Practical TMRAPping
A tutorial

Lars Marius Garshol
CTO, Ontopia
<larsga@ontopia.net>
Agenda

• Introduction
  – use cases
  – status

• Understanding TMRAP
  – the basics
  – the requests

• The HTTP binding
  – examples

• TM/XML
  – examples

• The HTTP binding
  – examples
What you will learn

- This tutorial assumes you already know
  - Topic Maps
  - tolog
  - XTM and LTM
  - HTTP and Web Services in general

- It will teach you
  - how TMRAP works conceptually
  - how to work with the returned XML data

- It will not teach you
  - how to do client programming
  - this will depend on your environment...
Introduction

Use cases

Status
The Vizigator

- The Vizigator uses TMRAP
  - the Vizlet runs in the browser (on the client)
  - a fragment of the topic map is downloaded from the server
  - the fragment is grown as needed
CMS integration

- TMRAP can be used to integrate CMSs that are not Java-based
  - events in the CMS trigger update requests to the TMRAP server
  - the end-user interface retrieves Topic Maps data via TMRAP
Building temporary topic maps

- The Amsterdam police has built a prototype of this
  - investigators can use TMRAP to extract related fragments from various systems
  - these are integrated into a single, temporary topic map that can be browsed to see the connections
Towards seamless knowledge

- As the number of portals multiplies, the amount of overlap increases...
- Take these three portals as an example:
  - forskning.no (Research Council web site aimed at young adults)
  - forbrukerportalen.no (Public site of the Norwegian Consumer Association)
  - matportalen.no (Biosecurity portal of the Department of Agriculture)
Genmodifisert mat

Sykdomsutbrudd av genmodifisert mais?
Professor Terje Traavik ved Genok har gått ut med foreløpige funn som antyper at et sykdomsutbrudd i en filippinsk landsby kanske kan kobles til genmodifisert mals. Saken har skapt furøre verden over.

10.03.2004

Britene sier ja til genmodifisert mais
Storbritannia har gitt grønt lys for å dyrke genmodifisert mais på britisk jord. Dette er en historisk avgjørelse, selv om britene stiller en rekke betingelser til dyrkingen.

08.03.2004

Advarer mot genmanipulerte organismer
Et panel av amerikanske forskere mener vi vet for lite om konsekvensene når vi slipper genmanipulerte organismer ut i naturen. Hå oppfordrer de til større forsiktighet.

25.02.2004

Ja eller nei til genmodifisert mat?
"To be or not to be" er ikke det høtest spørsmålet i Storbritannia for tiden. Det er snarere ja eller nei til genmodifisert mat. Etter lekkasje fra en regeringskomite har spekulasjonene tatt av i pressen.
Genetisk modificert mat

EU forbyr fortsatt gen-mat
18.11.2003

EU har foreløpig avvist en søknad om import av en gennemført matprodukt.

Internasjonal motstand mot USAs GMO-rettskak
25.07.2003

Forbrukerorganisasjonene i EU og USA har igangsatt et arbeid i henhold til den amerikanske regjeringen om å treffe saker mot EU om gennemførte organiser (GMO).

Rapport fra Roma:
02.07.2003

FN setter regler for gen-mat

Codex alimentarius

- Kan ikke garantere ren øko-mat
27.06.2003
Genetisk modificerte matvarer


Les mer om genmodifisering:

- Biologisk omdøpning
- Genetisk omdøpning
- Genmodifisering - nye muligheter?
- Genmodifisering - på vilt växter?
- Genmodifisering - på vilt växter?
- Genmodifisering - nederlag?
- Genmodifisering - forsikring og skyte

**Hva er genmodifisert mat?**
- Gener og genteknologi
- Genmodifisering - nye muligheter?
- Genmodifisering - på vilt växter?
- Genmodifisering - nederlag?
- Genmodifisering - forsikring og skyte

**Les mer om genmodifisering:**

- Biologisk omdøpning
- Genetisk omdøpning
- Genmodifisering - nye muligheter?
- Genmodifisering - på vilt växter?
- Genmodifisering - nederlag?
- Genmodifisering - forsikring og skyte
Three semantic portals – One common subject

→ one “virtual portal” with seamless navigation in all directions
Current status

• **TMRAP implemented in the OKS**
  – implementation not 100% complete
  – extended as needed
  – was partly extended for this tutorial (to be released in OKS 3.2)

• **Other applications built based on TMRAP**
  – Vizigator
  – TMBuilder (by Cerpus)
  – Amsterdam Police prototype
  – ...

