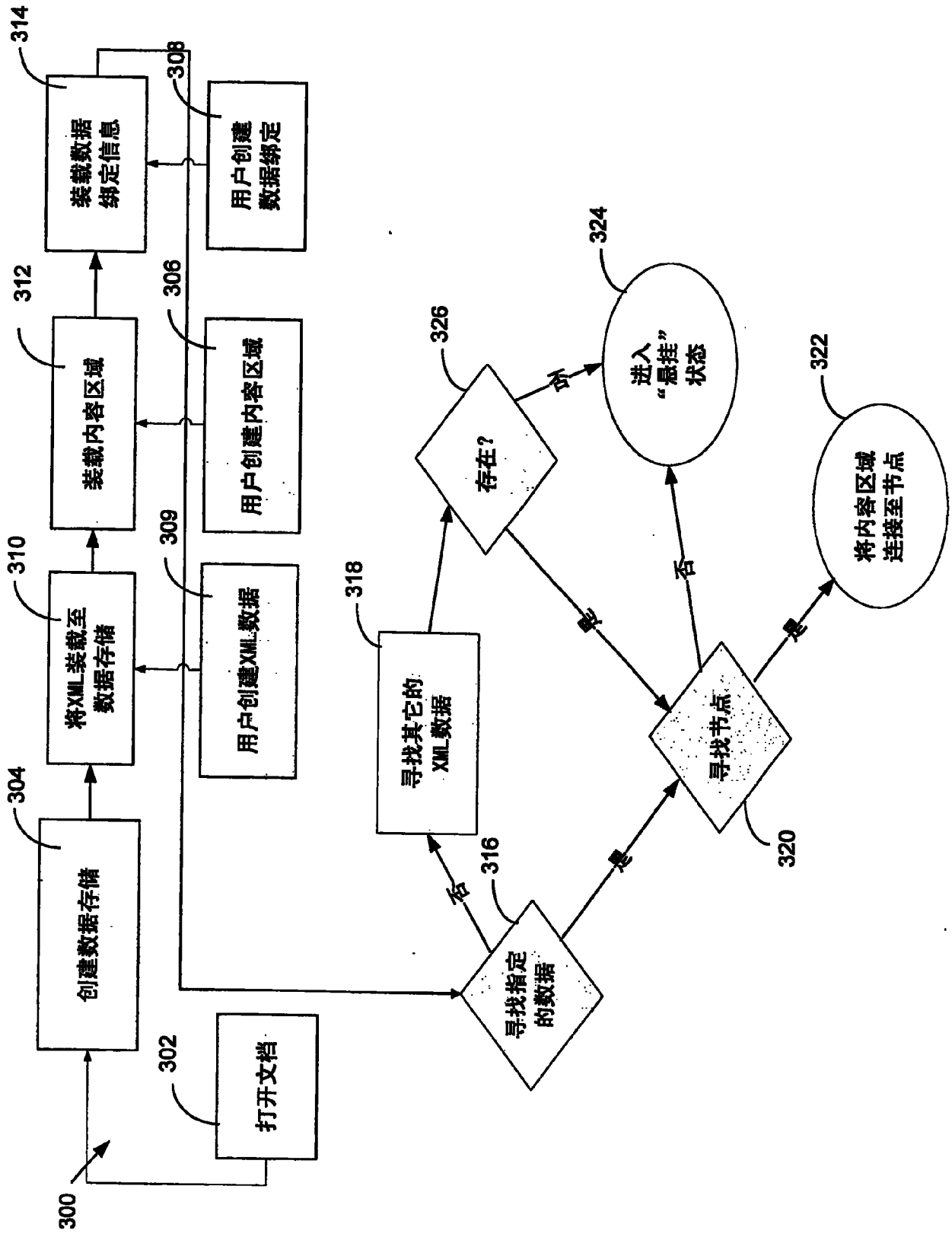

300
302
打开文档
304
创建数据存储
310
将XML装载至数据存储
309
用户创建XML数据
312
装载内容区域
306
用户创建内容区域
314
装载数据绑定信息
308
用户创建数据绑定
316
寻找指定的数据
否
是
318
寻找其它的XML数据
320
寻找节点
否
是
326
存在?
否
是
324
进入“悬挂”状态
322
将内容区域连接至节点

图 3