Chapter 6

How to Start Topic Mapping Right Away with the XTM Specification

Sam Hunting

XTM Topic Mapping

XTM stands for "XML Topic Maps." By the end of this chapter, you will know 100 percent of what you need to know to start creating XML topic maps, even if you know 0 percent (or even less) now.

As for XML, all you need to know for now is that XML adds the angle brackets to words in documents that otherwise look like plain English (`<topic>`), that when these words have been "marked up" they are called *elements*, and that XML elements live in XML documents, one of which, an XML topic map, we are about to create. (You may choose to stop reading here and check the XML entry in the Resources section near the end of this chapter, or see Chapter 3 on markup.)

In this chapter, we will step through the creation of two topic maps. Because topic maps are simple and intuitive, we're going to start from the bottom up, with the angle brackets, and finish with theory, rather than working from the top down. By working through the examples you will understand

- All XTM elements
- How to merge XTM topic maps
- Some XTM pitfalls

To aid our understanding, we will avoid the following subjects: philosophy, history, politics, and theology (PHPT). Why would this be hard? Topic mapping jargon contains a lot of words that have been hijacked from PHPT; among them *subject*, *topic*, *associate*, *occur*, *resource*, *name*, and that hardy perennial, *is*. Also *indicates* and *identity*.

Fortunately, we can keep it simple by sticking to words in angle brackets like `<topic>`—they, at least, are clearly defined in the XTM specification, into which philosophy, religion, history, and politics do not enter.[1]

Why Topic Maps?

The purpose of topic maps is to interchange knowledge. Knowledge interchange may seem hard (knowledge is seen as subjective or as existing only in the mind), but in fact it can be approached pragmatically.

Let's make an analogy between tomato soup and topic maps.

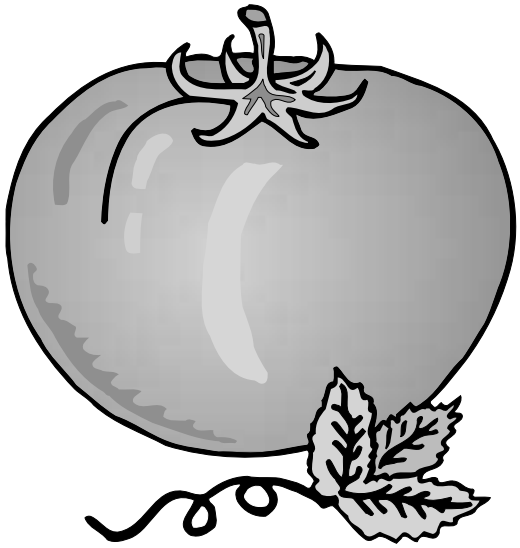
Tomato Soup	Topic Maps
Taste	Knowledge
Can of tomato soup	Topic map document
Directions on the can	Processing methods

Thus, although "taste" is often regarded as very subjective, in fact we exchange it every day, via cans of tomato soup. Subjectivity holds as well for "knowledge," which we can exchange with topic maps. As the taste of tomato soup is interchanged in cans, so knowledge is interchanged in topic map documents. As you follow the directions on the side of the can to make (or at least reconstitute) tomato soup, so topic map software uses specified processing methods to make (or at least reconstitute) knowledge. (See Chapter 4.) If you pour two cans of soup into the same pan you might be said to merge them—but let's not get ahead of ourselves.

Inspired by the humble tomato, let's pick an area to actually map: *cuisine*. Why? It's a tradition of the XTM Authoring Group to share excellent meals. Cuisine also has a variety of rich relationships to express in topic map associations, like recipes and menus. Rationalizing further, cuisine requires the use of several human (in the jargon, *natural*) languages; meeting this requirement will exhibit a powerful feature of topic map applications called *scope*. A print representation of an occurrence of our topic occurs on the next page.

In developing our topic maps, let's imagine that we live in a world much like today's world, except that it is our topic map future, and topic maps are everywhere. Hundreds

[1]Any lie that occurs in this chapter is clearly indicated by a footnote marker associated with the (truthful) text, as seen in this footnote and many of the notes following.



of thousands of J. R. R. Tolkien fans have merged their shrines into a single representation of Middle Earth. Closer to home, it is possible to buy tomatoes via cell phone.

