DITA

DITA 是基于XML的体系结构，用于编写、制作、交付面向主题的信息类型的内容。DITA的内容可以通过不同的方法进行重用，生成不同的交付内容。

DITA topics

主题根据信息类型的不同，可以分为concept（概念）、task（任务）、reference（引用），和troubleshooting（故障处理）等基本类型，这些主题通过Map文件组织起来形成文档。Map可被认为是文档目录结构，根据文档不同类型，有不同的章节划分方式。所有DITA主题都具有相同的基本结构：标题和内容主体（可选）。DITA主题应该有.dita文件扩展名。

DITA maps

DITA maps是将主题和其他资源组织成结构化信息集合的文档。 DITA maps指定层次和主题之间的关系。DITA maps应该有.ditamap文件扩展名。

重用和过滤

DITA提供了各种机制，包括conref和keyref等内容引用，对内容进行重用。同时通过DITAVAL文件，对不同的读者对象、平台、产品、版本等进行内容过滤。

协作和共享

将内容主题化，将格式统一到样式表，通过Map组织内容章节目录。这些方法使得文档的开发任务可以很方便地分解到各个文档编写人员手中，生成格式统一，内容规范的文档。

DITA文件是基于XML的文本文件，可以很方便地进行存储和传输，实现文档的异地共享，协同作业。

信息类型

信息类型描述了一些主题，如概念，任务或参考。 通常，不同的信息类型支持不同类型的内容。 例如，任务通常具有一组步骤，而参考主题具有一组习惯部分，例如语法，属性和用法。

DITA链接

DITA严重依赖于链接。 其提供链接的目的包括定义发布结构（DITA映射）的内容和组织，主题到主题的导航链接和交叉引用，以及通过引用重用内容。 所有的DITA链接使用相同的寻址工具，使用键和键引用，基于URI的地址或DITA特定的间接地址。

约束

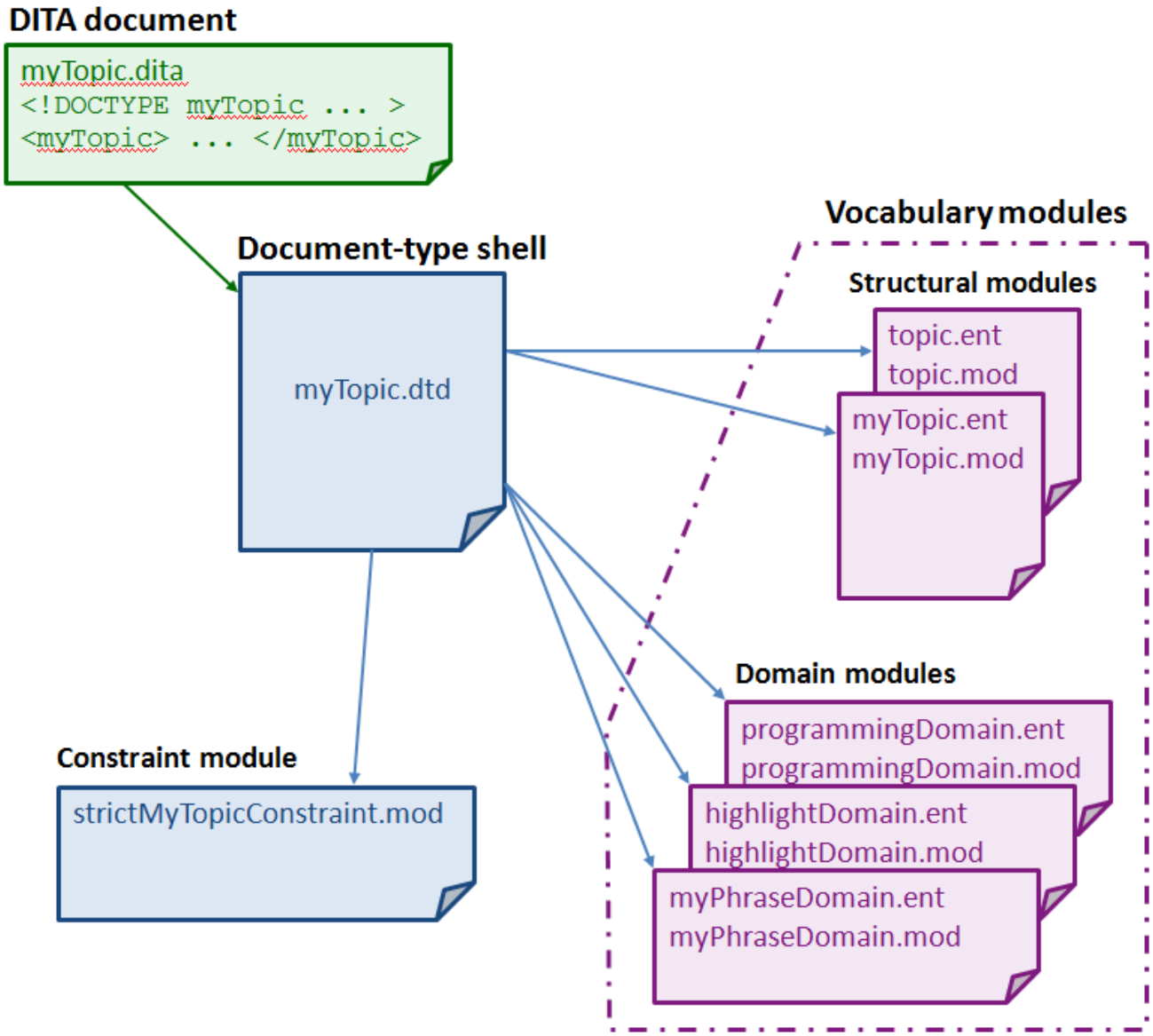
约束模块为相应的词汇表模块定义附加约束，以限制特定元素类型的内容模型或属性列表，从集成域模块中删除扩展元素，或者使用域提供的扩展元素类型替换基本元素类型。

