Web databases (COAP 3180)

Code: web-db

Author and version

- Daniel K. Schneider
- E-mail: Daniel.Schneider@tecfa.unige.ch

- Version: 1.0 (modified 28/2/10 by DKS)

Prerequisites

- Using web browsers
- Some HTML

Availability

url: **http://edutechwiki.unige.ch/en/Help:COAP-3180** **COAP 3180 Course Homepage**

Objectives

- Review of how Internet and the web works
- Advance view on the role of databases

Disclaimer

- There may be typos (sorry) and mistakes (sorry again)

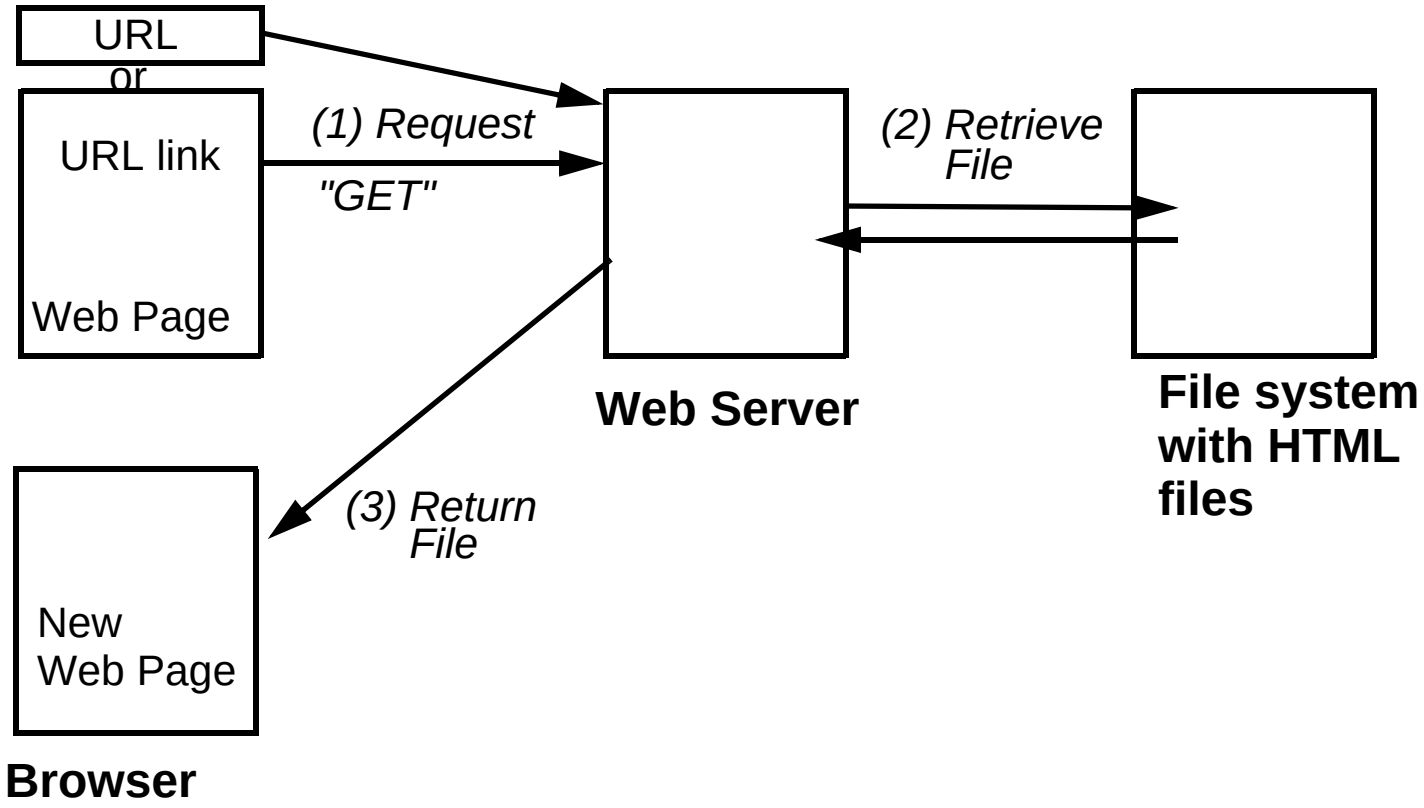
Contents

1. How Internet and the web works

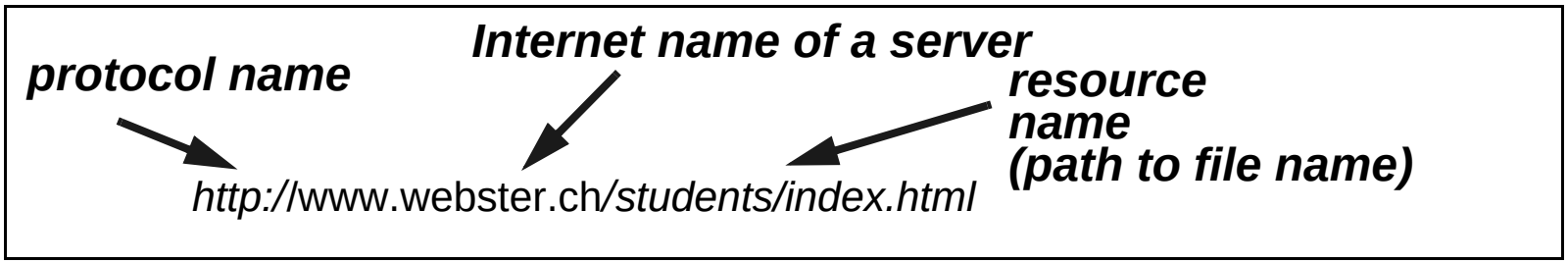
1.1 A little bit of History

- 1962: Invention of modern networking architecture (packets)
- 1969: first trial of Arpanet (future Internet)
- 1973: TCP, Transmission Control Protocol
- 1978: TCP/IP Protocol - Addition of IP, the Internet Protocol
- Seventies: Mail, forums, file transfer, remote connection services ....
- 1992: WWW, the World-Wide Web - an Internet service that relies on HTTP and HTML
- 1995: The World discovers the WWW, webserver - database interactions
- 1998: XML
- 2007: 500 million connected computers, hundreds of protocols and services