Appetizer

Topic map documents are very simple: there are only 19 XTM elements. So, when you finish this section, you will know 50 percent of what you need to know to start topic mapping. In this section, you'll learn about the following elements (listed alphabetically):

```
<baseName>
<baseNameString>
<occurrence>
<resourceRef>
<scope>
<subjectIdentity>
<subjectIndicatorRef>
<topic>
<topicRef>
```

Introducing <topic>, <baseName>, <scope>, <baseNameString>, and <occurrence>

Let's create a topic element and give it an ID in case we need to identify it later:

```
<topic id="myTomato">
```

(I lied about what you need to know about XML. XML elements also have attributes, like the -id- attribute in the example above.)

> **PITFALL:** An ID must be a string that is unique within the topic map document. It doesn't have to make sense, like *myTomato*; it just has to be unique.[2] In most examples, IDs are a pitfall because for teaching and readability we make the ID strings in tutorials and on whiteboards look like names that have a deeper meaning than mere uniqueness. They don't.

Since we want to talk about tomatoes, let's type in a base name[3] for our topic that reflects our intent: "tomato." Since we don't want these potentially shady characters wandering round the landscape,[4] we wrap them up in a <baseNameString> element like so:

```
<topic id="myTomato">
  <baseName>
    <baseNameString>tomato</baseNameString>
  </baseName>
</topic>
```

Just in case anyone doubts that what we're talking about really is a tomato, let's supply a little more information about the topic by adding an image of a tomato, suitably ripened for site display.

```
<topic id="myTomato">
  <baseName>
    <baseNameString>tomato</baseNameString>
  </baseName>
  <occurrence>
    <resourceRef xlink:href="tomato.gif"/>
  </occurrence>
</topic>
```

---

[2]Another lie. An ID does have to be unique (within an XTM document), but also it can't start with a number, and if you could use a character to simulate profanity in a comic strip or for punctuation, you probably can't use it in an ID. For more information see the XML listing in Resources near the end of this chapter.

[3]You'll see what the base name is the basis of when we talk about variants later in this chapter. Why didn't XTM just call the element *name*? Because that word is all mixed up with PHPT.

[4]This is a lie. We're just avoiding "mixed content." See the XML listing in Resources near the end of this chapter.

You can think of the `-xlink:href-` attribute as working just like the `-href-` attribute of an HTML `<A>` element: it points to a resource you want to get, in this case, the GIF file that is the occurrence of the tomato topic. That `xlink:` prefix means that the attribute also conforms to the XLink World Wide Web Consortium (W3C) specification (see the Resources section).

In a topic map, like any map, *the map is not the territory.*[5] Here the topic whose ID is *myTomato* is part of the map; the resource in the file `tomato.gif` is part of the territory. For this reason, topic maps are sometimes called *information overlays.* They are superimposed over resources without changing them, just as a road map is an information overlay for the road you are driving—not because it is draped over the steering wheel of your car but because the connections that images of roads make between images of cities on the map are useful when driving to real cities over real roads.

But wait a minute. Ultimately, in our topic map future I want to use a topic map to order my tomatoes from My Tomato Purveyor (MTP), Inc., using my cell phone. So I need to be 100 percent certain that when my cell phone sends "tomato," MTP's server knows I mean "tomato." The base name string "tomato" works fine for humans, but maybe machines need some help. I may need to say what the *subject* of my topic is more precisely.

Introducing `<subjectIdentity>`

In this section, you'll learn about the following elements (again listed in alphabetical order) and how to choose between them:

```
<resourceRef>
<subjectIdentity>
<subjectIndicatorRef>
<topicRef>
```

Let's give the "tomato" topic an identity that both machines and humans can understand. I know, MTP knows, and the computer can "understand" that the USDA specification for the kind of tomato I want lives at the following URI: *http://www.fed.gov/usda/doc/tomatogr.htm#gradeA.* Thus I can express our mutually agreed-upon knowledge and declare the subject of our topic in the following way.

```
<topic id="myTomato">
  <subjectIdentity>
 <subjectIndicatorRef xlink:href="http://www.fed.gov/usda/doc/
    tomatogr.htm#gradeA"/>
```

