Connectivity, webservices and XML data bases

Code: conn-ws

Author and version

- Daniel K. Schneider
- Email: Daniel.Schneider@tecfa.unige.ch
- Version: 0.2 (modified 11/12/07 by DKS)

Prerequisites

- Some knowledge about HTTP
- Some knowledge of XML

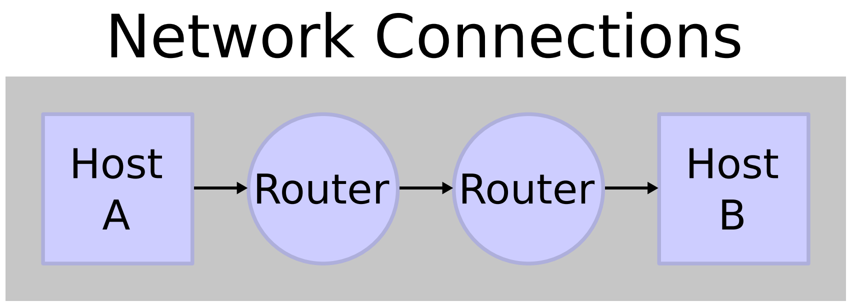
Objectives

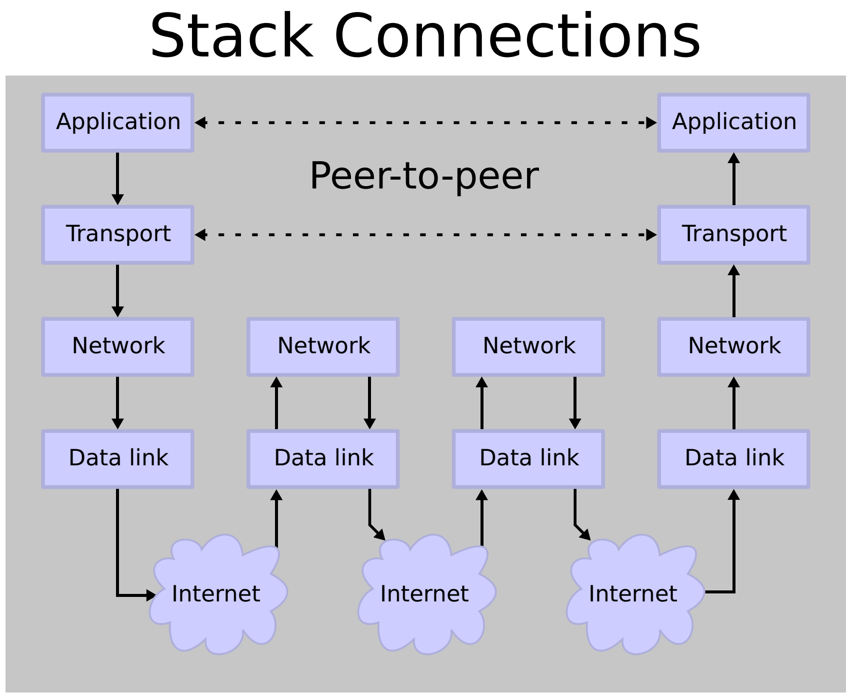
- Understand some networking technology at the services level
- Disclaimer: This is informal extra information that will help you understand where database-related services come in ... These slides have ***DRAFT*** status !!

1. Table of Contents

2. Introduction

- Internet refers to a complex stack of technologies (starting with physical layers up to application layers such as the world-wide web). An application talking to another application must use a transport protocol that relies on other lower level networking technology.





3. Networking history

1960

Licklider (1960) wrote "Man-Computer Symbiosis": “Man-computer symbiosis is an expected development in cooperative interaction between men and electronic computers. It will involve very close coupling between the human and the electronic members of the partnership. The main aims are 1) to let computers facilitate formulative thinking as they now facilitate the solution of formulated problems, and 2) to enable men and computers to cooperate in making decisions and controlling complex situations without inflexible dependence on predetermined programs.”