• **No other implementations at the moment**
TMRAP

Basics

Understanding TM/XML

The requests
TMRAP basics

• Abstract interface
  – that is, independent of any particular technology
  – coarse-grained operations, to reduce network traffic

• Protocol bindings exist
  – plain HTTP binding
  – SOAP binding

• Supports many syntaxes
  – XTM 1.0
  – LTM
  – TM/XML
  – custom tolog result-set syntax
TM/XML

• Non-standard XML syntax for Topic Maps
  – defined by Ontopia (presented at TMRA’05)
  – implemented in the OKS

• XSLT-friendly
  – much easier to process with XSLT than XTM
  – can be understood by developers who do not understand Topic Maps
  – dynamic domain-specific syntaxes instead of generic syntax
  – predictable (can generate XML Schema from TM ontology)
get-topic

- Retrieves a single topic from the remote server
  - topic map may optionally be specified
  - syntax likewise

- Main use
  - to build client-side fragments into a bigger topic map
  - to present information about a topic on a different server
get-topic

• **Parameters**
  – **identifier**: a set of URIs (subject identifiers of wanted topic)
  – **subject**: a set of URIs (subject locators of wanted topic)
  – **item**: a set of URIs (item identifiers of wanted topic)
  – **topicmap**: identifier for topic map being queried
  – **syntax**: string identifying desired Topic Maps syntax in response
  – **view**: string identifying TM-Views view used to define fragment

• **Response**
  – topic map fragment representing topic in requested syntax
  – default is XTM fragment with all URI identifiers, names, occurrences, and associations
  – in default view types and scopes on these constructs are only identified by one `<Ref xlink:href="..."> XTM element`
  – the same goes for associated topics
## Syntax identifiers

- **XTM 1.0**  application/x-xtm
- **LTM**  text/x-ltm
- **AsTMa**  text/x-astma
- **TM/XML**  text/x-tmxml
- **tolog**  text/x-tolog
get-topic-page

- Returns link information about a topic
  - that is, where does the server present this topic
  - mainly useful for realizing the portal integration scenario
  - result information contains metadata about server setup
get-topic-page

• **Parameters**
  – **identifier**: a set of URIs (subject identifiers of wanted topic)
  – **subject**: a set of URIs (subject locators of wanted topic)
  – **item**: a set of URIs (item identifiers of wanted topic)
  – **topicmap**: identifier for topic map being queried
  – **syntax**: string identifying desired Topic Maps syntax in response

• **Response is a topic map fragment**
  
  [oks : tmrap:server = "OKS Samplers local installation"]
  [opera : tmrap:topicmap = "The Italian Opera Topic Map"]
  
  {opera, tmrap:handle, [[opera.xtm]]}
  tmrap:contained-in(oks : tmrap:container, opera : tmrap:containee)
  tmrap:contained-in(opera : tmrap:container, view : tmrap:containee)
  tmrap:contained-in(opera : tmrap:container, edit : tmrap:containee)
  [russia = "Russia" @"http://www.topicmaps.org/xtm/1.0/country.xtm#RU"]
get-tolog

• **Returns query results**
  – main use is to extract larger chunks of the topic map to the client for presentation
  – more flexible than get-topic
  – can achieve more with less network traffic
get-tolog

• **Parameters**
  - **tolog**: tolog query
  - **topicmap**: identifier for topic map being queried
  - **syntax**: string identifying desired syntax of response
  - **view**: string identifying TM-Views view used to define fragment

• **Response**
  - if syntax is “tolog”
    - an XML representation of the query result
    - useful if order of results matter
  - otherwise, a topic map fragment containing multiple topics is returned
    - as for get-topic
add-fragment

• **Adds information to topic map on the server**
  – does this by merging in a fragment

• **Parameters**
  – **fragment**: topic map fragment
  – **topicmap**: identifier for topic map being added to
  – **syntax**: string identifying syntax of request fragment

• **Result**
  – fragment imported into named topic map
**update-topic**

- **Can be used to update a topic**
  - add-fragment only adds information
  - update sets the topic to exactly the uploaded information

- **Parameters**
  - `topicmap`: the topic map to update
  - `fragment`: fragment containing the new topic
  - `syntax`: syntax of the uploaded fragment
  - `identifier`: a set of URIs (subject identifiers of wanted topic)
  - `subject`: a set of URIs (subject locators of wanted topic)
  - `item`: a set of URIs (item identifiers of wanted topic)

- **Update happens using TMSync**
  - to learn more about this, attend my talk tomorrow (1430-1500)
delete-topic

- Removes a topic from the server
- Parameters
  - **identifier**: a set of URIs (subject identifiers of wanted topic)
  - **subject**: a set of URIs (subject locators of wanted topic)
  - **item**: a set of URIs (item identifiers of wanted topic)
  - **topicmap**: identifier for topic map being queried
- Result
  - deletes the identified topic
    - includes all names, occurrences, and associations
HTTP binding

Basics

How to use
HTTP binding basics

• The mapping requires a base URL
  – e.g http://localhost:8080/tmrap/

• This is used to send requests
  – http://localhost:8080/tmrap/method?param1=value1&...
  – GET is used for requests that do not cause state changes
  – POST for requests that do

• Responses returned in response body
Exercise #1: Retrieve a topic

• Use the get-topic request to retrieve a topic from the server
  – base URL is http://localhost:8080/tmrap/
  – find the identifying URI in Omnigator
  – just print the retrieved fragment to get a look at it

• Note: you must escape the “#” character in URIs
  – otherwise it is interpreted as the anchor and not transmitted at all
  – escape sequence: %23

• Note: you must specify the topic map ID
  – otherwise results will only be returned from loaded topic maps
  – in other words: if the topic map isn’t loaded, you get no results
Solution #1 (in Python)

```python
import urllib

BASE = "http://localhost:8080/tmrap/tmrap/
psi = "http://www.topicmaps.org/xtm/1.0/country.xtm%23RU"

inf = urllib.urlopen(BASE + "get-topic?identifier=" + psi)
print inf.read()
inf.close()
```
Solution #1 (response)

<topicMap xmlns="http://www.topicmaps.org/xtm/1.0/"
  xmlns:xlink="http://www.w3.org/1999/xlink">
  <topic id="id458">
    <instanceOf>
      <subjectIndicatorRef xlink:href="http://psi.ontopia.net/geography/#country"/>
    </instanceOf>
    <subjectIdentity>
      <subjectIndicatorRef xlink:href="http://www.topicmaps.org/xtm/1.0/country.xtm#RU"/>
      <topicRef xlink:href="file:/.../WEB-INF/topicmaps/geography.xtmm#russia"/>
    </subjectIdentity>
    <baseName>
      <baseNameString>Russia</baseNameString>
    </baseName>
  </topic>
</topicMap>
Processing XTM with XSLT

• This is possible, but unpleasant
  – the main problem is that the XML is phrased in terms of Topic Maps, not in domain terms
  – this means that all the XPath will talk about “topic”, “association”, ... and not “person”, “works-for” etc

• The structure is also complicated
  – this makes queries complicated
  – for example, the XPath to traverse an association looks like this:

```
//xtm:association
    [xtm:member[xtm:roleSpec / xtm:topicRef / @xlink:href = '#employer']
        [xtm:topicRef / @xlink:href = concat('#', $company)]]
    [xtm:instanceOf / xtm:topicRef / @xlink:href = '#employed-by']
```
Learning TM/XML

How it works
General principles of TM/XML

• Fixed structure
  – document element is topic map
  – level below is topics
  – level below is properties of topics
  – no more levels

• Element types generated from the types of the objects
  – person topics get “person” elements, etc
  – PSIs turn into QNames (foo:bar)
  – IDs turn into normal element type names
Fragment structure

<topicmap ...>

<topic-type id="...">

<iso:topic-name>

<tm:value>...</tm:value>

</iso:topic-name>

<occurrence-type>...</occurrence-type>

<association-type role="role-type" topicref="..."

otherrole="role-type"/>

</geography:country>

</topicmap>
Exercise #2: Retrieve a topic in TM/XML

- Use the get-topic request to retrieve a topic from the server
  - base URL is http://localhost:8080/tmrap/
  - find the identifying URI in Omnigator
  - just print the retrieved fragment to get a look at it
  - syntax identifier for TM/XML is "text/x-tmxmxml"
Solution #2 (in Python)

```python
import urllib

BASE = "http://localhost:8080/tmrap/tmrap/"
psi = "http://www.topicmaps.org/xtm/1.0/country.xtm%23RU"
syntax = "&syntax=text/x-tmxml"

request = BASE + "get-topic?identifier=" + psi + syntax
inf = urllib.urlopen(request)
print inf.read()
inf.close()
```
Solution #2 (response)

<topicmap ...

<geography:country id="id458">
<tm:identifier>http://www.topicmaps.org/xtm/...</tm:identifier>
<iso:topic-name>
<tm:value>Russia</tm:value>
</iso:topic-name>
<geography:located-in scope="psi.ontopia.net:geography" role="geography:container" topicref="id345" otherrole="geography:containee"></geography:located-in>
<!-- ... -->
</geography:country>
</topicmap>
Exercise #3: Presenting topics

• Make an XSLT stylesheet that presents composer topics
• Use TMRAP to get TM/XML fragments to try it out with
TMRAP in more depth

Understanding fragments
Using more requests
Understanding fragments

- The default fragments are defined as “stub”s
  - this means you get the requested topic only
  - all referenced topics are stubs
  - this means we only have their identity; nothing more

- This is not what you want for displaying a topic
  - in this case you want the names of all associated topics as well
  - (otherwise you can’t display the associations)
  - to do this, use “&view=names”
Exercise #4: Displaying topics properly

• Improve the composer stylesheet so it also displays the names of associated topics
How to build a web site from this

• Write one XSLT stylesheet per topic type
• Write a script that for each topic types
  – does a query to extract the identifiers of all topics of each type,
  – then retrieve the fragment for each topic, and
  – run the XSLT stylesheet on it
• Ensure that there is linking logic in your stylesheet that matches the file names that your script produces
How you would do it for real

• In a real setting there would be an application hosted on a server
  – in practice this could be ASP.NET, Zope (Python), PHP, ...

• Pages would be built dynamically
  – fragments would be loaded from the topic map and presented dynamically
  – caching could happen in front of the application, in front of the TMRAP server, or both

• So many of the concerns on the previous slide would not apply
  – however, to avoid having to get into the detail of a client platform we choose a simplified approach here
get-tolog

- As described earlier, this request has two modes
  - fragment mode, which produces a fragment (in XTM or TM/XML), and
  - tolog mode, which produces an XML representation of the result set
    - tolog mode is the default

- In fragment mode
  - only single-column queries are allowed,
  - every value must be a topic, and
  - the result is a fragment containing all the queried topics

- In tolog mode
  - there are no restrictions on queries, and
  - a special “tolog” XML syntax is used to represent the result set
The “tolog” syntax

```xml
<result ...>
  <head>
    <column>COMPOSER</column>
    <column>OPERA</column>
  </head>
  <body>
    <row>
      <value>
        <x:subjectIndicatorRef
      </value>
      <value>28</value>
    </row>
  </body>
</result>
```
Exercise #5: Create a composers page

• Should list all composers by name
  – get the necessary data using get-tolog
  – try out both fragment mode and tolog mode
Performance issues

- Neither of the two approaches we’ve found so far will actually allow us to produce the composer page + individual pages efficiently
  - with fragments: we only get the composer topics, not the names of associated topics
  - with tolog: it’s possible to get the data here, but very awkward, as we have to explicitly ask for everything
  - we need better control over what is returned

- The solution is, once again, views
Views supported by get-tolog

• In tolog mode
  – stub the default, gives just identities of returned topics
  – name gives just names of returned topics
  – full produces full fragments, just like in fragment mode
  – full-name also full fragments, but now including a name for associated topics

• This allows us to return exactly the XML we need, in one operation
Exercise #6: Create a composer site

- **With a single TMRAP request**
  - create an overview page of composers, plus
  - one page for each composer

- **This can be done with ... what ...?**
Updating the topic map

• **add-fragment**
  – always adds to the topic map (no existing data removed)

• **update-topic**
  – updates the chosen topic to only have the uploaded data

• **delete-topic**
  – removes a topic from the topic map
Exercise #7: Adding a new composer

- Use TMRAP to add a new composer to the topic map
  - this is easiest to do with opera.hytm, because you can use IDs
Exercise #8: Correcting the composer

- Let’s assume we got some information about the composer wrong, and correct it
Exercise #9: Covering our tracks

- And now, let’s remove this composer to clean up the topic map
Conclusion

More information
Conclusion

- **TMRAP is a versatile generic web service interface**
  - not bound to the OKS, but only implemented in the OKS
  - has a dependency on tolog at the moment, but could support TMQL

- **Allows many different kinds of applications to be built**
  - pure presentation applications as well as updates

- **Takes Topic Maps away from simple, monolithic applications**
More information

• Original TMRAP vision
  – Seamless Knowledge: Spontaneous Knowledge Federation using Topic Maps (Pepper & Garshol)
  – http://www.ontopia.net/topicmaps/materials/Seamless%20Knowledge%20with%20TMRAP.ppt

• TMRA’05 paper
  – slides: http://www.informatik.uni-leipzig.de/~tmra05/PRES/LMGa.pdf

• OKS documentation
  – Developer’s Guide included