



第四章

名稱空間

名稱空間 (namespace) 在 XML 中有兩個目的：

1. 為了區分不同 XML 應用中，使用同樣名稱的元素和屬性。
2. 在單一 XML 應用中歸類相關的元素和屬性，讓軟體能容易地識別它們。

第一個目的很容易解釋和體會，但在實用上，則是第二個較為重要。

名稱空間在實作上乃是將每個元素和屬性加上一個前置字 (prefix)。每個前置字都對應到一個 URI。預設的 URI 也可以提供給未加上前置字的元素。附屬於同一個 URI 的元素和屬性，會屬於同一個名稱空間。標準 URI 可用來識別不同 XML 應用的元素。

為什麼需要名稱空間？

一些文件兼有來自於不同 XML 應用的標記。例如，XHTML 文件可能含有 SVG 圖片和 MathML 方程式。XSLT 樣規 (stylesheet) 包含了 XSLT 指令和來自於結果樹字彙的元素。XLink 則總是和它們所出現之文件的元素共生，因為 XLink 本身只定義屬性，而未定義任何元素。

在某些情形下，這些應用可能使用同樣的名稱，來代表不同的事物。例如，SVG 中的 `set` 元素是個抽象的容器，用來賦予一群元素一個共通特性；而 MathML 中的 `set` 元素，則代表了一個數學上的集合，例如所有正偶數之集合。你應清楚何時該用 MathML 的 `set`，而何時又該使用 SVG 的 `set`，否則正確合法的驗證、呈現、索引以及許多其它工作，可能會因受到混淆而失敗。

請考慮範例 4-1。這份範例是個簡單的繪畫作品清單，包含了每幅畫的名稱、畫家、作畫日期，以及一些簡單的描述：

範例 4-1：畫作清單。

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<catalog>

  <painting>
    <title>Memory of the Garden at Etten</title>
    <artist>Vincent Van Gogh</artist>
    <date>November, 1888</date>
    <description>
      Two women look to the left. A third works in her garden.
    </description>
  </painting>

  <painting>
    <title>The Swing</title>
    <artist>Pierre-Auguste Renoir</artist>
    <date>1876</date>
    <description>
      A young girl on a swing. Two men and a toddler watch.
    </description>
```

```

</painting>

<!-- 還有更多畫作 ... -->

</catalog>

```

現在假設範例 4-1 將會以 web 網頁的形式被提供，而你希望它能被搜尋引擎存取。一種可能的方法是：使用資源描述框架（Resource Description Framework, RDF），將 metadata 嵌入此網頁中。如此描述該網頁，以利任何搜尋引擎或網路爬蟲的工作。使用 Dublin Core metadata 字彙（<http://purl.oclc.org/dc/>）它是一種圖書編目分類資訊所使用的標準字彙，並且可以用 XML 或其它的語法來編碼。那麼這個網頁的 RDF 描述會如下：

```

<RDF>
  <Description about="http://ibiblio.org/examples/impressionists.xml">
    <title> Impressionist Paintings </dc:title>
    <creator> Elliotte Rusty Harold </creator>
    <description>
      A list of famous impressionist paintings organized
      by painter and date
    </description>
    <date>2000-08-22</date>
  </Description>
</RDF>

```

此處我們使用 RDF 中的 Description 和 RDF 元素以及 Dublin Core 的元素：title、creator、description 和 date。我們無法選擇要不要使用這些名稱，因為它們已被建立於各自的規格中。如果我們想要一個能讀懂 RDF 和 Dublin Core 的標準軟體，且能讀懂我們的文件，那麼就必須使用這些名稱。範例 4-2 將這份描述和真正的畫作清單結合。

範例 4-2：一份畫作清單，包含了關於清單的編目資訊。

```

<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<catalog>

  <RDF>
    <Description

```

範例 4-2：一份畫作清單，包含了關於清單的編目資訊。（續）

```
    about="http://ibiblio.org/examples/impressionists.xml">
<title> Impressionist Paintings </title>
<creator> Elliotte Rusty Harold </creator>
<description>
    A list of famous impressionist paintings organized
    by painter and date
</description>
<date>2000-08-22</date>
</Description>
</RDF>

<painting>
    <title>Memory of the Garden at Etten</title>
    <artist>Vincent Van Gogh</artist>
    <date>November, 1888</date>
    <description>
        Two women look to the left. A third works in her garden.
    </description>
</painting>