文档类型shell

文档类型shell是一个XML语法文件，它指定DITA文档中允许的元素和属性。文档类型shell集成了结构模块、域模块和约束模块。此外，文档类型shell指定主题是否可以嵌套和如何嵌套。

DITA文档必须有相关的文档类型定义，或者所有必需的属性必须在文档实例中显式显示。大多数DITA文档都有一个相关的文档类型shell。引用文档类型shell的DITA文档可以使用标准XML处理器进行验证。这种验证使处理器能够读取XML语法文件并确定@ domains和@ class属性的默认值。

下图演示了基于dtd的DITA文档、文档类型的shell以及它所使用的各种词汇模块之间的关系。类似的结构也适用于使用其他XML语法的DITA文档。



特化

DITA的特化特性允许创建新的元素类型和属性，同时重用尽可能多的现有设计和代码。这些元素类型和属性是由现有的类型派生出来的。当需要新的结构类型或新域时，就使用特化。

DITA特化可以在您希望对您的设计进行更改时使用，以增强一致性或描述性，或者对于不能使用当前数据模型处理的输出有特定的需求。

dtd语法主题特化模块由两个文件组成:.ent定义用于将模块集成到文档类型的shell dtd中的实体的文件。 .mod文件，声明元素类型。

对于基于XSD的结构模块，有两个XSD文档:

一个* Grp.xsd定义了用于将模块集成到文档类型的shell XSDs。* Mod.xsd该模块声明模块的元素类型和属性。

特化层次结构

结构特化

结构特化是由主题或映射类型开发的。结构特化可以向DITA添加新的文档类型。新文档类型中定义的结构可以直接使用或继承其他文档类型中的元素。例如：concept, task, 和reference 是从topic特化的，而bookmap则是从map特化的。

域特化

域定义特定信息领域或主题领域的标记，如编程或硬件。每个结构类型或域都是它的父类的子类。例如，任务的特化仍然是一个任务，而用户接口域的特化仍然是用户接口域的一部分。

属性的特化

属性特化允许定义新的有条件的处理属性，新属性需要从@props或@ base中特化:

从@props特化的属性被识别为有条件的处理属性

从@ base特化的属性没有与之相关的现有行为

表1显示了API描述的APIdesc信息类型的部分特化。 每一列代表一种信息类型，特化从左到右发生。 也就是说，每种信息类型都是其左边邻居的特化。 每行表示一组映射元素，右侧映射到更一般的等价元素的更多特定元素。

表1. APIdesc specialization

Topic	**Reference**	**APIdesc**
(topic.mod)	(reference.mod)	(APIdesc.mod)
topic	reference	APIdesc
title		APIname
body	refbody	APIbody
simpletable	properties	parameter
strow	property	
section	refsyn	usage

这里声明的大多数内容模型依赖于topic.mod中声明的元素或实体。 因此，如果主题的结构被增强或改变，大部分改变将被自动提取。 另外引用不必重新声明它与主题共享的任何内容。

当一个特化的类型声明新元素时，它必须为新元素提供一个类属性。类属性必须包含特化类型祖先中每个主题类型的映射，甚至包括没有发生元素重命名的映射。映射应以主题开始，并以当前的元素类型结束。