[5]This is a lie. In a topic map, sometimes the map *is* the territory, since a topic map (being electronic) can be an occurrence of a topic, just like our tomato GIF file. Since this feature (and it is a feature) of topic maps is littered with PHPT we, as beginners, will avoid it.

```
  </subjectIdentity>
  <baseName>
    <baseNameString>tomato</baseNameString>
  </baseName>
  <occurrence>
    <resourceRef xlink:href="tomato.gif"/>
  </occurrence>
</topic>
```

It turns out that all these years, although without knowing it, the USDA and governments in general have been creating rich sets of resources for topic mappers to point to when they want to agree that they are talking about the same subject, like grade A tomatoes. These resources are called *Published Subject Indicators* (PSIs) and are discussed further in Chapter 5.

Of course, the Canadian government has been grading tomatoes too, and to tell MTP that it's all one to me whether I get Canadian tomatoes or American tomatoes, I would add a second `<subjectIndicatorRef>` pointing to the Canadian PSI.

> **PITFALL:** If I used the subject identity for USDA grade A tomatoes for a topic with the `<baseName>` "potato," that would really confuse the humans, the machines, and/or both. So I won't do that.

We also could have defined the subject identity of our topic using `<resourceRef>`, as shown below.

```
<subjectIdentity>
  <resourceRef xlink:href="tomato.gif"/>
</subjectIdentity>
```

We and MTP would then agree that the file `tomato.gif` unambiguously specified our subject, instead of just being information relevant to it, which is what an occurrence is. We could define our subject this way, but we would be wrong to do so in this situation. Why? A resource specified with `<subjectIdentity>` *indicates* a subject (in our case, tomato). But a resource specified with `<resourceRef>` *constitutes* the subject, *is* the subject (in this case, the exact bits and bytes that make up that single image at the address given in the `<resourceRef>` element's `-xlink:href-` attribute value). If we were graphic artists, we might care about that. As cooks, or people who claim to be cooks, we don't.

Finally, we could have specified the subject identity of our topic using `<topicRef>`, as shown below.

```
<subjectIdentity>
  <topicRef xlink:href="#anotherTomato"/>
</subjectIdentity>
```

The <topicRef> element points to the <topic> element that in turn has a subject. (The <topicRef> element must point to a <topic> element.) We and MTP would then agree that the <topic> element with the ID *anotherTomato* specified our subject. We will see one reason we might want to point at a <topic> element that specifies our subject identity, rather than pointing at the subject directly, in the next section.

Introducing <scope>

Unlike an ID, a <baseName> might be required to make sense to at least some humans; unfortunately the <baseName> we have chosen makes sense only in English. Since cuisine is by definition French, let's give our topic a second name in that language, where FR stands for "French" and EN stands for "English."

```
<topic id="myTomato">
  <baseName>
    <scope>
      <topicRef xlink:href="#EN"/>
    </scope>
    <baseNameString>tomato</baseNameString>
  </baseName>
  <baseName>
    <scope>
      <topicRef xlink:href="#FR"/>
    </scope>
    <baseNameString>tomate</baseNameString>
  </baseName>
  <baseName>
    <baseNameString>tomato</baseNameString>
  </baseName>
  . . .
</topic>
```

Here we use <scope> to turn <baseName> elements on and off. *Tomate*, for example, will be the <baseName> data for the <topic> element from the point of view of humans who prefer to speak French. But what about the third <baseName>, the one with no <scope>? This is the default <baseName>—the one that is always on.[6] This is the <baseName> that a human who speaks neither English nor French should see.

Now, why did we use <topicRef> to make our first two <scope> elements? Let's set up the <topic> elements to point at and use PSIs to express their subject identities.

[6]More lies. First, some cuisine is Chinese. Second, in the jargon, "topic characteristics" are "valid" within scopes. Scopes don't really turn anything on or off—applications do that. The scope that is "always on" is called the *unconstrained scope*.

```
<topic id="EN">
  <subjectIdentity>
    <subjectIndicatorRef
    xlink:href="http://www.topicmaps.org/xtm/1.0/language.xtm#en"/>
  </subjectIdentity>
</topic>
<topic id="FR">
  <subjectIdentity>
    <subjectIndicatorRef
    xlink:href="http://www.topicmaps.org/xtm/1.0/language.xtm#fr"/>
  </subjectIdentity>
</topic>
```

Using this setup has several advantages. First, the topic map is easier to read (and to write). `EN` is a lot shorter than `http://www.topicmaps.org/xtm/1.0/language.xtm#en`.