1962 Packet switching

The basis of all modern computer networks. A message sent from A to B can be broken down into packets. Packets travel from point A to point B and then are reassembled. The basic structure of a packet looks like:

```
sender - receiver - message
```

1969 Arpanet and telnet

In December 1969, the first version of Arpanet (Internet) went online. It connected four computers from four universities (UCLA, Stanford Research Institute, UCSB, and the University of Utah). The project leader was Bob Kahn from BBN (Cambridge,MA).

Telnet (TELecommunication NETwork): Remote connection to another computer over a internet-like network.

1971 File transfer protocol FTP

Early version of FTP, revised in 1980 and 1985. Still popular (but consider using SFTP or SCP instead, since FTP is inherently insecure).

1972 Email

Ray Tomlinson (BBN) created the first e-mail program.

1973 Birth of Transmission Control Protocol (TCP):

According to Vint Cerf's FAQ: " During 1973, we developed the concepts underlying the Internet and prepared a preliminary paper in September of that year that we presented to the International Network Working Group (INWG). In December 1974 the first full draft of TCP was produced."

1978 TCP/IP

TCP/IP, the main technical pillar of Internet emerged in mid-late 1978 in nearly final form and was finalized in 1991. "The Internet protocol suite is the set of communications protocols that implements the protocol stack on which the Internet and many commercial networks run. It is part of the TCP/IP protocol suite, which is named after two of the most important protocols in it: the Transmission Control Protocol (TCP) and the Internet Protocol (IP), which were also the first two networking protocols defined.

1989 Archie

Peter Deutsch et al. (McGill University in Montreal) created Archie, an index machine for public ftp sites. Something like the grandfather of Google. This service allowed people to find software and texts.

1991 Gopher

The University of Minnesota developer gopher named after a mascot but also means "go fer". Gopher was a user-friendly server that allowed administrators to build menus to access local or remote files and services (e.g. phone directories, library interfaces).

1992 The Web

im Berners-Lee et al. invented the World Wide Web with its two main components: HTTP and HTML.

1998 XML

Soon after, the first XML-based networking application standards emerged (e.g. SOAP and XML-RPC).

4. The network protocol stack

4.1 The five-layer TCP/IP model

url: **http://en.wikipedia.org/wiki/TCP/IP_model**

The TCP/IP model or Internet reference model is a layered abstract description for communications and computer network protocol design. Created in the 1970s by DARPA for use in developing the Internet's protocols. Still the most widely adopted view.

The five-layer TCP/IP model
5. Application layer
DHCP · DNS · FTP · Gopher · HTTP · IMAP4 · IRC · NNTP · XMPP · POP3 · SIP · SMTP · SNMP · SSH · TELNET · RPC · RTCP · RTSP · TLS · SDP · SOAP · GTP · STUN · NTP · (more)
4. Transport layer
TCP · UDP · DCCP · SCTP · RTP · RSVP · IGMP · (more)
3. Network/Internet layer
IP (IPv4 · IPv6) · OSPF · IS-IS · BGP · IPsec · ARP · RARP · RIP · ICMP · ICMPv6 · (more)
2. Data link layer
802.11 · 802.16 · Wi-Fi · WiMAX · ATM · DTM · Token ring · Ethernet · FDDI · Frame Relay · GPRS · EVDO · HSPA · HDLC · PPP · PPTP · L2TP · ISDN · (more)
1. Physical layer
Ethernet physical layer · Modems · PLC · SONET/SDH · G.709 · Optical fiber · Coaxial cable · Twisted pair · (more)

... if you wish to know more, follow up the links.

4.2 The 7-layer OSI model

1979. OSI has two major components: an abstract model of networking (the Basic Reference Model, or seven-layer model) and a set of concrete protocols.