1.2 Simple web pages



- URL = Universal Resource Locator = unique addresses for resources.
  `Syntax: URL = protocol://address/resource name`



1.3 The HTTP Protocol

- Hypertext Transfer Protocol (HTTP) is a communications protocol used to transfer or convey information on the World Wide Web. (Wikipedia)

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

- Sample message sent from browser to server

```
GET /index.html HTTP/1.1
Host: www.example.com
```

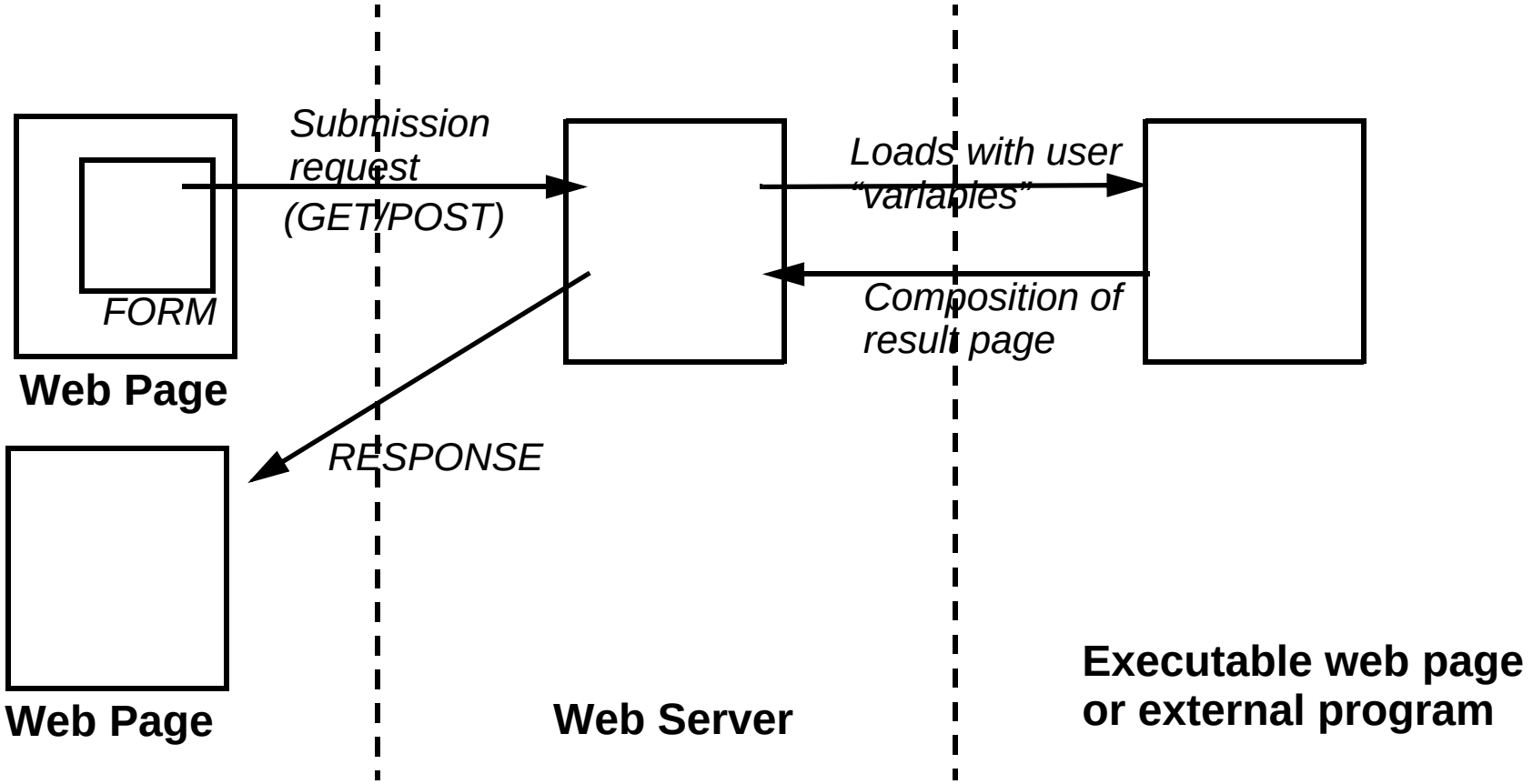
- Sample message sent from server to browser in response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.27 (Unix)  (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
```

Principles:

- Browser tells server what ressource it wants and also what browser can deal with + user preferences
- Server returns the resource (if things go well) and tells the browser what kind of resource it is (e.g. HTML).

1.4 HTML Forms and server-side scripts



Form example (see next slide for HTML code):

Form example (HTML code)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head><title>Simple test with</title></head>
<body>
<h1>Simple test with PHP</h1><hr>
 <form action="process-it.php" method="post">

 What do you know about HTML ?
 <input type="radio" name="choice" value="1" checked>little
 <input type="radio" name="choice" value="2">some
 <input type="radio" name="choice" value="3">everything

 <br>
 What is your programming experience ?
 <input type="radio" name="choice2" value="1" checked>none
 <input type="radio" name="choice2" value="2">some
 <input type="radio" name="choice2" value="3">good
 <P>
 <input type="submit" value="See result!">
 </form>
</body>
</html>
```