Second, the topic map is easier to maintain. The subject indicated by the topic `EN` turns out to be one of the two-letter ISO codes for human languages. (ISO, the International Organization for Standardization, has, like governments, been publishing PSIs without knowing it was doing so for many years.) Better yet, it turns out that this ISO language list has been translated into a topic map and is available on the TopicMaps.org site. (Check the value of the `-href-` attribute.) Best of all, suppose that we discover that there is a better listing of language code PSIs somewhere (see *http://www.oasis-open.org/committees/tm-pubsubj/*), and MTP and we agree to use it. In that case, we need only update the two `-href-` attributes in the subject indicators of the `<topic>` elements `EN` and `FR`, and in the machine, all the `<topicRef>` elements that point to them will update too.

Main Course

In this section, you'll learn about the following elements:

```
<association>
<instanceOf>
<member>
<roleSpec>
```

Four more elements—so when you finish this section, you will know 70 percent of what you need to know to start topic mapping.

Of course, a tomato is no good in isolation. Well, it is, but we really care about tomatoes when they are associated with other things—in our case, with caramel—using menus and recipes.

Introducing <association>, <member>, and <roleSpec>

First, we'll set up the association between the tomato and the dish. Let's make a topic for a dish.[7]

```
<topic id="myConfite">
  <baseNameString>
    tomate confite farcie aux douze saveurs
  </baseNameString>
</topic>
```

Now let's associate our tomato and this dish by making them members of an <association>, using <topicRef> to define the <member> elements.[8]

```
<association id="tomate_confite_association">
  <member>
    <topicRef xlink:href="#myTomato"/>
  </member>
  <member>
    <topicRef xlink:href="#myConfite"/>
  </member>
</association>
```

That there is an association (some association) between *myConfite* and *myTomato* is not very informative, however. We need to explain what roles the two topics are playing in the association. So let's make some topics for roles.

```
<topic id="anIngredient">
  <baseName>
    <baseNameString>an ingredient</baseNameString>
  </baseName>
</topic>
```

[7] This dish is an occurrence of a subject whose topic is named "the crisis in French cooking" by Adam Gopnik in his book *Paris to the Moon* (Random House, 2000). In fact, its chef conceives of the dish as a proof by example (achieved with immense labor) that the topic, *tomato*, is properly associated with a dish in the scope of the dessert phase of a meal, since the tomato is an instance of the class *fruit* and not an instance of the class *vegetable*.

[8] Inside <member> we could also have used <resourceRef>, for something that *is* a subject. We could also have *indicated* a subject with <subjectIndicatorRef>—even used <subjectIndicatorRef xlink:href="#myTomato"/>, which is equivalent to <topicRef xlink:href="#myTomato"/>. However, since we are pointing at something that we expect to create and manage as a <topic> in our topic map, it makes sense here to use <topicRef>.

(To keep the examples short, we'll leave the <baseName> elements out from now on. This is perfectly OK topic mapping—it's up to you whether your topics have <baseName> elements or not. In fact, the only thing that you must do with a <topic> element is give it an ID. We'll also tumble into the pitfall of giving elements IDs that look like names, purely for the sake of exposition.)

Here's another topic for another role.

```
<topic id="aDish"/>
```

Now we add the roles to our association using <roleSpec>.

```
<association id="tomate_confite_association">
  <member>
    <roleSpec>
      <topicRef xlink:href="#anIngredient"/>
    </roleSpec>
    <topicRef xlink:href="#myTomato"/>
  </member>
  <member>
    <roleSpec>
      <topicRef xlink:href="#aDish"/>
    </roleSpec>
    <topicRef xlink:href="#myConfite"/>
  </member>
</association>
```

Now we can interchange the knowledge that in this association, the tomato plays the ingredient role, and the confite plays the dish role.