OSI Model			
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation and encryption
		5. Session	Interhost communication
	Segment	4. Transport	End-to-end connections and reliability (TCP)
Media layers	Packet/Datagram	3. Network	Path determination and logical addressing (IP)
	Frame	2. Data link	Physical addressing (MAC & LLC)
	Bit	1. Physical	Media, signal and binary transmission

Session

- controls the dialogues/connections (sessions) between computer

Presentation

- syntax and semantics, e.g. Abstract Syntax Notation One (ASN.1) or XML

Application

- things like FTP, HTTP etc. (i.e. services used by application programs)

5. Application layers and services

- Usually rely on transport procols like TCP and UDP
- Common servers have specific ports assigned to them (HTTP has port 80; FTP has port 21; etc.)

Important basic Internet services

- File Transfer e.g. FTP (TCP/IP Protocol)
- Mail Transfer.g. SMTP/POP3/IMAP (TCP/IP protocols)
- WWW, e.g. HTTP (TCP/IP protocol), an extension of the telnet protocol

Web services (more recent services mostly built on top of HTTP)

- There are a variety of specifications associated with web services. These specifications are in varying degrees of maturity and are maintained or supported by various standards bodies and entities. Specifications may complement, overlap, and compete with each other. (Wikipedia).

5.1 Note on end user applications

End user applications ***use*** the services of the application layer, but are not part of the application layer itself.

- File Transfer applications use e.g. FTP (TCP/IP Protocol)
- Mail Transfer clients use e.g. SMTP/POP3/IMAP (TCP/IP protocols)
- Web browsers using HTTP (TCP/IP protocol)

6. Database connectivity

Direct connections

- APIs that allow a client (e.g. a database tool or a scripting language to connect directly to a database).
- Examples
  - Open Database Connectivity (ODBC) specification offers a procedural API for using SQL queries to access data. An implementation of ODBC will contain one or more applications, a core ODBC library, and one or more "database drivers". The core independant core library is an "interpreter" between applications and database drivers, whereas the database drivers contain the DBMS-specific details.
  - Java Database Connectivity (JDBC) is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases. JDBC may interact with OBDC drivers.

  url: **http://en.wikipedia.org/wiki/Java_Database_Connectivity**
  - ADO.NET is a set of software components that can be used by programmers to access data and data services. It is a part of the base class library that is included with the Microsoft .NET Framework
  - Proprietary, e.g. the MySQL libraries in PHP. Note: Most professional PHP programmers use a database independant library such as ADODB.
  - XML:DB API (XAPI) is designed to enable a common access mechanism to XML databases. The API enables the construction of applications to store, retrieve, modify and query data that is stored in an XML database. These facilities are intended to enable the construction of applications for any XML database that claims conformance with the XML:DB API. The API can be considered generally equivalent to technologies such as ODBC, JDBC or Perl DBI.

Through web services

- An application will send requests to a webservice that in turn will then send database queries.

7. Web services

The W3C defines a Web service as a software system designed to support interoperable Machine to Machine interaction over a network. Web services are frequently just Web APIs that can be accessed over a network (e.g. Internet), and executed on a remote system hosting the requested services.

7.1 Overview

The Web service protocol stack is an evolving set of application layer protocols used to define, discover, and implement Web services. The core protocol stack has 4 layers:

- Transport: E.g. HTTP, SMTP, FTP, and newer protocols.
- XML messaging: E.g. Simple Object Access Protocol (SOAP) or XML-RPC. These define messages containing a service request and a response. SOAP and XML-RPC etc. are independent of any particular transport and implementation technology.
- Service description: E.g. Web Services Description Language (WSDL) - describes what a service does in a machine readable way
- Service discovery: E.g. Universal Discovery, Description, Integration (UDDI) - a service to publish available services