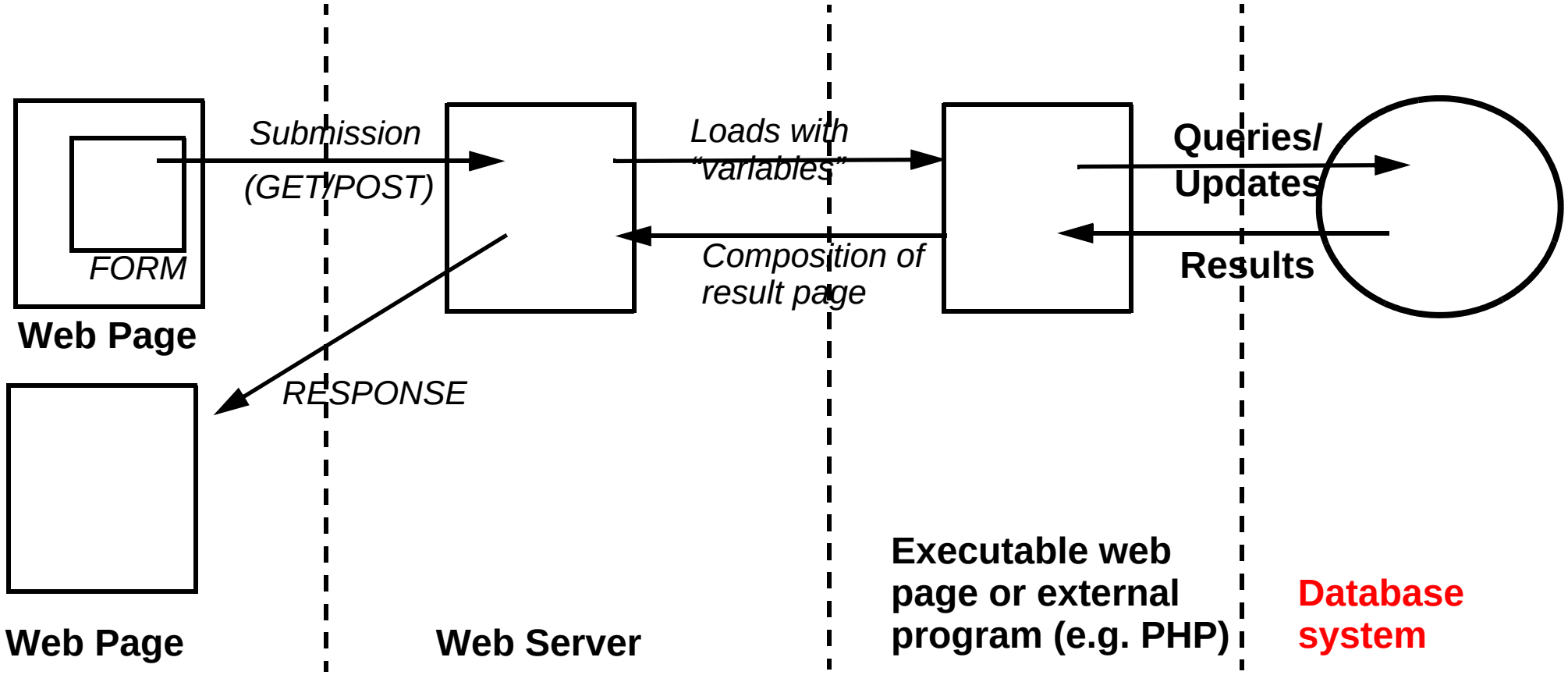
Form example (PHP code)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head><title>Simple test with</title></head>
<body>
 <h1>Simple test with PHP</h1><hr>
 <?php

  // Get variables from the form
  $choice = $_POST['choice'];
  $choice2 = $_POST['choice2'];