<painting>
    <title>The Swing</title>
    <artist>Pierre-Auguste Renoir</artist>
    <date>1876</date>
    <description>
        A young girl on a swing. Two men and a toddler watch.
    </description>
</painting>

<!-- 還有更多畫作 ... -->

</catalog>
```

現在我們發現一個問題。好幾個元素被重覆使用於文件的不同部份，來代表不同的意義。title 元素用來作為整個網頁的標題，也用作表示畫作的標題。date 元素則被用來表示網頁製作的日期和作畫的日期。description 元素的其中一個描述了整個網頁，而另一個則用以描述畫作。

這種情形造成了各種問題。由於具有相同名稱的 catalog 和 Dublin Core 元素會有不同的內容模型，使得正確合法性的驗證變得困難。對網頁瀏覽器而言，顯示畫作的敘述時，你可能希望隱藏網頁的敘述部份，但並非所有的樣規語言都能分辨這兩者。處理軟體可能看得懂 Dublin Core 的 date 元素之日期格式，卻無法了解畫作中 date 元素的日期格式。

我們可以改變元素名稱，例如，使用 painting_title 來代替 title，使用 date_painted 代替 date，等等。但是，如果有一大堆文件已經用舊的字彙來標記，要將它們全換成新版本的字彙可不是件容易的事。更何況，若要將所有情況都做這種改變，也不太可能，特別是當名稱衝突發生在兩三種標準字彙之間，而不光只在你自定的字彙與某種標準字彙之間時。比方說，RDF 就幾乎無法避免使用和 Dublin Core 元素相同的名稱，像是 Description 和 description。

在其它狀況下，名稱衝突可能不會發生，但是它對於想快速、果斷地決定某個元素或屬性是屬於哪種 XML 應用的軟體，仍然十分重要。例如，一個 XSLT 處理器必須分辨 XSLT 指令和文字化的結果樹元素。

名稱空間的語法

名稱空間為元素和屬性指派 URI，以消除同名所產生的模稜兩可。一般說來，來自於同一個 XML 應用的所有元素和屬性，都會被指派到同一個 URI；而來自於不同 XML 應用的元素和屬性，所被指派的 URI 也就不同。URI 將元素和屬性劃分成不重疊的集合。具有相同名稱卻有不同 URI 的兩個元素，視為不相同；反之，名稱和 URI 都相同者，才是相同的元素。名稱空間和 XML 應用之間，通常是一對一的對應關係，然而，少部份 XML 應用會使用多個名稱空間，來區分該應用中的不同部份。例如，XSL 就為 XSL 轉換 (XSLT) 和 XSL-FO，使用了不同的名稱空間。

全稱名稱、前置字與本地部份

因為 URI 常含有像是 /、% 和 ~ 這些 XML 名稱裡不合法的字元，所以我們用較短的前置字 (prefix)，像是 rdf 和 xsl，在元素和屬性名稱中代表這些 URI。每個前置字都恰好關聯到一個 URI。兩個名稱如果具有代表相同 URI 的前置字，那

麼便是在同一個名稱空間之中；相反地，若兩個名稱的前置字關聯到不同的 URI，那麼便是在不同的名稱空間中。名稱空間中的元素和屬性之名稱會包含一個冒號 (:)：