url: **http://en.wikipedia.org/wiki/Web_service**

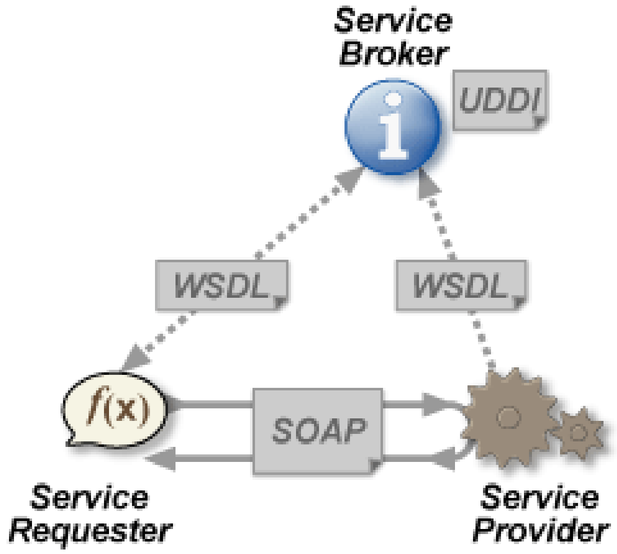
url: **http://en.wikipedia.org/wiki/List_of_Web_service_specifications**

url: **http://en.wikipedia.org/wiki/List_of_web_service_protocols**

url: **http://en.wikipedia.org/wiki/List_of_Web_service_Frameworks**

7.2 SOAP, UDDI and WSDL

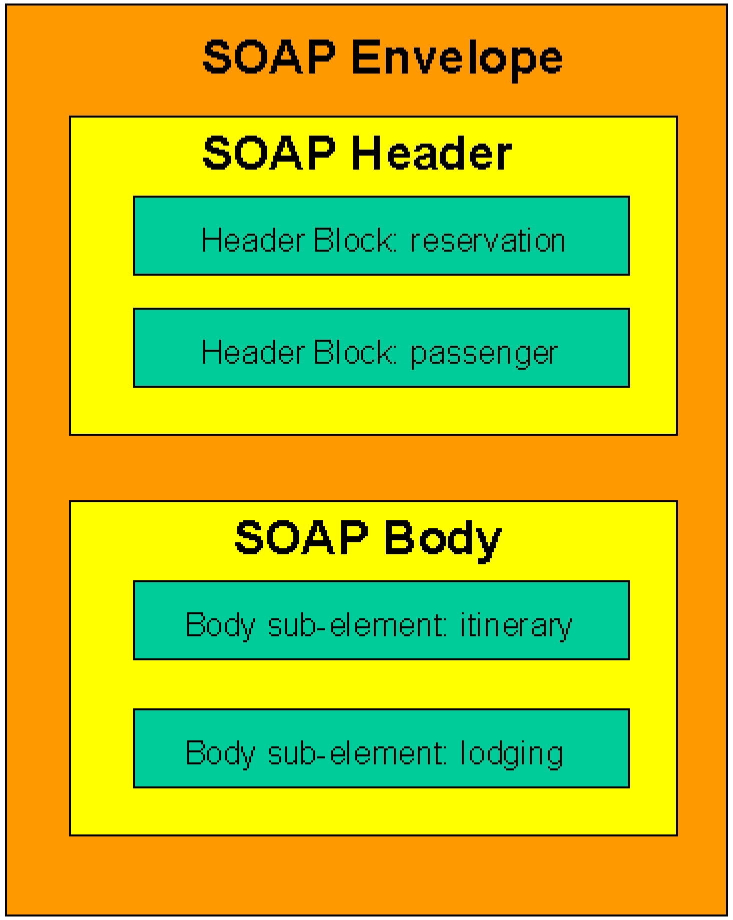
- SOAP - An XML-based, extensible message envelope format with "bindings" to underlying protocols. The primary protocols are HTTP and HTTPS, although bindings for others, including SMTP and XMPP, have been written.
  - SOAP originally stood for Simple Object Access Protocol, and lately also Service Oriented Architecture Protocol. SOAP became a W3C standard in 1993. Before SOAP existed other protocols such as XML-RPC (still popular) and CORBA or DCOM. Services like CORBA are less popular since they can’t go past most firewalls and since they are more complicated.
- Web Services Description Language (WSDL) - An XML format that allows service interfaces to be described along with the details of their bindings to specific protocols. Typically used to generate server and client code, and for configuration.
- Universal Description Discovery and Integration (UDDI) - A protocol for publishing and discovering metadata about Web services that enables applications to find them, either at design time or runtime.