应用一般样式表或转换 因为以新信息类型（例如APIdesc）编写的内容在已有信息类型（例如引用和主题）中映射到等效或较少限制的结构，所以已有的转换和进程可以安全地应用于新内容。默认情况下，新信息类型中的每个专用元素将被视为一般等价物的一个实例。 为了覆盖这个默认行为，可以简单地为该元素类型创建一个新的更具体的规则，然后导入默认的样式表或转换，从而扩展行为，而无需直接编辑原始样式表或转换。通过引用的重用降低了维护成本（每个站点只维护它唯一需要的规则），并且增加了一致性（因为核心转换规则可以集中维护，而对核心规则的更改将反映在导入它们的所有其他转换中）。

@ class属性

每个DITA元素的特化层次结构被声明为@ class属性的值。@ class属性提供了从元素的当前名称到更一般的等价元素的映射。 @ class属性的值有以下语法要求: 最初的“-”或“+”字符后跟一个或多个空格。在结构词汇模块中定义的元素类型使用“-”，在域模块中定义的元素类型中使用“+”。

当词汇模块声明新元素类型时，它必须为其声明的每个元素类型提供一个@ class属性。@ class属性必须包含针对特化类型的祖先中的每个结构类型或域的映射，即使没有元素重命名的发生。映射必须以基本类型的值(例如主题或映射)开始，并以当前元素类型结束。

给定元素类型的特化层次结构必须反映基本类型和特化类型之间的任何中间模块，即使是没有元素重命名的类型。

创建域特化时，新元素仍然需要一个 class属性，但应该以“+”而不是“ - ”开头。这意味着任何泛化转换都会以不同的方式处理元素。

DTD公共标识符

每个文档shell(.dtd文件)或模块组件(.mod或.ent文件)有一个公共标识符。公共标识符可以引用文档类型shell或模块组件的最新版本或特定版本。 OASIS维护的DTD文件的公共标识符使用以下格式: "-//OASIS//DTD DITA version information-type//EN" 版本要么是DITA版本号(例如，1.0、1.1、1.2、1.3)，要么完全省略。 信息类型是topic或map类型驼峰式的命名，例如Concept或BookMap。

DITA to HTML Help (CHM)

htmlhelp转换生成HTML输出，CSS文件以及生成一个Microsoft HTML帮助文件所需的控制文件。除HTML输出和CSS文件外，此转换还会返回以下文件，其中XXX是master DITA map的名称。

Filename	Description
XXX.hhc	Table of contents
XXX.hhk	Sorted index
XXX.hhp	HTML Help project file
XXX.chm	Compiled HTML Help Note: This file is generated only if the HTML Help Workshop is installed on the build system.

运行HTML Help转换，需将transtype参数设置为htmlhelp，或者传递--format = htmlhelp选项到dita命令行。

chm添加页眉页脚

在F:\dita-ot-2.5.4\plugins\org.dita.xhtml\xsl\xslhtml\dita2htmlImpl.xsl找到DEFAULT PAGE LAYOUT，修改代码如下：

```
<xsl:template name="chapter-setup">
<html>
  <xsl:call-template name="setTopicLanguage"/>
  <xsl:value-of select="$newline"/>
  <xsl:call-template name="chapterHead"/>
  <xsl:call-template name="chapterBody"/>
  <div class="hrcopyright">
    <hr size="2"/>
  </div>
  <div class="hwcopyright">世行测控专有和保密信息</div>
  <div class="hwcopyright">版权所有 © 2015 - <xsl:value-of select="fn:year-from-
dateTime(xs:dateTime(fn:current-date()))"/> 北京中科世行测控技术有限公司</div>
</html>
</xsl:template>
```

在F:\dita-ot-2.5.4\plugins\org.dita.xhtml\resource下的.css文件中添加以下代码：设置字体、颜色、居中显示

```
.hrcopyright {
     color: #3f4e5d;
     margin-top: 18pt;
 }

 .hwcopyright {
     text-align: center;
     font-family: Arial;
 }
```

问题： 因为DITA Open Toolkit（简称DITA OT）对中文支持较弱，发布的时候乱码

原因： 导致乱码的原因是ditamap中 map元素没有添加 xml:lang="zh-cn"

根据DITA Specification，map元素中的xml:lang属性可定义整个文档的语言，如果顶层topic没有设置该属性值的话，处理器会使用默认的设置，DITA OT中默认语言是en-us，因此发布的时候 会使用英文的编码和字体，这样导致乱码是难免的

表格和图

表格的标题中英文显示处理

如，表 1而非 Table 1，且汉字和数字之间无空格（表1），英文和数字之间有空格（Table 1）

ditamap中 将map元素的xml:lang属性的值设置为"zh-cn"