```
rdf:description
xlink:type
xsl:template
```

在 : 之前的所有東西，稱作前置字。而在 : 之後的所有東西，稱作本地部份 (local part)。包含冒號在內的完整名稱，稱作全稱名稱 (qualified name)、QName 或未處理的名稱 (raw name)。前置字用來辨別元素或屬性所歸屬的名稱空間，本地部份則用以識別在該名稱空間中的元素和屬性。

在一份包含了 SVG 與 MathML 之 set 元素的文件中，其中一個可能是 `svg:set` 元素，而另一個是 `mathml:set` 元素。這個區別可減少元素之間的混淆。在一個將文件轉換成 XSL-FO 的 XSLT 樣規中，XSLT 處理器可以將具有 `xsl` 前置字的元素，視為一個 XSLT 指令，而具有 `fo` 前置字的，則被視為文字化結果之元素。

前置字可以由任何除了冒號 (:) 之外的合法 XML 名稱所組成。以 `xml` 三個字元所開頭的前置字，會被保留給 XML 和與它相關的規格所用。除此之外，你可以用任何合宜的方式，作為前置字名稱。在 XML 1.0 中，新加入的一個限制是，本地部份也不可以包含冒號。簡言之，冒號在 XML 中唯一合法的使用方式，是用於在全稱名稱中，分開名稱空間前置字和本地部份。

將前置字連結到 UR

每個全稱名稱的前置字都必須關聯到一個 URI。例如，所有 XSLT 1.0 樣規中的 XSLT 元素，都關聯到 `http://www.w3.org/1999/XSL/Transform` URI。另外，慣用的前置字 `xsl` 用來代替 `http://www.w3.org/1999/XSL/Transform` 這個較長 URI。

要將前置字連結 (bind) 到名稱空間 URI 的方法：為某個具有全稱名稱的元素 (或它的某個祖先元素) 加上一個名為 `xmlns:prefix` 的屬性。prefix 為你真正前置字的名稱。例如，`rdf:RDF` 元素的 `xmlns:rdf` 屬性，將前置字 `rdf` 連結到名稱空間 URI `http://www.w3.org/TR/REC-rdf-syntax#`。

URI 慣用法

你不可以直接使用 URI 作為前置字。首先，大部份 URI 中都有的斜線，即不為 XML 名稱的合法字元。然而，直接使用完整的 URI，而不使用特定的前置字來代表，有時相當有用。在許多 XML 郵寄名單和 XML 文件中，可以看到的一種慣用法是：將 URI 以大括弧 ({}) 括起來，放在名稱之前。全稱名稱 `xsl:template` 可被寫為 `{http://www.w3.org/1999/XSL/Transform}template` 來代替。但是這種方式只在，當 URI 本身很重要，而前置字並不然時，便於人與人之間的溝通。XML 解析器和 XSLT 處理器皆不會接受，也看不懂這種長格式。

```
<rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#">
  <rdf:Description about="http://ibiblio.org/examples/impressionists.xml">
    <title> Impressionist Paintings </title>
    <creator> Elliotte Rusty Harold </creator>
    <description>
      A list of famous impressionist paintings organized
      by painter and date
    </description>
    <date>2000-08-22</date>
  </rdf:Description>
</rdf:RDF>
```

連結的有效範圍，包括了連結所宣告處的元素以及該元素的內容。`xmlns:rdf` 屬性為 `rdf:RDF` 元素和它的子元素們宣告了前置字 `rdf`。RDF 處理器可以認出 `rdf:RDF` 和 `rdf:Description` 是 RDF 元素，因為兩者都具有一個前置字 `rdf`，該前置字連結到 RDF 規格書中所定義的特定 URI。處理器不會把 `title`、`creator`、`description` 和 `date` 元素當作 RDF 元素，因為它們並未連結到 `http://www.w3.org/TR/REC-rdf-syntax#` 這個 URI 的前置字。

前置字的宣告，可放在所有使用該前置字的最外層之元素中，或者在任一個此元素的祖先元素裡。這可能是文件的根元素或較低層的元素。例如，要將所有的 Dublin Core 元素附屬於名稱空間 <http://purl.org/dc/>，只要在 `rdf:Description` 元素裡加上一個 `xmlns:dc` 屬性即可，如同範例 4-3 所示。可以這麼做是因為這份文件中所有 Dublin Core 元素都會出現在一個 `rdf:Description` 元素裡；但如果文件中這類的元素散佈在各處，那麼將名稱空間宣告放在根元素的起始標籤中，可能會方便許多。如果有需要，一個元素可包含多個不同的名稱空間宣告。

範例 4-3：一份包含 RDF 和 Dublin Core 的文件。

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<catalog>

  <rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#"
    <rdf:Description xmlns:dc="http://purl.org/dc/"
      about="http://ibiblio.org/examples/impressionists.xml">
    <dc:title> Impressionist Paintings </dc:title>
    <dc:creator> Elliotte Rusty Harold </dc:creator>
    <dc:description>
      A list of famous impressionist paintings organized
      by painter and date
    </dc:description>
    <dc:date>2000-08-22</dc:date>
  </rdf:Description>
</rdf:RDF>

  <painting>
    <title>Memory of the Garden at Etten</title>
    <artist>Vincent Van Gogh</artist>
    <date>November, 1888</date>
    <description>
      Two women look to the left. A third works in her garden.
    </description>
  </painting>

  <painting>
    <title>The Swing</title>
```

```
<artist>Pierre-Auguste Renoir</artist>
<date>1876</date>
<description>
  A young girl on a swing. Two men and a toddler watch.
</description>
</painting>

<!-- 還有更多畫作 ... -->

</catalog>
```