7.3 SOAP

url: **http://en.wikipedia.org/wiki/SOAP**

- SOAP mostly uses either HTTP or HTTPS as transport protocol (so HTTP which is an application layer actually turns into a transport layer ...).
- With SOAP one can either send messages or remote procedure calls to another application
- Structure of a SOAP message to make a passenger reservation.


SOAP Envelope
SOAP Header
Header Block: reservation
Header Block: passenger
SOAP Body
Body sub-element: itinerary
Body sub-element: lodging

Example code of the passenger reservation

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"  env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
  <env:Body>
  <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>   <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>  <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>  <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate> <p:departureTime>mid-morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging
   xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
  </env:Body>
</env:Envelope>
```

7.4 XML-RPC

- XML-RPC is a remote procedure call protocol which uses XML to encode its calls and HTTP as a transport mechanism.

  url: **http://en.wikipedia.org/wiki/XML-RPC**
- XML-RPC was first created by Dave Winer of UserLand Software in 1998 with Microsoft. As new functionality was introduced, the standard evolved into what is now SOAP. Some people still prefer XML-RPC to SOAP because of its simplicity, minimalism, and ease of use.
- An example of a typical XML-RPC request would be:

```
<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
        <value><i4>40</i4></value>
    </param>
  </params>
</methodCall>
```

- An example of a typical XML-RPC response would be:

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
        <value><string>South Dakota</string></value>
    </param>
  </params>
</methodResponse>
```

7.5 REST

url: **http://en.wikipedia.org/wiki/REST**

- Representational State Transfer (REST) is a style of software architecture for systems like the World Wide Web. The term originated in a 2000 doctoral dissertation about the web written by Roy Fielding. It is very popular, because simple.
- Basically, REST only refers to a collection of architectural principles. The term is also often used to describe any simple interface that transmits domain-specific data over HTTP without an additional messaging layer such as SOAP. These two meanings can conflict as well as overlap.

Important REST concepts:

- the existence of resources (sources of specific information), each of which can be referred to using a single global identifier (a URI).
- A constrained set of well defined operations, e.g. GET, POST, PUT, DELETE (defined by the HTTP protocol).
- A constrained set of content types

8. eXist

8.1 eXist - an XML database system and portal

D. Schneider’s Test database:

url: **http://tecfax.unige.ch:8080/exist/**

Purpose

- eXist is a XML database and portal
- Open source Java, European team of developers led by Wolfgang Meier

Features

- XQuery
- XUpdate
- XQuery update)
- XInclude
- XSLT (including pluggeable extensions)
- Free-text searching (with extensions to XQuery - &=)
- Function libraries
- Database and user management, triggers

Database organization

- Documents (files) are organised in collections (folders)
- XML Documents stored in an efficient, B+ tree structure with indexes
- Non-XML resources (XQuery, CSS, JS, JPEG ..), etc also can be stored in the collections

- **Deployment**

  Deployable in different architectures

  - Embedded in a Java application using XML:DB API
  - Part of a Cocoon pipeline (Cocoon is an XML-based publication framework)
  - Ships with embedded Jetty HTTPServer, including a version of Cocoon

- **Multiple Interfaces**
  - Via servlet and Cocoon Xquery generators (i.e. through the eXist portal)
  - REST-style API (through normal HTTP GET,POST,PUT and DELETE URLs)
  - SOAP
  - XML:DB API
  - XML-RPC
  - WebDav

  ... Most of these Interface work best through Java libraries. Some are supported in Perl, Python or PHP (but documentation is really lacking).