Introducing <instanceOf>

Now we can distinguish the ingredient from the dish. But what about the association itself? What type of association is it? Let's make another topic.

```
<topic id="ingredient_of"/>
```

(Of course, in a real topic map, we could add a lot of detail to this topic—a PSI, one or more base names, and so on. Here, for purposes of exposition, we suggest all that robustness with the ID.)

Now let's revise our existing <association> element to say that the association tomate_confite_association is an association of the type ingredient_of.

```
<association id="tomate_confite_association">
  <instanceOf>
    <topicRef xlink:href="#ingredient_of"/>
  </instanceOf>
  <member>
    <roleSpec>
      <topicRef xlink:href="#anIngredient"/>
    </roleSpec>
    <topicRef xlink:href="#myTomato"/>
  </member>
  <member>
    <roleSpec>
      <topicRef xlink:href="#aDish"/>
    </roleSpec>
    <topicRef xlink:href="#myConfite"/>
  </member>
</association>
```

And in the same way, we can add other associations of the same type to our topic map.

```
<association id="caramels_confite">
  <instanceOf>
    <topicRef xlink:href="#ingredient_of"/>
  </instanceOf>
  <member>
    <roleSpec>
      <topicRef xlink:href="#anIngredient"/>
    </roleSpec>
    <topicRef xlink:href="#myCaramel"/>
  </member>
  <member>
    <roleSpec>
      <topicRef xlink:href="#aDish"/>
    </roleSpec>
    <topicRef xlink:href="#myConfite"/>
  </member>
</association>
```

Of course, for this markup we also have to add a new topic:

```
<topic id="myCaramel"/>
```

Caramel is one of the ingredients of the confite because the confite is in fact a dessert.

Now I can ask the topic map (using, in our topic map future, topic map software that understands the future topic map query language) for all the dishes that have tomatoes as ingredients, so I can consolidate my order to MTP when I finally flip open my cell phone and call them.

Back to the word *type*. This is one of those words that (among computer programmers at least) tends to generate the sort of heated PHPT-driven discussion I promised to avoid. We (and also computer programmers) say that "a tomato is a type of fruit," "2 is a type of number," "a lion is a type of animal," and so on. That means that fruits, numbers, and animals are all classes, and tomatoes, 2, and lions are all instances of each class.[9] Similarly, apples, 3, and aardvarks are also instances of the same three classes. Since we are in the topic map world, we make our classes with topics, but the same relationship between instance and class still holds.

Topic maps are fun to write, so once we get started making associations it's hard to stop. Let's make a menu.

```
<association id="entree_dessert">
  <instanceOf>
    <topicRef xlink:href="#menu"/>
  </instanceOf>
  <member>
    <roleSpec>
      <topicRef xlink:href="#dessert"/>
    </roleSpec>
    <topicRef xlink:href="#myConfite"/>
  </member>
  <member>
    <roleSpec>
      <topicRef xlink:href="#entrees"/>
    </roleSpec>
    <topicRef xlink:href="#myFoieGras"/>
  </member>
</association>
```

We also—as you may have predicted—have to make the following elements in our map.

```
  <topic id="menu"/>
  <topic id="dessert"/>
  <topic id="entrees"/>
  <topic id="myFoieGras"/>
```

And while we're thinking of typing, let's go back and type our confite, too.

```
<topic id="myConfite">
  <instanceOf>
    <topicRef xlink:href="#dessert"/>
```

[9] Typing is important for writing correct programs. Let's suppose that we feed a program an instance of class mineral (an instance whose type is mineral) when our program expects an instance of class animal (an instance whose type is animal)—a stone instead of a lion. The program should complain, and the programmer should figure out whether an instance of animal or mineral is required.

```
  </instanceOf>
  <baseName>
    <baseNameString>
      tomate confite farcie aux douze saveurs
    </baseNameString>
  </baseName>
</topic>
```

This way, if we want to ask our topic map for all the desserts, we'll get them. Better yet, since we know that there is a role on the menu called *dessert*, and we know that our confite is an instance of the class *dessert*, we can make sure that topics are playing sensible roles in our associations. The example below would not be sensible.

```
<member>
  <roleSpec>
    <topicRef xlink:href="#dessert"/>
  </roleSpec>
  <topicRef xlink:href="#myFoieGras"/>
</member>
```

> **PITFALL:** It's fair to say that in later versions of the topic map standard, there will be more sophisticated ways to say how members will play roles in associations. That is why I used the not-exactly-rigorous word *sensible* in the paragraph above.