  // Compute the score
  $score = $choice + $choice2;
  // Display the score
  echo "<h3>Your score is " . $score . "</h3>";
  if ($score < 3) {
    echo "<p>You are a beginner</p>";
  } elseif ($score < 5) {
    echo "<p>You have some knowledge</p>";
  } else {
    echo "<p>You are an expert !</p>";
  }
  ?>
</body>
</html>
```

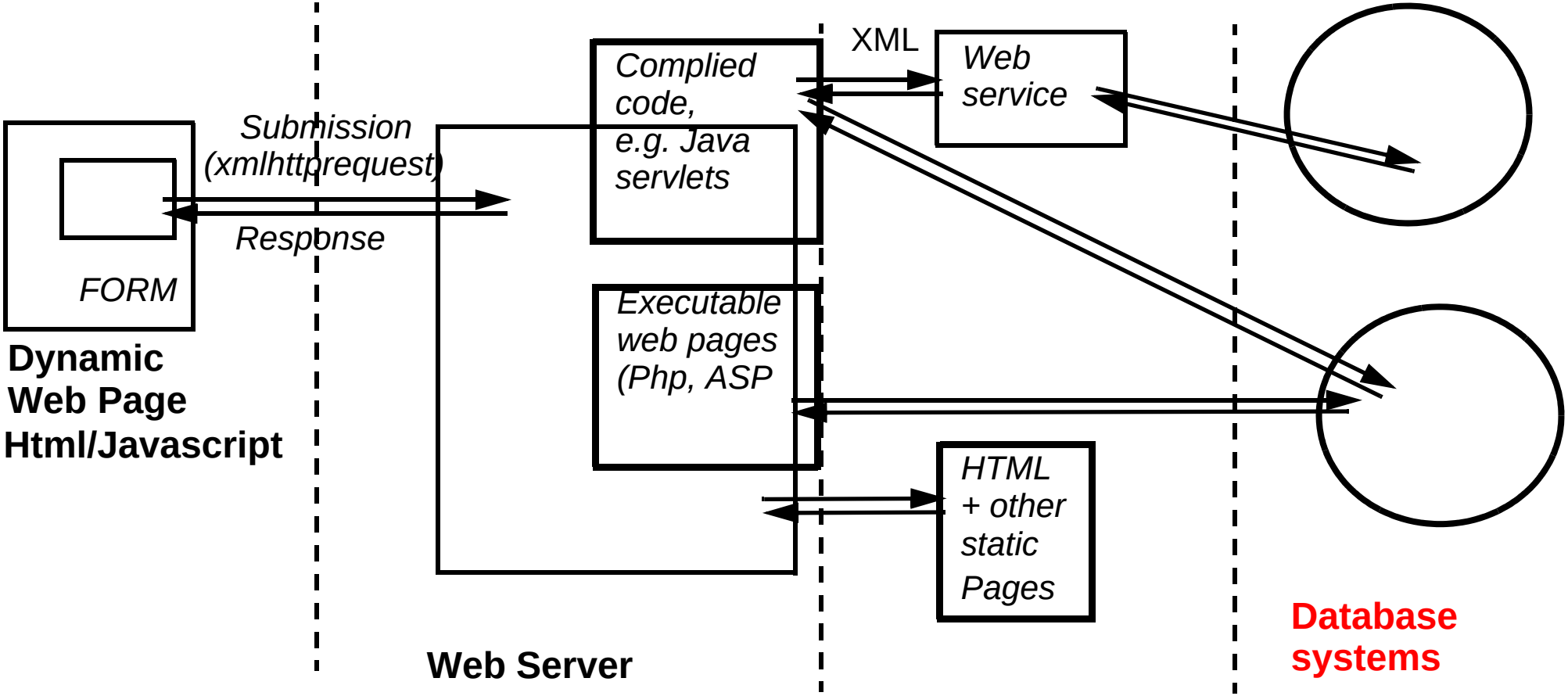
1.5 HTML Forms, server-side scripts and databases



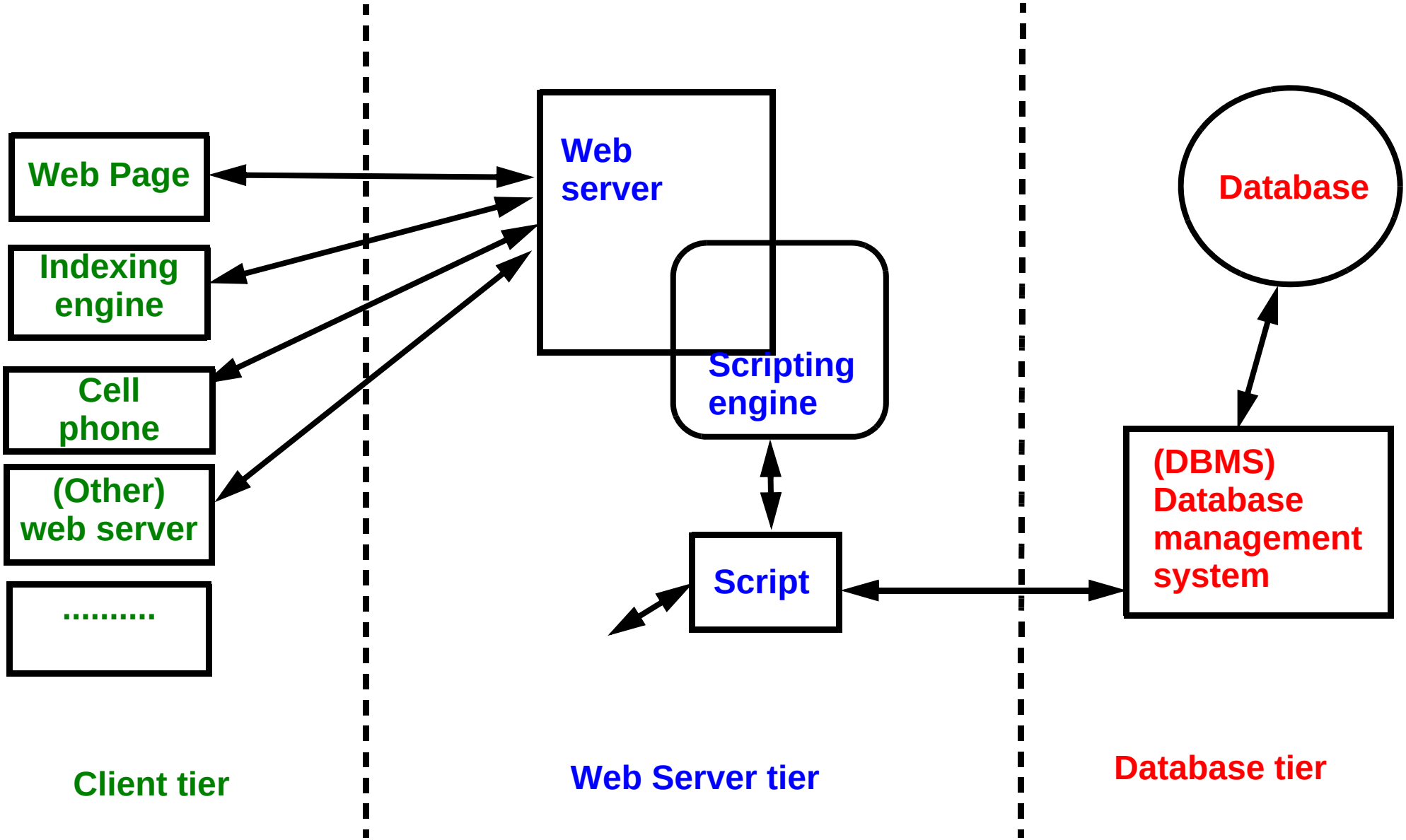
1.6 PHP-MySQL example code - HTML output

