W3CXMLSchema与文档类型定义 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/144/2021_2022_W3CXMLSche_c104_144869.htm 许多开发者都期待着 XML 模式能够很快取代 DTD 用于指定 XML 文档类型。尽管 David Mertz 相信 XML 模式在开发者宝库中是一种无价工具，但他对该模式将替代 DTD 持怀疑态度。“ XML 问题 ”专栏的这一部分逐步尝试对模式和 DTD 进行比较，并阐明在 XML 模式世界中发生的事件。 虽然 W3C XML Schema在许多场合中胜过 DTD，但仍然还有一些 DTD更胜一筹的领域。开发者要不断地进行艰难的选择（这在 XML世界中是司空见惯的事）。让我们开始对其中一些选项进行区分。 目前形势 将 XML用作数据表示格式的主要原因是有可能指定文档的结构化需求：确定究竟什么类型的内容和子元素可以出现在元素中的一些规则（以及以什么顺序和基数性等）。在传统SGML 派系中，文档规则的表示曾经是 DTD W3C XML 1.0建议书的正式规范确实明确提供了 DTD.不过，有一些 DTD无法实现的相当常见的约束；DTD的主要限制在于它们缺乏数据类型的表达（可以指定一个元素必须包含PCDATA，但无法指定它必须包含例如 nonNegativeInteger ）。 还有一个次要问题，即 DTD无法简化子元素基数性的规范（可以简洁地指定“ 一个或多个 ”子元素，但即使可能，指定“ 七到十二之间 ”也会过分冗长，甚至完全曲解）。 为了对付 DTD的各种限制，一些 XML用户曾呼吁一些指定文档规则的替代方法。总是有可能从编程上检查 XML文档中的条件，但从本质上说，往往更需要施加更严格的标准，即“ 一个不满足一组正式规则的文档就是 无

效文档 ”。W3C XML Schema是满足这些要求的一个主要答案（但不是这里唯一可供选择的模式）。StevenHolzner 在 XML 内幕中将 XML模式归纳为具有以下特征，值得在这里重申： 随着时间的推移，许多人都向 W3C 抱怨 DTD太复杂，并要求使用一些更简单的方法。W3C听取并指定了一个委员会来解决这一问题，然后拿出了一个比以往任何 DTD都复杂得多的解决方案 （p.199）。 Holzner 继续道 几乎所有XML 程序员都同意（包括我自己） 如果不考虑其复杂性，W3C XML Schema还是提供了许多重要能力，并值得用于许多确认规则类。来源：www.examda.com 至少有两个基本的和概念性的缺陷存在于所有 “ 到处模式 ” 的目标中。第一个问题是刚刚于2000 年 12 月 15 日结束其复审阶段的 W3C XML Schema CandidateRecommendation，不包含任何实体；通过扩展，它可以包括参数实体。第二个问题是，尽管存在增强的表达方式，仍然有许多文档规则不能用XML 模式表示（一些建议提议利用 XSLT来增强确认表达，但也可能存在和使用其它方法）。换句话说，模式无法执行DTD长期以来能够执行的所有操作，而从另一方面来讲，模式也无法表达人们希望对文档施加的进一步规则的完整集合。从更实际的角度来说，用于XML 模式的工具不如用于 DTD的工具来得成熟（特别在确认方面，这是核心问题）。 XML文档确认规则整体上仍处于混乱状态。不幸的是，我无法预言每件事的最终结果将会怎样。（有关何时使用 DTD 可能比较有意义的摘要，请参阅侧栏 “ 何时使用DTD. ”）同时，让我们看看 DTD 和 XML模式能够表达哪些内容的一些具体细节。丰富的类型 W3C XML Schema真正出色的地方是在表达属性值和元素内容的类型约

束上。而这恰恰是 DTD最薄弱的地方。除了提供非常丰富的一组内置 simpleType 以外，XML 模式还允许您使用类似规则表达式的语法派生出新的 simpleType 。内置类型包括您在使用编程语言时遇到的： string 、 int 、 float 、 unsignedLong 、 byte 等等；但它们还包括大多数编程语言生来不具备的一些类型： timeInstant （即日期／时间）、 recurringDate （年中的天）、 uriReference 、 language 、 nonNegativeInteger 。 例如，使用 DTD时，将有类似于清单 1 中的声明： 清单 1：DTD "item" 元素定义 使用 W3C XML Schema时，声明可以更具体（对 W3C Schema 的最初规定稍有修改）：清单 2：XML 模式 "item"元素定义 遇到以下情况时，DTD 仍然是您的首选：文档规则的简洁表示很重要。 希望下游用户能够通过内部参数集覆盖并将类型专门化。 您的文档规则主要考虑元素的嵌套而不是内容的语义约束（如同使用散文标记）。 惯常使用的工具支持 DTD 胜于支持模式。 从表面上看，这些元素定义中有两个显著特性。首先是模式本身是格式完整的 XML 实例，其标记使用 xsd 名称空间（实际上，DTD也是这样，但它只有处理指令，而没有这样的内容）；其次（根据第一点的结论）是模式远比DTD 繁琐。 除了语义方面的准确性以外，还可以看到模式示例能够执行一些DTD 不可能完成的操作。 prodName 类型在定义之间基本上是相同的，但模式中的 USPrice 和 shipDate 规范分别是 decimal 和 date 类型。 作为文本文件，具有这些元素的 XML实例在元素内部包含一些 ASCII（或Unicode）字符；不过，具有模式意识的确认器可以在 decimal 和 date 元素内部要求更具体的字符格式（其它类型也是一样）。 更有趣的是属性 partNum ，它属于派生的专

门类型。类型 SKU 不是内置类型，而是跟在 "SKU"声明中给定模式后的一系列字符（具体来说，它必须有三位：一个连线和两个大写字母，按这样的顺序）。也有可能将 SKU 用于元素类型；它在这种情况下定义属性只是一种巧合。 在元素定义的 DTD版本中，所有这些有趣的（如果是专门的，也可能相当复杂）类型一定简单地称之为 PCDATA ，至于字符数据是什么样没有更多说明（在属性情况中是 CDATA ）。 在类型丰富的元素／属性值中，模式巧妙地从描述 XML实例的语法渐变到描述其语义。语法分析纯化论者可能会就我的描述提出异议：“ 内置模式类型是从语法上定义的，因此构建在这些内置类型上的模式在形式上也是符合语法的。 ”但在实际情况中，当声明一个给定的元素必须是 date 时，您实际上希望的是让元素包含一个日期。当然，表达语义信息不是件坏事，但有人会争论说最好同样将它限制在应用程序级别，而不是格式声明。毕竟有一些语义特性 即使是简单的特性避开了模式，但在应用程序中和模式所表达的内容一样重要。 例如，当然"stock-keeping unit" 必须类似于"999-AA" ；但可能您还提供在十三以内的小装置。 integer 被13 整除性不能使用 XML 模式表达（因此您仍然无法在这一级别上为 widgetquantity提供所需的约束）。这里的重点是说，即使有模式（胜过DTD）的额外能力，仍可能需要在应用程序级别上执行后确认来确定 XML文档是否 在功能上有效。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com