這份文件的 DTD 可以為 `dc:description` 和 `description` 元素包含不同的內容模型。一份樣規也可以為 `dc:title` 和 `title` 賦予不同的樣式。某個會依照日期來排列這份目錄的軟體，則會注意 `date` 元素，而忽略 `dc:date` 元素。

在這範例中，沒有前置字的元素，像是 `catalog`、`painting`、`description`、`artist` 和 `title`，並不屬於任何名稱空間。此外，沒有前置字的屬性，如前例中 `rdf:Description` 元素的 `about` 屬性，也不屬於任何名稱空間。若一個屬性所在的元素屬於 `http://www.w3.org/TR/REC-rdf-syntax#` 名稱空間，並不足以讓該屬性本身也屬於這個名稱空間。唯一讓一個屬性歸屬於一個名稱空間的方法是，讓它有個已宣告的前置字，例如 `xlink:type` 和 `xlink:href`。

文件中可對前置字重新定義，讓同一個前置字在不同元素裡指稱不同的名稱空間 URI。在這種情形下，前置字的宣告以最接近祖先元素所宣告的為準。但是，在大多數的情形，重新定義前置字是不大好的想法，除了造成混淆之外，並沒有什麼用途。

名稱空間 UR

許多 XML 應用有慣用的前置字。譬如，SVG 元素通常會使用前置字 `svg`；RDF 元素通常會使用前置字 `rdf`。但是，這些前置字只是習慣用法而已，當你有需要、為求方便，甚至是突發奇想的時候，這些前置字仍然可以改變。當在一個前置字可被使用之前，必須先被連結至一個 URI，像是 `http://www.w3.org/2000/svg` 或 `http://www.w3.org/1999/02/22-rdf-syntax-ns#`。這些 URI 並非前置字，它們都已標準化，在 URI 不變的前提下，前置字可被任意改變。RDF 處理器會注意的是

URI，而非某個特殊的前置字。只要沒人在 w3.org 網域之外使用 w3.org 網域內的名稱空間 URI，而且假設 W3C 可以監控他們的人使用名稱空間的方法，那所有衝突都可避免。

名稱空間 URI 並不一定要指到一份實際存在的文件或網頁。事實上，它們根本不需要用到 http 協定。它們可能使用其它的，像是 mailto 這種 URI 並不指向文件的協定。然而，如果你使用 http URI，來定義你自己的名稱空間，將規格書文件擺在這個 URI 上，會是不錯的主意。W3C 已經受不了接到人們不斷告訴他們，他們的規格書中之名稱空間 URI 是個根本不存在的鏈結，所以，他們乾脆在他們的名稱空間 URI 上，加了一個簡單的網頁。儘管如此，你沒有理由也得這麼做。許多名稱空間 URI 在你使用瀏覽器瀏覽時，都會造成「404-Not Found error」的 HTTP 錯誤。名稱空間 URI 純粹只是形式上的辨別字而已，它們並非某個網頁的位址，也不打算讓別人鏈結。

解析器會逐字比較名稱空間 URI，即使只是一個不重要的地方，都會被視為定義不同的名稱空間。例如，`http://www.w3.org/1999/XSL/Transform`、`http://www.W3.ORG/1999/XSL/Transform`、`http://www.w3.org/1999/XSL/Transform/` 和 `http://www.w3.org/1999/XSL/Transform/index.html` 可能都指到同一個網頁。但是，在 XSLT 樣規中，只有第一個是合法的。如果四種都拿來使用，則會建立四個不同的名稱空間。

使用 xmlns 屬性設定預設名稱空間

你通常會知道某個元素的所有內容皆來自於某個 XML 應用。例如，你很可能在 SVG 的 `svg` 元素裡，只找到同樣屬於 SVG 的元素。要表示一個不含前置字的元素和所有它的子孫元素，都屬於同一個名稱空間，你可以在最外層的元素加上不含前置字的 `xmlns` 屬性：

```
<svg xmlns="http://www.w3.org/2000/svg"
      width="12cm" height="10cm">
  <ellipse rx="110" ry="130" />
  <rect x="4cm" y="1cm" width="3cm" height="6cm" />
</svg>
```

這裡雖然沒有一個元素有前置字，但 `svg`、`ellipse` 和 `rect` 元素都在 `http://www.w3.org/2000/svg` 名稱空間中。

屬性與元素就完全不同了。預設名稱空間只對元素而非對屬性有效。在上例裡，`width`、`height`、`rx`、`ry`、`x` 和 `y` 屬性都不在名稱空間中。

你可以在某個元素中加入一個 `xmlns` 屬性，來改變預設的名稱空間。範例 4-4 是一份將預設名稱空間設為 `http://www.w3.org/1999/xhtml` 的文件。這個名稱空間宣告套用到這份文件的大多處。但是 `svg` 元素也有個 `xmlns` 屬性，把它自己與它內容的預設名稱空間重設成 `http://www.w3.org/2000/svg`。然而，XLink 資訊被包含在屬性中，所以這個屬性必須明確地加上前置字，來表示它位於 XLink 的名稱空間中。

範例 4-4：一份使用預設名稱空間的 XML 文件。

```
<?xml version="1.0"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <head><title>Three Namespaces</title></head>
  <body>
    <h1 align="center">An Ellipse and a Rectangle</h1>
    <svg xmlns="http://www.w3.org/2000/svg"
         width="12cm" height="10cm">
      <ellipse rx="110" ry="130" />
      <rect x="4cm" y="1cm" width="3cm" height="6cm" />
    </svg>
    <p xlink:type="simple" xlink:href="ellipses.html">
      More about ellipses
    </p>
    <p xlink:type="simple" xlink:href="rectangles.html">
      More about rectangles
    </p>
    <hr/>
    <p>Last Modified May 13, 2000</p>
  </body>
</html>
```

預設的名稱空間並未套用到具有前置字的元素或屬性。它們仍然屬於它們的前置字所代表的名稱空間。但是，如果一個不具前置字的子元素，屬於某個有前置字的元素，則該子元素仍然屬於預設名稱空間。

為 xmlns 作屬性宣告

若名稱空間只用來分辨來自某個 XML 應用的元素和屬性，而不能區分同名稱的不同元素時，DTD 可以為該應用的主要容器元素加上一個固定值 (fixed) xmlns 屬性，來保證每個東西都在正確的名稱空間中，而不需要確實加上 xmlns 屬性。例如，下面的 ATTLIST 宣告將所有 svg 元素的預設名稱空間固定為 <http://www.w3.org/2000/>：

```
<!ATTLIST svg xmlns CDATA #FIXED "http://www.w3.org/2000/">
```

這允許你在 svg 元素中忽略 xmlns 屬性。

文件不一定能利用這種忽略所帶來的便利。所需要的是解析器必須能讀取 DTD。所有解析器都會讀取內部 DTD 子集，並處理任何它們從中找到的 ATTLIST 宣告，但一些不做正確合法性驗證的解析器可能因忽略外部 DTD 子集，而被弄混。理想上，你應該使用會讀取外部 DTD 子集的正确合法性驗證解析器，雖然你可能是在驗證功能被關掉的情況下使用它。

解析器如何處理名稱空間？

名稱空間是在 XML 1.0 後才加到規格書中。然而，設計時考慮到要保證往後的相容性。因此，不懂得名稱空間的 XML 1.0 解析器，也不會在讀取那些用到名稱空間的文件時，遇到麻煩。因為冒號在 XML 1.0 的元素和屬性名稱裡，是合法的字元，所以舊版本的解析器只會回報某些名稱包含了冒號。唯一可能的問題出在，當不同的全稱名稱被解析成同一個完整名稱，或當同樣的全稱名稱，在文件的不同處，表示了不同的完整名稱。

一個看得懂名稱空間的解析器，會在它所執行的正常結構良好性檢查之外，加上幾種檢查，特別是它會確定所有的前置字都對應到 URI。使用了未對應前置字的文件會被丟棄，除非是以 XML 1.0 或 XML 規格書裡名稱空間所規定之方式使用的 xml

和 `xmlns`。此外，它會丟棄名稱中包含了多於一個冒號的元素或屬性。除此之外，它的功能幾乎完全像是一個不懂名稱空間的解析器。其它位於純粹 XML 解析器之上的軟體（例如，一個 XSLT 引擎），可能以不同方式看待元素，完全看它們所屬於的名稱空間。然而，只要合乎所有的結構良好性和名稱空間限制，XML 解析器本身通常不在意名稱空間這回事。

有個例外可能會發生在不太可能發生的情形——具有不同前置字的元素卻屬於同一個名稱空間。在這種情況下，懂得名稱空間的解析器會回報這些元素是一樣的，而不懂得名稱空間的解析器則認為它們是不同的。同樣不太可能的情形是，當兩個有相同全稱名稱的元素或屬性，卻在不同的名稱空間之中，因為相同的前置字，在文件的不同處，代表了不同的 URI。比較有可能發生的是：在不同的預設名稱空間中，有兩個沒有前置字的名稱。這兩種情形，懂得名稱空間的解析器會認為它們是不同的；而不懂得名稱空間的解析器則把當它們當成一樣。許多解析器讓你視情況開啟或關閉處理名稱空間的功能。

名稱空間和 DTD

名稱空間完全獨立於 DTD 之外，而且可被用於正確合法或不正確合法的文件。文件可以有一份 DTD 但不使用名稱空間，也可以只使用名稱空間而不用 DTD。名稱空間不會以任何方式改變 DTD 語法，也不會改變正確合法性的定義。例如，一份使用某個名為 `dc:title` 元素的正確合法文件，它的 DTD 必須包含一個 `ELEMENT` 宣告，來為 `dc:title` 元素安排適當的內容：

```
<!ELEMENT dc:title (#PCDATA)>
```

文件中該元素的名稱必須配合 DTD 中宣告的元素名稱——準確地說是包含了前置字的名稱。DTD 不能忽略前置字而只宣告了 `title` 元素。具前置字的屬性也類似如此。例如，若文件內某元素用 `xlink:type` 和 `xlink:href` 屬性，則 DTD 必須宣告 `xlink:type` 和 `xlink:href` 屬性，而不是 `type` 和 `href`。

相反地，如果一個元素使用了 `xmlns` 屬性，來設定預設名稱空間，而不為該元素加上前置字，則在 DTD 中該元素的名稱就必須以沒有前置字的方式宣告。正確合法性驗證器既不知道也不關心名稱空間，它所看到的只是某些元素和屬性名稱恰巧包含了冒號；而它所關心的是：這樣的名稱只要被宣告，即為完全地正確合法。

名稱空間前置字的參數實體參考

DTD 必須宣告包含前置字的名稱，而不只是宣告簡單的名稱，或者宣告本地部份和名稱空間 URI 的某種組合，這種要求使得要在正確合法的文件中改變前置字，變得很困難。問題在於，要改變前置字必須改變所有 DTD 中使用到前置字的宣告。但是，由於當初設計者的高瞻遠矚，參數實體參考可以讓你不需要為這麻煩感到頭疼。

首先，將名稱空間裡前置字和冒號的部份，獨立定義成參數實體：

```
<!ENTITY % dc-prefix "dc">
<!ENTITY % dc-colon ":">
```

接下來，將全稱名稱也定義成參數實體參考：

```
<!ENTITY % dc-title      "%dc-prefix;%dc-colon;title">
<!ENTITY % dc-creator    "%dc-prefix;%dc-colon;creator">
<!ENTITY % dc-description "%dc-prefix;%dc-colon;description">
<!ENTITY % dc-date       "%dc-prefix;%dc-colon;date">
```

定義全稱名稱

不要忽略這一步，也不要試著在 ELEMENT 和 ATTLIST 宣告裡，直接使用 dc-prefix 和 dc-colon 參數實體。這種方法會失敗，因為對於一個實體而言，除了另一個實體的替代文字中被用到之外，其它情況的使用，XML 解析器都會在它的替代文字旁加上額外空白。

接著，在所有宣告中，使用實體參考來代替全稱名稱：

```
<!ELEMENT %dc-title; (#PCDATA)>
<!ELEMENT %dc-creator; (#PCDATA)>
<!ELEMENT %dc-description; (#PCDATA)>
<!ELEMENT %dc-date; (#PCDATA)>
<!ELEMENT rdf:Description
  ((%dc-title; | %dc-creator; | %dc-description; | %dc-date; |)*)
>
```

現在，一份文件就可以透過對參數實體之定義的變更，來改變前置字。在某些情況下，直接編輯 DTD 會改變定義；而在其它情形下，於文件的內部 DTD 子集中覆寫原來的定義，也能達成變更。例如，要將前置字 `dc` 改成 `dublin`，只須在 DTD 裡，正常定義之前的某處，加上以下的實體定義：

```
<!ENTITY % dc-prefix "dublin">
```

若你想用 `xmlns` 來代替前置字的使用，可以將 `dc-prefix` 和 `dc-colon` 實體重新定義成空字串：

```
<!ENTITY % dc-prefix "">  
<!ENTITY % dc-colon "">
```