```
<?php
$link = mysql_connect( "localhost", "nobody", "")
        or die ("Unable to connect to SQL  server");
mysql_select_db("demo", $link)
        or  die ( "Unable  to  select  database");
$result = mysql_query( "select  *  from  demo1  limit  100")
        or die ("Data not found. Your SQL query didn't work... ");
?>
 <table  border="1">
 <?php
 while  ($row  =  mysql_fetch_row($result))  {
     echo  "<tr>";
     for  ($i=0;  $i<mysql_num_fields($result);  $i++)  {
       echo  "<td>";
       echo  "$row[$i]";
       echo  "</td>";
     }
     echo  "</tr>";
     }
 ?>
 </table>
```

1.7 A more complex setup ....



2. The typical three-tier architecture model



3. Role of XML

XML can play multiple roles with respect to web databases

(1) As communication format

- Data can be XML before they are put into a database (insures data integrity)
- Output from database can be transferred in XML format (separates content from form)
- Servers can communicate via XML communication languages (e.g. SOAP)

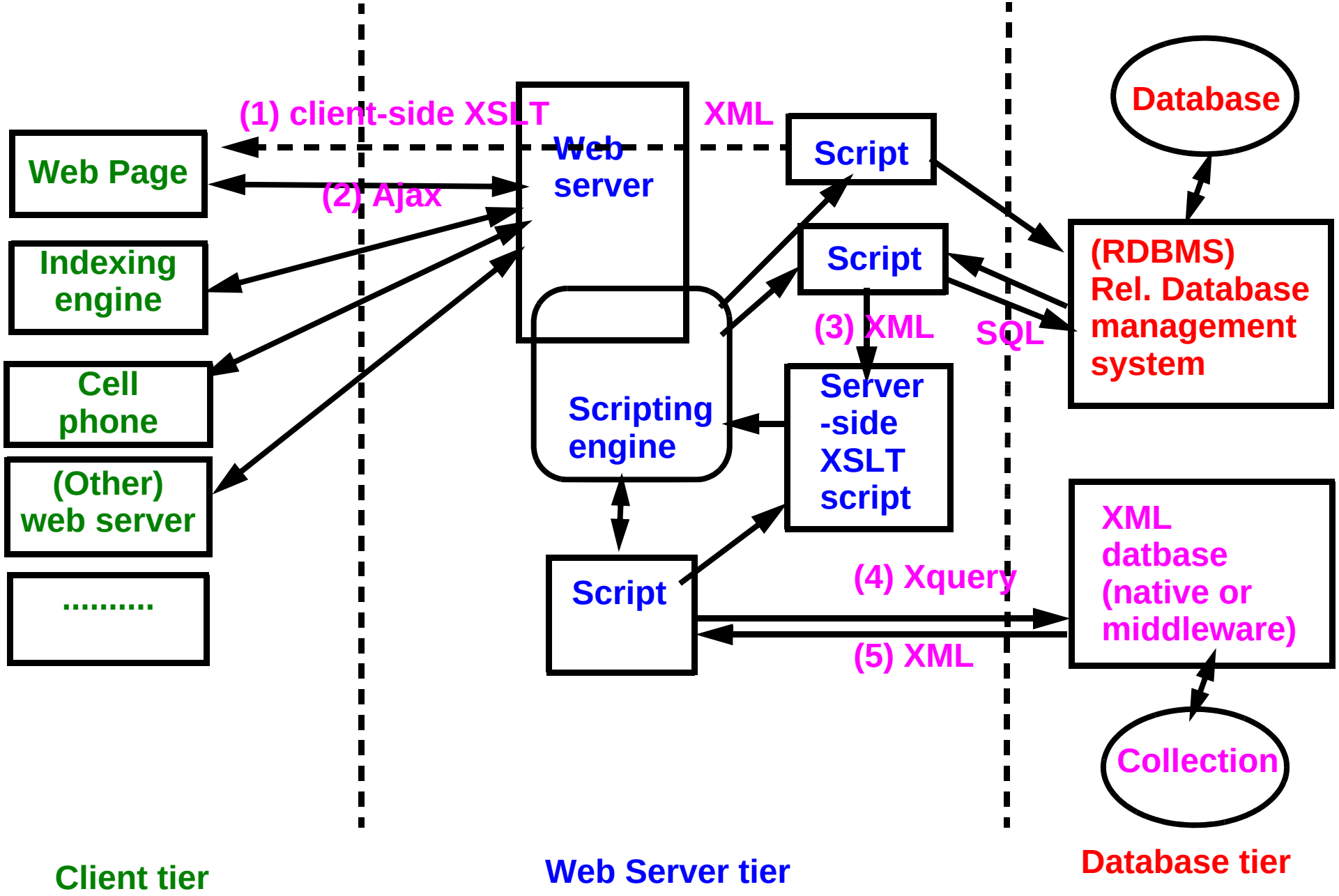
(2) As data format

- XML files can be considered a very simple form of database (e.g. a long list of references)
- XML can be stored in relation databases, e.g. some short elements in fields to be indexed and the rest in large blobs.
- XML can be stored as XML in a database, usually an XML database

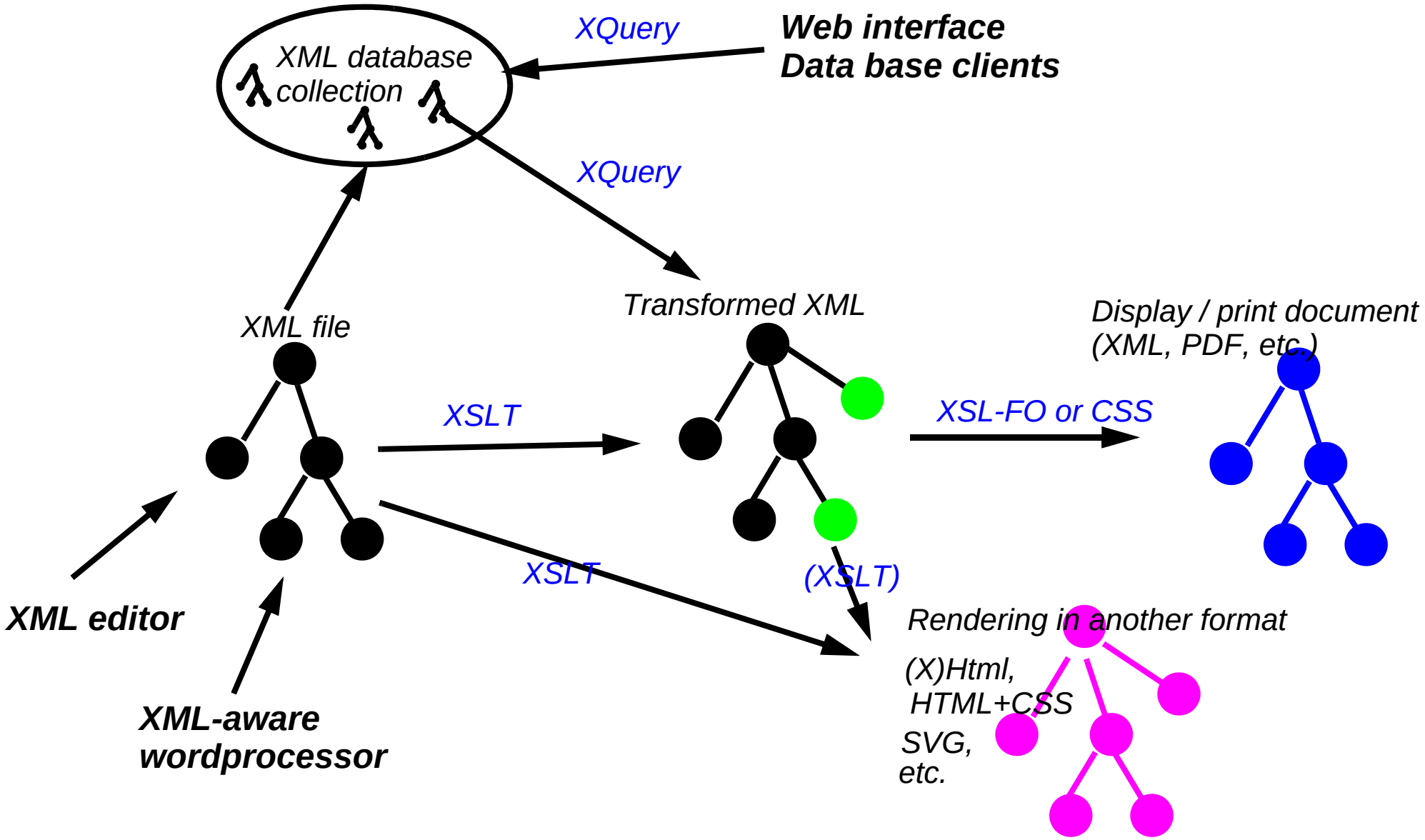
Native XML databases

- Native XML databases can preserve physical structure (entity usage, CDATA sections, etc.) as well as comments, PIs, DTDs, etc.
- Native XML databases can store XML documents without knowing their schema (DTD), assuming one even exists.
- The only interface to the data in native XML databases is XML and related technologies (such as XQuery, XPath, the DOM) or an XML-specific API, such as the XML:DB API. XML-enabled databases, on the other hand, offer direct access to the data, such as through ODBC.

3.1 Some XML in a simple three tier architecture model



3.2 XML document workflows



4. Types of databases and technologies

A global definition

"In computer science, a database is a **structured collection of records or data** that is stored in a computer system so that a computer program or person **using a query language** can consult it to answer queries. The records retrieved in answer to queries are information that can be used to make decisions. The computer program used to manage and query a database is known as a database management system (DBMS)." (Wikipedia, retrieved 22:30, 12 September 2007 (MEST)).