F:\dita-ot-2.5.4\xsl\common\strings-en-us.xml中在Figure和Table之后加空格

```
]<strings xml:lang="en-us">        <!-- US English -->

<!-- various labels (not author controlled) -->
<str name="Figure">Figure </str>
<str name="Table">Table </str>
```

在F:\dita-ot-2.5.4\plugins\org.dita.xhtml\xsl\xslhtml\dita2htmlImpl.xsl中找到去掉xsl:text标签中的空格

```
<xsl:otherwise>
 <xsl:call-template name="getVariable">
  <xsl:with-param name="id" select="'Figure'"/>
 </xsl:call-template>
 <xsl:text></xsl:text>
 <xsl:value-of select="$fig-count-actual"/>
 <xsl:text>. </xsl:text>
</xsl:otherwise>
```

在F:\dita-ot-2.5.4\plugins\org.dita.pdf2\cfg\common\vars\zh_CN.xml，去掉'图'和'表'之后的空格

```
<variable id="Figure.title">图<param ref-name="number"/>: <param ref-name="title"/></variable>
<variable id="Figure Number">图<param ref-name="number"/></variable>

<!-- Text to use for table titles. Renders as part of <table>/<title>
     elements. -->
<variable id="Table.title">表<param ref-name="number"/>:<param ref-name="title"/></variable>
<variable id="Table Number">表<param ref-name="number"/></variable>
```

在F:\dita-ot-2.5.4\plugins\org.dita.xhtml\xsl\xslhtml\tables.xsl找到 去掉xsl:text标签中的空格

```
<xsl:otherwise>
  <xsl:call-template name="getVariable">
    <xsl:with-param name="id" select="'Table'"/>
  </xsl:call-template>
  <xsl:text></xsl:text>
  <xsl:value-of select="$tbl-count-actual"/>
  <xsl:text>. </xsl:text>
</xsl:otherwise>
```

Ant是什么

Ant相当于Linux环境下的shell脚本，只不过是用xml文档来编写的。在linux环境中可以通过编写shell脚本封装一系列繁琐而日常需要经常重复的操作。在进行这些操作时，只需运行这个脚本就可以批处理这些操作。Ant脚本也是一样，只不过它一般是为了方便Java项目的编译、运行、测试、打包等工作服务的。

Ant的作用

Ant脚本，通过一个xml文件来制定一系列文件的创建删除任务、编译任务、运行任务、测试任务、打包任务等。可以通过ant指令执行这个xml脚本，来批处理这些任务，这样就可以实现“一键”完成编码后的编译、运行、测试、打包导出等工作。

Ant脚本

一个Ant脚本即可完成至少一个项目的编译、运行、测试、打包等工作。Ant脚本命名：build.xml

脚本内容：

project 节点：一个脚本相当于一个project，用一个project来统领脚本中的众多操作命令。 project 元素是 Ant 构件文件的根元素， Ant 构件文件至少应该包含一个 project 元素，否则会发生错误。

property节点：属性节点，相当于Ant脚本中的变量，通过属性值来携带具体内容。在每个任务中通过${属性名} 访问其属性值，从而获取内容。

target节点：任务节点。 target为ant的基本执行单元，它可以包含一个或多个具体的单元/任务。多个target 可以存在相互依赖关系，target的执行顺序可以有两种方式控制：一种是依赖，depends属性，Adepends B，则B先执行；另一种就是内嵌：在target A中通过 命令执行B任务。

target属性

- if 属性：用于验证指定的属性是存在，若存在，所在 target 才会被执行。
- unless 属性：该属性的功能与 if 属性的功能正好相反，若不存在，所在 target 将会被执行。
- description 属性：该属性是关于 target 功能的简短描述和说明。

echo 命令

在控制台输出信息。包括 message 、 file 、 append 和 level 四个属性。

path节点：表示一个路径

path元素用来表示一个类路径，不过它还可以用于表示其他的路径。

- location 表示一个文件或目录。Ant在内部将此扩展为一个绝对路径。
- refid 是对当前构建文件中某处定义的一个path的引用。
- path表示一个文件或路径名列表。

Ant 的运行

安装好Ant并且配置好路径之后，在命令行中切换到构建文件的目录，输入Ant命令就可以运行Ant.若没有指定任何参数，Ant会在当前目录下查询build.xml文件。如果找到了就用该文件作为构建文件。如果使用了 –find 选项，Ant 就会在上级目录中找构建文件，直至到达文件系统的根目录。如果构建文件的名字不是build.xml ，则Ant运行的时候就可以使用 –buildfile file,这里file 指定了要使用的构建文件的名称，示例如下： Ant –buildfile test.xml使用当前目录下的test.xml 文件运行Ant ,执行默认的目标