Dessert

In this section, you'll learn about the following element:

`<mergeMap>`

This is only one more element, number 14 out of the 19 total elements, but it's a very important one. So let's say that when you finish this section, you will know 85 percent of what you need to know to start topic mapping.

The `<mergeMap>` element makes two or more topic maps into one topic map. In our topic map future, merging has allowed those Tolkien fans to merge their individual topic maps into a single giant shrine, with base names in English, Elvish, and so on.

Here, working on a smaller scale, we just need two topic maps to merge. Through a sophisticated analytical process we've made up another topic map that has the tomato topic in it—the actual recipe for the *tomate confite farcie aux douze saveurs*. Well . . . one step in the recipe, anyhow. The idea is that a step in a recipe is an association with the

following members: an ingredient, an amount (of that ingredient), and a process. So, "take a case of tomatoes and peel them." Here is the topic map.

```
<topicMap>
  <association id="peel_case_tomatoes">
    <instanceOf>
      <topic id="classTopic" xlink:href="#step"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink:href="#anotherIngredientTopic"/>
      </roleSpec>
      <topicRef xlink:href="#anotherTomatoTopic"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink:href="#anAmount"/>
      </roleSpec>
      <topicRef xlink:href="#case"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink:href="#aProcess"/>
      </roleSpec>
      <topicRef xlink:href="#peel"/>
    </member>
  </association>
  <topic id="anotherTomatoTopic">
    <subjectIdentity>
      <subjectIndicatorRef
           xlink:href="www.fed.goc/usda/doc/tomatogr.htm#gradeA"/>
    </subjectIdentity>
      <baseName>
        <scope>
          <topicRef xlink:href="#IT"/>
        </scope>
        <baseNameString>pomodoro</baseNameString>
      </baseName>
  </topic>
  <topic id="anotherIngredientTopic"/>
    <baseName>
      <baseNameString>an ingredient</baseNameString>
    </baseName>
  </topic>
  <topic id="anAmount"/>
  <topic id="step"/>
  <topic id="case"/>
  <topic id="aProcess"/>
  <topic id="peel"/>
  <topic id="IT">
    <subjectIdentity>
```

```
      <subjectIndicatorRef
      xlink:href="http://www.topicmaps.org/xtm/1.0/language.xtm#"/>
    </subjectIdentity>
  </topic>
</topicMap>
```

Here, for reference, are two topics from our original topic map. Compare them with the topics with IDs *anotherTomatoTopic* and *anotherIngredientTopic* in the topic map immediately above.

```
<topic id="myTomato">
   <subjectIndicatorRef
   xlink:href="www.fed.gov/usda/doc/tomatogr.htm#gradeA"/>
  </subjectIdentity>
<baseName>
    <scope>
      <topicRef xlink:href="#EN"/>
    </scope>
    <baseNameString>tomato</baseNameString>
  </baseName>
  <baseName>
    <scope>
      <topicRef xlink:href="#FR"/>
    </scope>
    <baseNameString>tomate</baseNameString>
  </baseName>
  <baseName>
    <baseNameString>tomato</baseNameString>
  </baseName>
</topic>
<topic id="anIngredient"/>
  <baseName>
    <baseNameString>an ingredient</baseNameString>
  </baseName>
</topic>
```

Watch closely! We now merge our topic maps, and somewhere in the computer the following magic happens.

- All topics with the same name in the same scope are merged (a *name-based merge*).
- All topics with the same subject identity are merged (a *subject-based merge*).

("Two things that are equal to the same thing are equal to each other."[10])

[10]Yet another lie. The merging requirements are more sophisticated and rigorous than this. See the XTM 1.0 specification at *http://www.topicmaps.org/xtmtm*.

In our two topic maps, two topics are merged. First, the topic with ID *anIngredient* is merged with the topic with ID *anotherIngredientTopic*. Why? They both have the same base name ("an ingredient") in the same scope (the unconstrained scope). Second, the topic with ID *myTomato* is merged with the topic with ID *anotherTomatoTopic*. Why? They both share a PSI, the USDA definition of a grade A tomato.

Now, what is the benefit of this merge? When two topics are merged into a single topic, that topic has all the topic characteristics of both topics—base names, occurrences, and roles played in associations—with any duplicate characteristics thrown away. The characteristics of our merged tomato topic become those listed below.

Base names:

- tomato in English (menu map)
- tomate in French (menu map)
- tomato in the unconstrained scope (both maps)
- pomodoro in Italian (recipe map)

Occurrences:

- tomato.gif, a resource (menu map)

Roles played in associations:

- ingredient in the *tomate_confite_association* association (menu map)
- ingredient in the *peel_case_tomatoes* association (recipe map)

Thus, by merging the two topic maps, we can get the quantity of tomatoes we need and the recipe we need them for. Using my cell phone in the topic map future, I merge our merged topic map with a third map (the price list for grade A USDA tomatoes available from MTP) and place an order. Shortly, our tomato salesman (Joe) merges the order with his sales projection topic map and smiles—at knowledge interchange in action.

PITFALL: The interaction between the two merging rules explains why it's a bad idea for a topic with the base name potato to be given a subject identity of *tomato* (where other topics with that identity have the base name tomato). If we did this, under a name-based merge, topics named tomato and potato in the English scope would not be treated as one topic, but under a subject-based merge, they would! Thus, for example, all the recipes associated with tomato as an ingredient will also be associated with potato as an ingredient (conflating New York– and New England–style clam chowder, for example, not to mention Italian cuisine and vodka manufacture).

Is this a bug or is it a feature? It's a feature. If you choose to give topics that others think have different subjects the same name, it makes sense to merge them. Why give the same name to two different things? And if others think that topics to which you give different names have the same subjects, it makes sense to merge them. Why give two different names to the same thing?

If, in the merge process, your tomatoes get mixed with your potatoes, there are tricks to detect the situation using the children of `<mergeMap>`. If you get in trouble like this, you aren't a beginner anymore, so these tricks are out of the scope of this chapter. See the XTM 1.0 specification for more information.

Brandy, Cigars

In this section, you'll learn about the following elements:

```
<parameters>
<resourceData>
<variant>
<variantName>
```

When you finish this section, you will have covered 18 of the 19 topic map elements, so you will know almost everything you need to know to start topic mapping.

Introducing `<variant>`, `<variantName>`, and `<parameters>`

Some of our clients want to be able to display our menus on their Palm Pilots, and Joe at MTP wants to use his cell phone too. So, we pick a very short name suitable for use in wireless activities.

```
<topic id="myTomato">
  <baseName>
    <baseNameString>
      tomato
    </baseNameString>
    <variant>
      <variantName>
        <resourceData>
          TMT
        </resourceData>
      </variantName>
```

```
      <parameters>
        <topicRef xlink:href="#cell_phone"/>
      </parameters>
    </variant>
  </baseName>
</topic>
```

Thus, a cell phone user would see the short `TMT` instead of the longer `tomato`. (Of course we also need to add `<topic id="cell_phone"/>`. A more sophisticated topic map would use PSIs for the Motorola and Nokia product lines and might have variant names appropriate to each line, but since this example is in our topic map future, we don't have values for the `-href-` attributes yet.)

This example shows that a variant is a variant of a base name and that a base name is called *base* because it has variants.

Introducing `<resourceData>`

Finally, sometimes we want to embed a little bit of territory right in the topic map document. For this purpose, we have `<resourceData>`, which can occur in the `<variantName>` and `<occurrence>` elements.

The `<resourceData>` element is just a shortcut for `<resourceRef>`. It would be foolish to have to create a file and a URI for every tiny piece of text in the whole topic map, so with `<resourceData>` we allow text to be entered into the topic map document directly.

Paying the Bill and Putting on Your Coat

The `<topicMap>` element is our nineteenth and final element. We'll wrap up our example by putting our first topic map inside a `<topicMap>` element; see Appendix A in this book. (Note especially the XML plumbing at lines 1 through 4: the `xml` and `DOCTYPE` lines, as well as the namespace declarations that are attributes of the `<topicMap>` element.)

Note also that it is possible to have topic map tags that don't contain anything at all. In fact, `<topicMap>`, `<topic>`, and `<subjectIdentity>` all have this characteristic.

Is this a bug or a feature? It's a feature; the XTM specification is designed for interchange. It is descriptive, not prescriptive. You, as author, may wish to practice better informational hygiene. However, there are scenarios in which an author might wish to create such esoterica as the following:

- A <topicMap> with no children
- A <topic> element with no subject identity, base name, or occurrences
- A <subjectIdentity> element that refers to nothing
- A <mergeMap> element with no children

One obvious reason is the authoring process—put in an empty topic, link to it with the ID, and plan to circle back and add the rest later. The more philosophical reason is that sometimes knowledge of the world is partial, and so the interchange syntax requires that as little as possible be known.

Now you have 100 percent of the knowledge required to start topic mapping.

Summary

Here is what you learned in this chapter in a classic bottom-up rather than top-down approach.

- Topic maps consist mainly of topics and associations, as you saw when we created topic maps that associate the tomato topic with recipes and menus.
- A topic map is an overlay on information resources, as you saw when we created an occurrence for the tomato topic.
- A topic is a stand-in, proxy, or surrogate for a subject, as you saw when we discussed PSIs.
- Topics have characteristics (names, occurrences, and roles played in associations), as you saw when we gave our topics base and variant names, created an occurrence, and gave our associated topics role specifications.
- The author controls the meaning of a topic map through topic characteristics and choices of subject, as you saw when we controlled the merging process through our choice of base names and subject identity.
- Scopes in topic maps define the validity of associations and allow fine-tuning of merge operations,[11] as you saw when we scoped the base names of topics for the human languages English, French, and Italian.

Topic maps are about agreement. Even though I say *tomato*, and you say *tomate* or even *pomodoro*, we don't have to call *anything* off. Topic maps allow us to say what we mean and mean what we say.

[11]A terminological inexactitude. In truth, scopes define namespaces, and they do so in a more flexible and powerful way than colonized syntax. Synonyms are permitted, for example. The claim has been made that this namespace system can lead the way to the federation of global knowledge.

Since you now know 100 percent of what you need to know to start topic mapping—start topic mapping!

Acknowledgments

I would like the thank the founding coeditors of the XTM specification, Michel Biezunski and Steven R. Newcomb, and the members of the XTM Authoring Group, for the privilege of editing the original drafts of the XTM 1.0 specification and coauthoring the XTM 1.0 DTD. Also, I would like to thank Jean Delahousse of Mondeca for clearing the physical and conceptual space for me to write this chapter. Finally, my thanks to the reviewers for their comments and suggestions.

Resources

To learn more about topic maps, here are three good entry points.

http://www.topicmaps.org/

http://www.topicmaps.net/

http://www.oasis-open.org/cover/topicMaps.html

And to show that topic maps are actual and not academic or theoretical, here are sites for topic map vendors and service providers, in no particular order.

http://www.infoloom.com/

http://www.mondeca.com/

http://www.ontopia.net/

http://www.empolis.com/

http://www.cogx.com/

http://www.semantext.com/

http://globalwisdom.org

http://www.etopicality.com

Other technical pieces related to topic maps are XML and XLink. XML is the W3C specification that says, among other things, that the tags we've discussed in this chapter are made from letters and angle brackets (`<tag>`) as opposed to being made, for

example, with curly braces and nonletter characters (`{_12^_}`). XLink provides the way to perform semantic linking in XML. For more information, see these Web sites.

XML: *http://www.w3.org/XML/#dev*

XLink: *http://www.w3.org/XML/Linking*

For a nontechnical introduction to XML, see the following resources.

http://www.webreview.com/2000/06_23/webauthors/06_23_00_1.shtml (parts 1 and 2)

http://www.webreview.com/2000/08_04/webauthors/08_04_00_4.shtml (parts 1 and 2)

The draft ISO Reference Model for topic maps (see Chapter 4) uses a graph for its data model. For perspective on graph theory, see Randall Ripert, "The Mathematical Structure of the World: The World as Graph," available online at *http://neologic.net/rd/Papers/Structure-J-20-Phil.html*.