4.1 Types

1. Simple/flat (e.g. a table with user name, password and name, e.g. excel sheets). Rows are entries and colums define various data types that can be entered.
2. Hierarchical (e.g. LDAP authentication/address serveurs, some XML datastructures, Windows registry): Objects are inserted within other objects. Each object can have both properties and child objects.
3. Relational (mainstream today, in particular SQL databases): Data entries describe properties of an object like in the simple model, but can be linked to identifiers of other data entries.
4. Object and object-relational databases (not covered in this course)
5. Native XML datastores

5. Database connectivity

(1) Direct connections

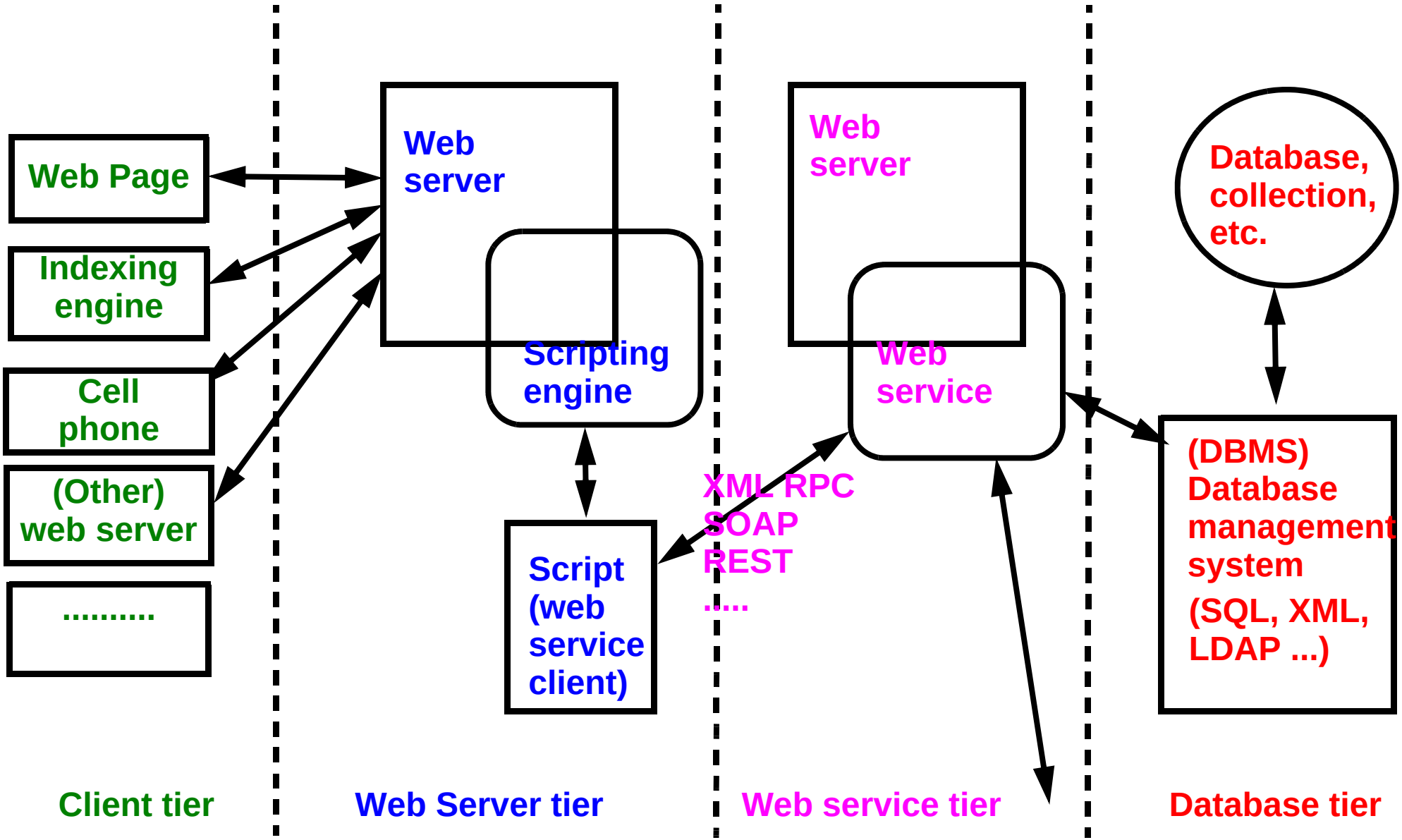
- APIs that allow a client (e.g. a database tool or a scripting language to connect directly to a database).
- Examples
  - *Open Database Connectivity* (ODBC) specification offers a procedural API for using SQL queries to access data. An implementation of ODBC will contain one or more applications, 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.
  - *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.

(2) Through web services

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

6. Web services

A simplified picture ....



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.

6.1 Overview of the W3C web service protocol

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**

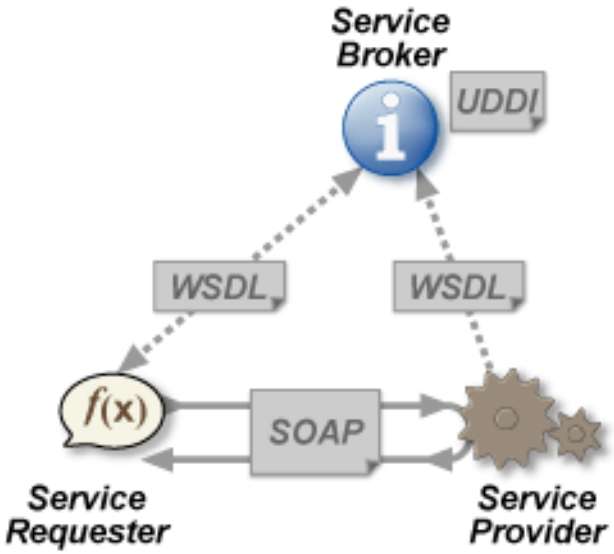
url: **http://en.wikipedia.org/wiki/List_of_Web_service_specifications**

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

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

6.2 SOAP, UDDI and WSDL

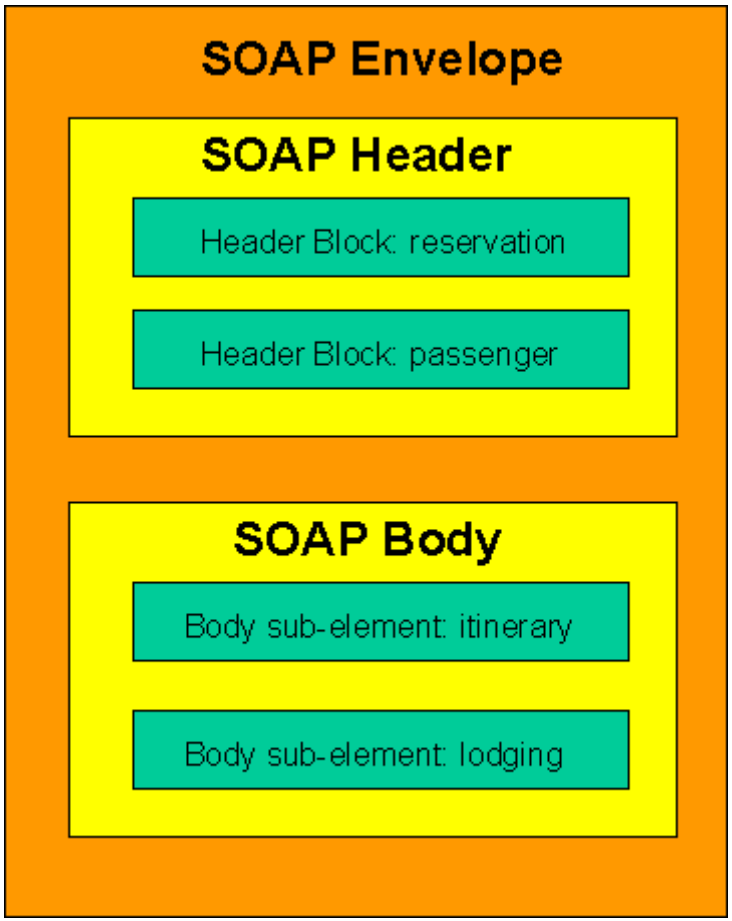
- **SOAP** - An XML-based, extensible message envelope format with "bindings" to underlying protocols. The primary protocols are HTTP and HTTPS, although bindings for others, including SMTP and XMPP, have been written.
  - SOAP originally stood for *Simple Object Access Protocol*, and lately also *Service 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.



6.3 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.



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>
```

6.4 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>
```

6.5 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

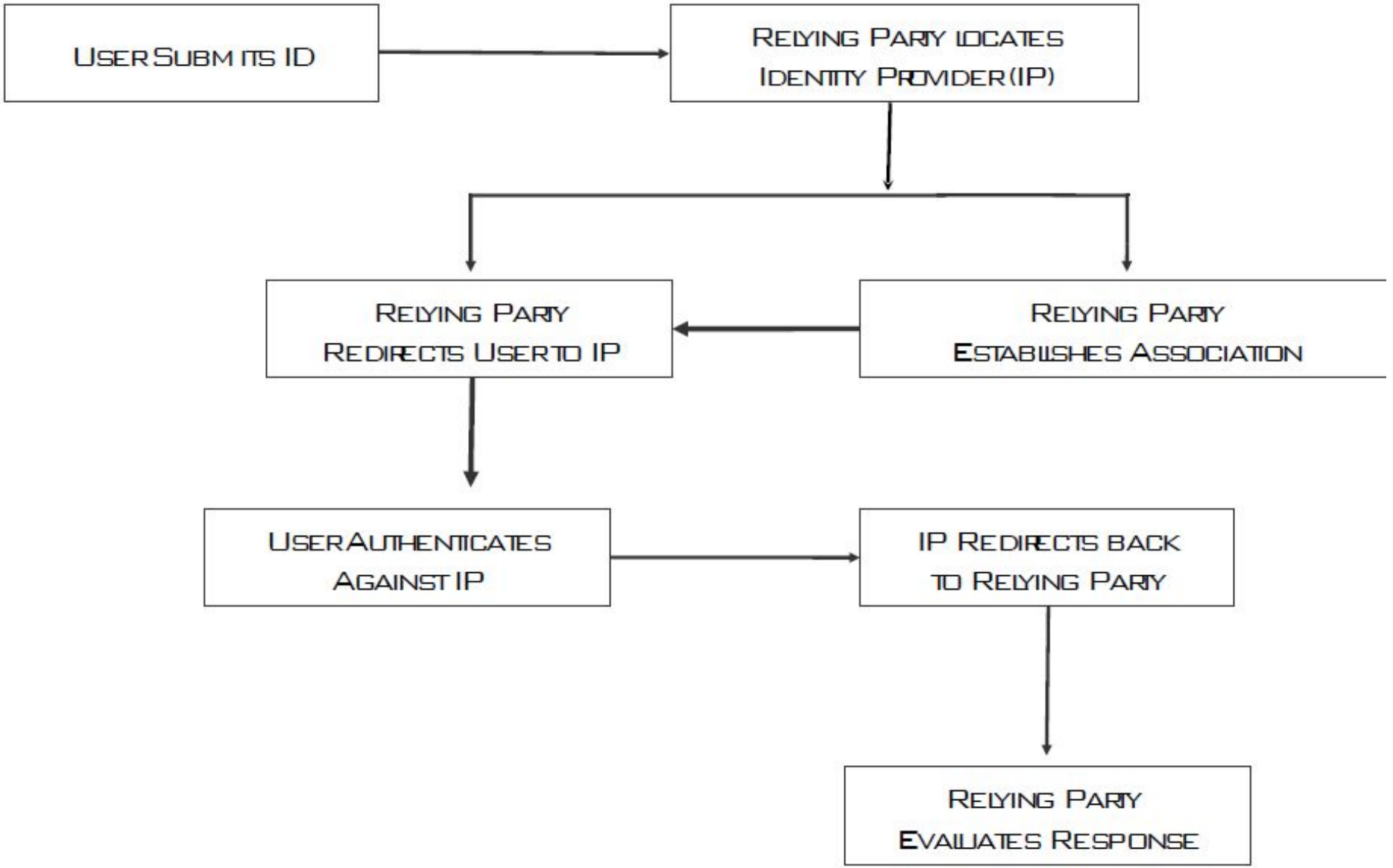
REST in simple terms:

- An application will request stuff from another server with a simple URL. E.g. a query to an XML database that uses a simle XPath expression, could look like this:

```
http://server.coap.net/exist/rest/db/shakespeare?_query=//
SPEECH[SPEAKER=%22JULIET%22]&_start=1&_howmany=10
```

7. Digital identity

- See also http://edutechwiki.unige.ch/en/Digital_identity



8. Things a web designer should know

- Create simple database applications
  - Create typed tables with web 2.0 services, enter data and connect these with your own webpages
  - generate very simple PHP/MySQL applications with generator software
- Install and/or use XAMP (Apache/MySQL/PhP) systems:
  - Simple web server configuration
  - Understand how database systems like MySQL work
  - Understand what the PHP scripting engine can do (and its dangers)
- Configure LAMP/WAMP-based applications (various kinds of portals)
  - Create databases and/or learn how to connect to a database with a user name and a password
  - Give the right answers to installation scripts
  - Configure the applications (modules, skins, users, etc.)
- Repair data of web applications
  - E.g. remove SPAM with SQL queries
  - Extend applications with modules (know how to submit SQL code to a db management tool)
- Understand various roles of XML in modern architectures, e.g.
  - Transform database output from XML to HTML with XSLT
  - Learn about XML databases, e.g. XQuery and XQuery update basics
- Understand the role of OpenID, LDAP and similar authentication/address book applications

At the end of the course you should have acquired the necessary foundations for moving on