8.2 Overview of the REST interface

- The rest interface is the most simple one to use, even without programming

Overview

GET

Retrieves a representation of the resource or collection from the database. XQuery and XPath queries may also be specified using GET's optional parameters applied to the selected resource.

PUT

Uploads a resource onto the database. If required, collections are automatically created, and existing resources are overwritten.

DELETE

Removes a resource (document or collection) from the database.

POST

Submits data in the form of an XML fragment in the content of the request which specifies the action to take. The fragment can be either an XUpdate document or a query request. Query requests are used to pass complex XQuery expressions too large to be URL-encoded.

8.3 Security

- Access to documents and operations depend on your user rights.
- Guests usually just have read access

8.4 Get REST requests

Summary of GET request parameters

_xsl=*XSL Stylesheet*	Applies an XSL stylesheet to the requested resource. If the _xsl parameter contains an external URI, the corresponding external resource is retrieved. Otherwise, the path is treated as relative to the database root collection and the stylesheet is loaded from the database. This option will override any XSL stylesheet processing instructions found in the source XML file. Setting _xsl to no disables any stylesheet processing. This is useful for retrieving the unprocessed XML from documents that have a stylesheet declaration.
_query=*XPath/XQuery Expression*	Executes a query specified by the request. The collection or resource referenced in the request path is added to the set of statically known documents for the query.
_indent=*yes* | *no*	Returns indented pretty-print XML.
_encoding=*Character Encoding Type*	Sets the character encoding for the resultant XML.
_howmany=*Number of Items*	Specifies the number of items to return from the resultant sequence.
_start=*Starting Position in Sequence*	Specifies the index position of the first item in the result sequence to be returned
_wrap=*yes* | *no*	Specifies whether the returned query results are to be wrapped into a surrounding <exist:result> element. The default value is yes.

- You may enter REST requests directly in the URL field of a web browser

Simple document retrieval:

Syntax: `http://`***`server_name`***`/exist/rest/db/`***`collection/`***`*.xml`

Examples:

```
http://tecfax.unige.ch:8080/exist/rest/db/shakespeare/plays/hamlet.xml
http://tecfax.unige.ch:8080/exist/rest/db/coap/cooking/dolores.xml
```

Result:

```
... the XML file + associated style
```

XML description of a document collection

Syntax: `http://.../exist/db/`***`collection/`***

Example:

```
http://tecfax.unige.ch:8080/exist/rest/db/coap
```

Result:

```
<exist:result xmlns:exist="http://exist.sourceforge.net/NS/exist">
   <exist:collection name="/db/coap" owner="coap" group="coap" permissions="rwurwur-u">
        <exist:collection name="cooking" created="2007-11-24T17:34:53.767+01:00"
owner="coap" group="coap" permissions="rwurwur--"/>
        <exist:collection name="dtd-examples" created="2007-11-24T16:51:53.201+01:00"
owner="coap" group="coap" permissions="rwurwur--"/>
        <exist:collection name="coap3180" created="2007-11-24T18:29:16.602+01:00"
owner="coap" group="coap" permissions="rwurwur--"/>
    </exist:collection>
</exist:result>
```

XPath and Xquery included

- Note: Using this interface to send complex queries is tricky since the XPath/XQuery expressions must be URL encoded (if your web browser doesn’t do it for you)

Served characters after percent-encoding																		
!	*	'	(	)	;	:	@	&	=	+	$	,	/	?	%	#	[	]
%21	%2A	%27	%28	%29	%3B	%3A	%40	%26	%3D	%2B	%24	%2C	%2F	%3F	%25	%23	%5B	%5D

Syntax: `http://.../exist/db/`***`collection`***`?_query=`***`XPath/Xquery_expression`***

Example 8-1: Simple XPath Query with a REST URL

Example 1 with Xpath (all in one line):

```
http://tecfax.unige.ch:8080/exist/rest/db/shakespeare?_query=//
SPEECH[SPEAKER=%22JULIET%22]&_start=1&_howmany=10
```

Example 2 with XPath:

```
http://tecfax.unige.ch:8080/exist/rest/db/shakespeare?_query=//
SPEECH[SPEAKER=%22JULIET%22]&_start=1&_howmany=10&_indent=yes
```

Example 8-2: XQuery with a REST URL

Example query

- must be all in one line !

```
http://tecfax.unige.ch:8080/exist/rest/db/?_query=for $t in document("/db/coap/cooking/
dolores.xml")//recipe let $name := $t//recipe_name return <name>{$name/text()} </name>
```

Result:

```
<exist:result xmlns:exist="http://exist.sourceforge.net/NS/exist" exist:hits="2"
exist:start="1" exist:count="2"><name>Cold Salmon in Creamy Spiced Sauce</
name><name>Fried Chicken with Cashew Nuts</name></exist:result>
```

Example 8-3: XQuery with a REST URL and our own XML wrapper

- Note: Must be all in one line !
- Results then will have to transformed for display, e.g. with a XSLT stylesheet.

  Example a with XQuery

  ```
  http://tecfax.unige.ch:8080/exist/rest/db/?_wrap=no&query=<result> { for $t in
  document("/db/coap/cooking/dolores.xml")//recipe let $name := $t//recipe_name return
  <name>{$name/text()} </name> } </result>
  ```

  Result:

  ```
  <result><name>Cold Salmon in Creamy Spiced Sauce</name><name>Fried Chicken with Cashew
  Nuts</name></result>
  ```

Example 8-4: HTML-like result

- This is your best bet if you just want to use an eXist database with simple links.

  ```
  http://tecfax.unige.ch:8080/exist/rest/db/?_wrap=no&_query=<html> <body>List of names:
  <ul> { for $t in document("/db/coap/cooking/dolores.xml")//recipe let $name := $t//
  recipe_name return <li>{$name/text()} </li> } </ul></body></html>
  ```

  Result:

  ```
  <html><body>List of names: <ul><li>Cold Salmon in Creamy Spiced Sauce</li><li>Fried
  Chicken with Cashew Nuts</li></ul></body></html>
  ```

Example 8-5: XQuery with a REST URL and and XSLT stylesheet

- Stylesheet can be in the eXist server or somewhere else. eXist will apply it before the query is executed. Not all that useful. Therefore you may want to disable it with "_xsl=no"

8.5 PUT REST Requests

- Documents can be stored or updated using an HTTP PUT request. The request URI points to the location where the document will be stored.
- You might send the PUT request via a script (e.g. PHP)

8.6 POST REST Requests

- POST requests require an XML fragment in the content of the request, which specifies the action to take.
- In order to send these, you must use a scripting language like PHP. Maybe it also can be done with XForms.

Structure of a POST Request

POST requests require the following type of XML fragment in the content of the request, which specifies the action to take.

You might send the POST request via a script (e.g. PHP)

```
<query xmlns="http://exist.sourceforge.net/NS/exist"
    start="[first item to be returned]"
    max="[maximum number of items to be returned]">
    <text>[XQuery expression]</text>
    <properties>
        <property name="[name1]" value="[value1]"/>
    </properties>
</query>
```

Example

```
<query xmlns="http://exist.sourceforge.net/NS/exist"
    start="1" max="20">
    <text>
        for \$speech in //SPEECH[LINE &= 'corrupt*']
        order by \$speech/SPEAKER[1]
        return
            <hit>{\$speech}</hit>
    </text>
    <properties>
        <property name="indent" value="yes"/>
    </properties>
</query>
```

8.7 Upload xq files and retrieve with the GET Rest Interface

- You can upload xq files to the database and (if permissions are ok) just retrieve them with a REST URL.

Example 8-6: