Server-side processing

Problem

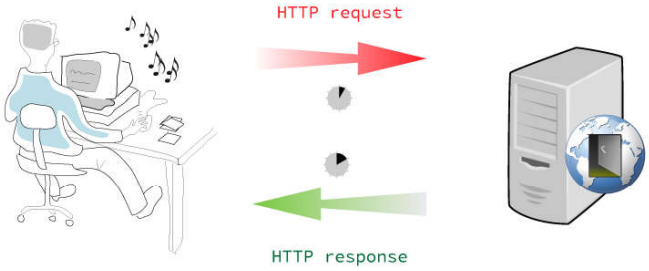
- HTML is too basic
  - Designed to display static page
  - Can’t access databases, spreadsheets , etc.
  - No security capabilities in HTML it self

What is Server-Side Processing?

- Technologies for developing web pages that include *dynamic* content—that is web applications.
- Can produce web pages that contain information that is connection- or time-dependent.
- A key technology for on-line shopping, employee directories, personalized and internationalized content.

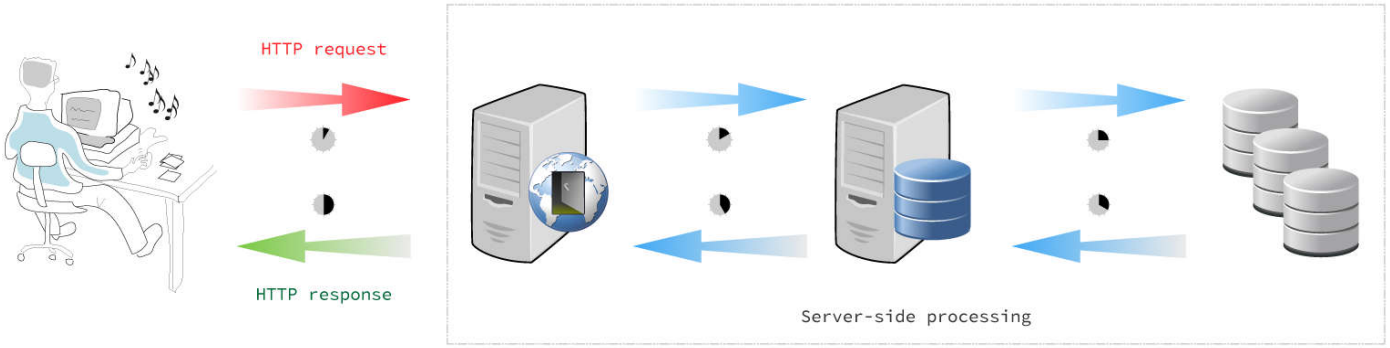
Scheme A

Static Website



Scheme B

Dynamic Website



Serve-Side Scripting

- Server-side scripting
  - Reside on server
  - A major use is database access
  - Usually generates custom response for clients
  - Cross-platform issues not a concern
  - Not visible to client
    - Only HTML + client-side scripts sent to client

CGI

- CGI : Common Gateway Interface

- Set of standard methods and routines used to write stand-alone software programs that know how to receive requests from a web server and return data to the server.

CGI (Cont)

- Allow browser to submit data to a program running on the server
  - Program is often called a 'CGI script'
  - Typically written in Perl
  - Can also be a 'real' program (e.g. Written in c)
- Used primarily for form submission
- Output from CGI usually dynamic and therefore not cached

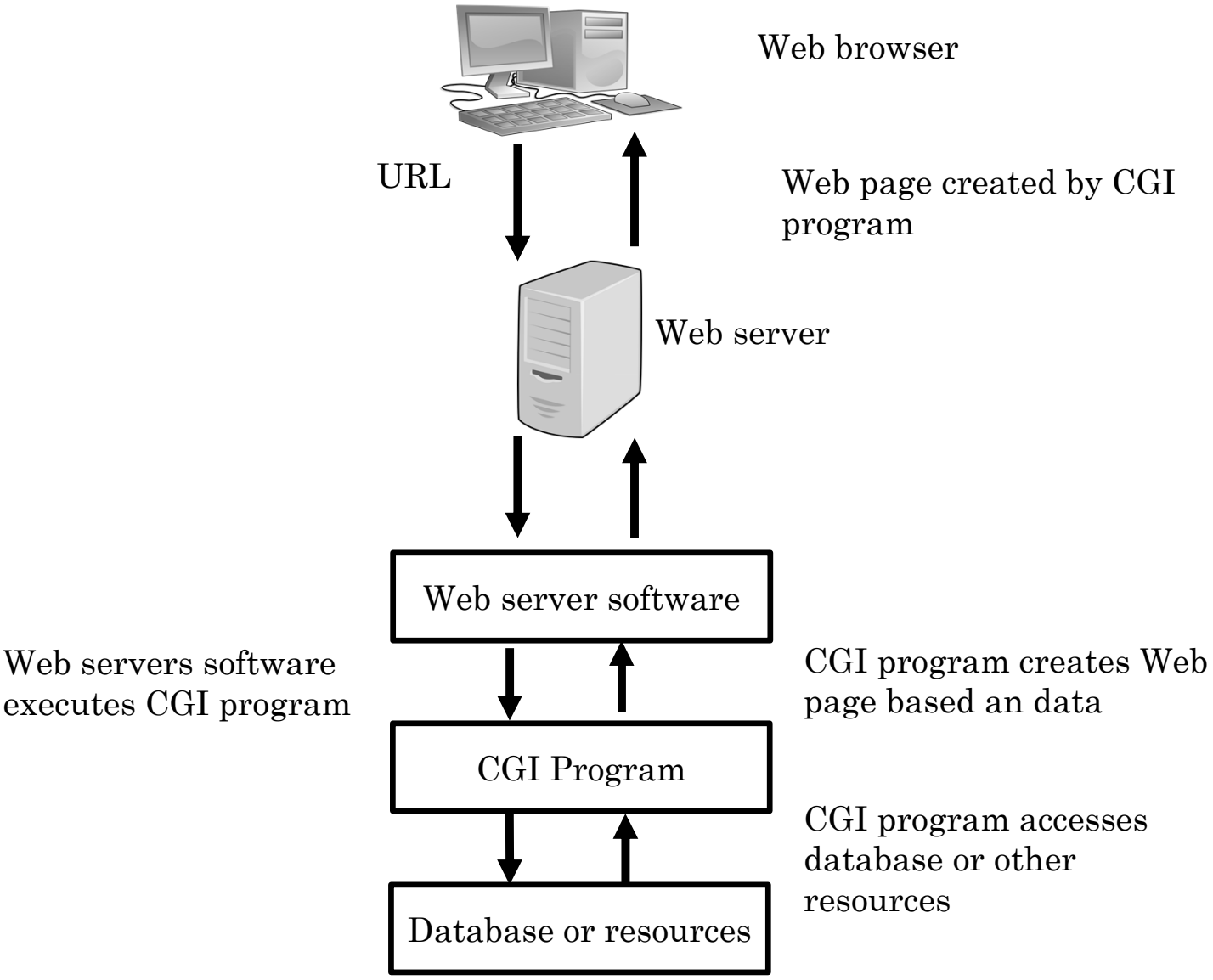
CGI process flow

- Step 1: A user, accessing an HTTP browser, sends a request to an HTTP server via HTML.
  - This HTML includes a request to execute a CGI program, and any parameters the CGI program might need.

- Step 2 : The HTTP server receives the request from the browser, processes the HTML, and encounters the request to execute a CGI program.

CGI process flow (Cont)

- Step 3: The CGI program executes. In its execution, it may:
  - Access no other resources.
  - Access database either locally or remotely
  - Access other applications or initiate the execution of other programs
  - Access other network resources
- Step 4: The HTTP server receives a result set from the CGI program, and sends the data and/or response back to the HTTP browser via HTML
- Step 5: The HTTP, browser receives the HTML sent to it from the HTTP server and formats and displays the data received


Web browser
URL
Web page created by CGI program
Web server
Web server software
Web servers software executes CGI program
CGI program creates Web page based an data
CGI Program
CGI program accesses database or other resources
Database or resources

CGI and database

- HTML has no facilities to directly query a database.
- Through CGI, this capability exists.
- By utilizing CGI scripts, a request can be sent from within HTML, and processed by HTTP server, to query the database for specific information, and then display the result set in dynamically built HTML code.

CGI and database(Cont)

- With this capability there is no need to manually change a web page whenever data on that page changes.
- Simply place the data in a database, and build a CGI script to access the data and display it dynamically .
- Whenever a request is made to view the page that contains the CGI script, the web server initiates a request to database and formats the most current data into a dynamic Web page.

ASP: Active server pages

- A Microsoft server based scripting environment designed for dynamic content
- An ASP page in an HTML page that contains server-side scripts that processed by the web server before being sent to the user's browser
- Server-side script run when a bowser requests an .aspx fie from the web server
- Generally can only be used on windows servers and web servers

Example: handling a form data

```
<%
IF request (“username”)= “ahmed” &&
request (“password”)= “1234” then
runDemo()
%>
```

PHP

- Initially as a simple set of Perl scripts for tracking accesses to online resume
- Php “Personal Home Page Tools”
- A scripting language designed for the web
- Designed similar to Active server pages
  - Embed php commands int web pages
- Open sources, low cost
- Interpreted, not compiled
  - Cross-Platform
  - Embedded in HTML

Php example

- Embedding php in html

```
<html>
<body>
<strong> hello world </strong>
<?
        echo 'This is a PHP message!';
?>
</body>
</html>
```

Cookies

- Small piece of data generated by a web server, stored on the client's hard drive.

- Servers as an add-on to the http specification (remember, HTTP by itself is stateless)
- Controversial, as it enables web sites to track web users and their habits.

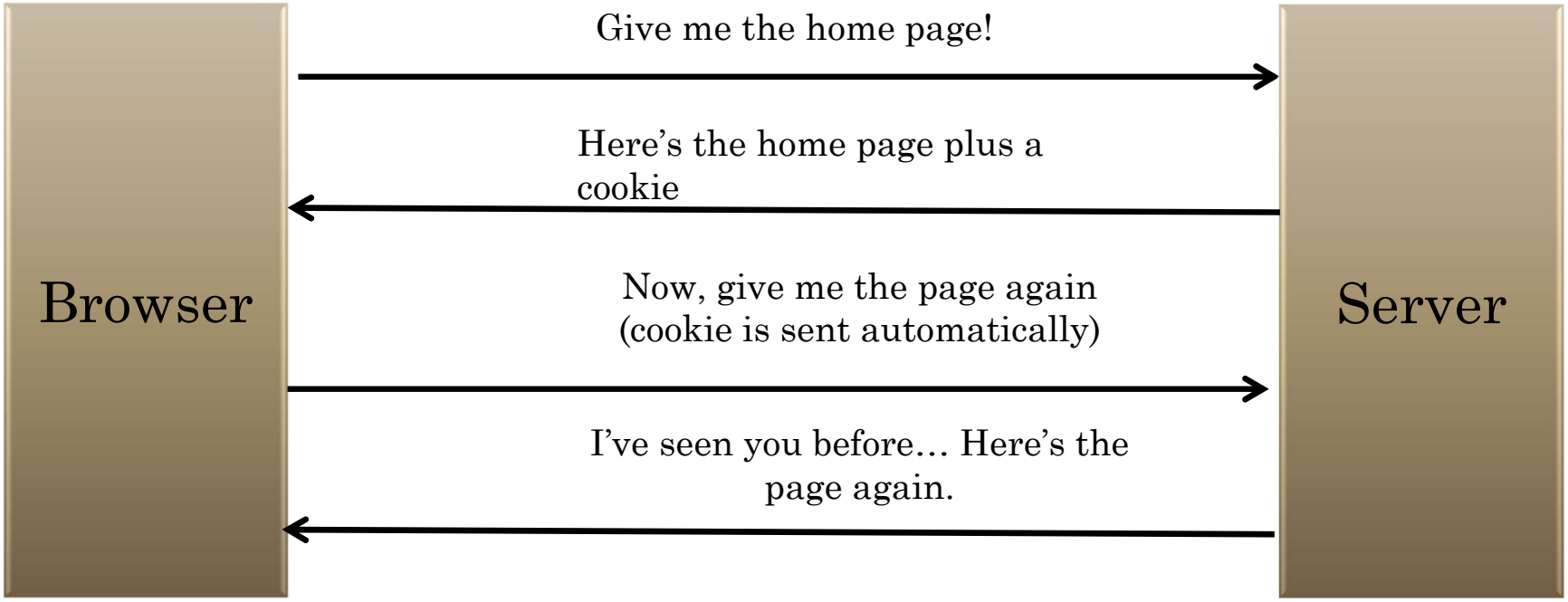
A Cookie's scenario

- Web site xyz.com wants to track the number of unique visitors who access its site.
- If xyz.com checks the HTTP servers logs, it can be determine the number of "hits", but cannot determine the number of unique visitors.
- That's because HTTP is stateless. It retains no memory regarding individual users .
- Cookies provide a mechanism to solve this problem

Tracking Unique Visitors

- Step 1: Person A request home page from xyz.com
- Step 2: xyz.com Web server generates a new unique ID
- Step3 : Server returns home page plus cookie set to the unique ID
- Step 4: Each time person A returns to xyz.com, the browser automatically send the cookie along with the GET request.

Cookie Conversation

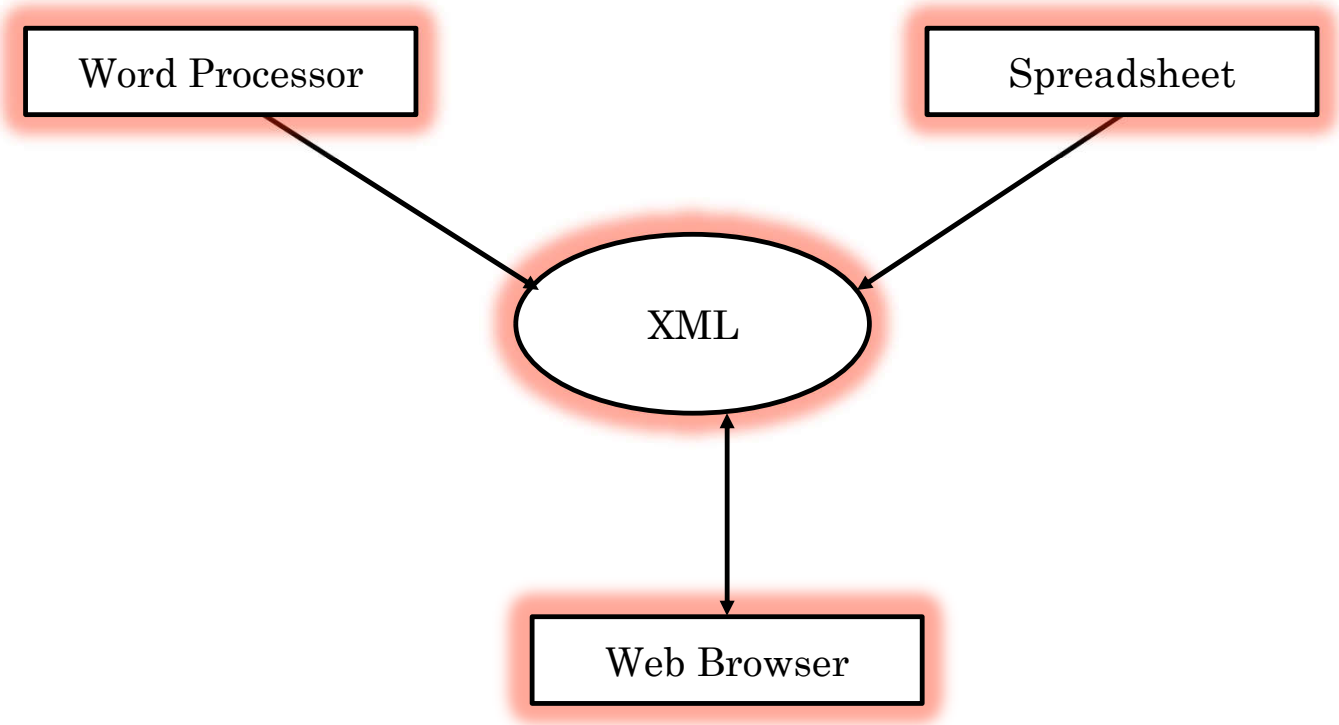


Why use Cookies?

- Tracking unique visitors
- Creating personalized web site
- Tracking users across your site:
  - E.g. do users that visit your sports news page as visit your sports store?

XML
Extensible Markup Language

Document exchange



Components of a document

- **Content:** the components (words, images, etc). Which make up a document.
- **Structure:** the organization and inter-relationship of the components
- **Presentation:** how a document looks and what processes are applied to it.

Separating these things means…

- Content can be re-used
- Structure can be formally validated
- Presentation can be customized for
  - Different media
  - Different audiences
- The information can be uncoupled from its processing

What is metadata?

- Data about data

- Data associated with objects which relieves their potential users of having to have full advance knowledge of their existence or characteristics.

What is XML?

- XML stands for **EX**tensible **M**arkup **L**anguage
- It is called extensible because it is no a fixed format like HTML
- XML is set of rules for designing text formats that let you structure data
- XML tags are not predefined. You must **define your own tags**

What does XML Look Like?

- It is only a text file and it doesn't require you to have a particular operating system or hardware.

```
<?xml version="1.0"?>
<Document>
<Greeting>
Welcome to XML
</Greeting>
<Message>
This is an XML document. Bet you're surprised.
</Message>
</document>
```

Why XML?

- XML makes the structure of the document explicit to computer programs.
- An HTML page encodes information in a form easily processed by humans.
- HTML lacks a structure that facilitates information processing.

Example: It is hard to do the following with HTML

- News in HTML:
  - What's the headline of the story?
  - Who is the author?
- Product info in HTML:
  - What is the price of the item?
  - What category of item is it?
- If you want to publish information in a form that software clients can process it
  - You need to produce pages in which the structure is explicit for software to exploit.
- That's what XML does.

The different between HTML and XML?

- HTML was designed to **display** data and to focus on **how data looks**

- XML was designed to **structure** data and to focus on **what data is**

XML looks like a bit like HTML

- Like HTML , XML makes use of tags (words bracketed by '<' .. '>').
- HTML is a specific markup language that contains a fixed set of elements and attributes.
  - The tags used to HTML documents and the structure of HTML documents are predefined (I.e. <p>, <h1>,…)
- XML uses the tags only to delimit pieces of data , and leaves the interpretation of the data completely to the application that reads it.

XML vs. HTML: Information Storage

- HTML
  - Information is stored in HTML in its **final** form

- XML:
  - Information stored in XML can be presented in a variety of ways for different audiences and scenarios , since the data and display are separate.

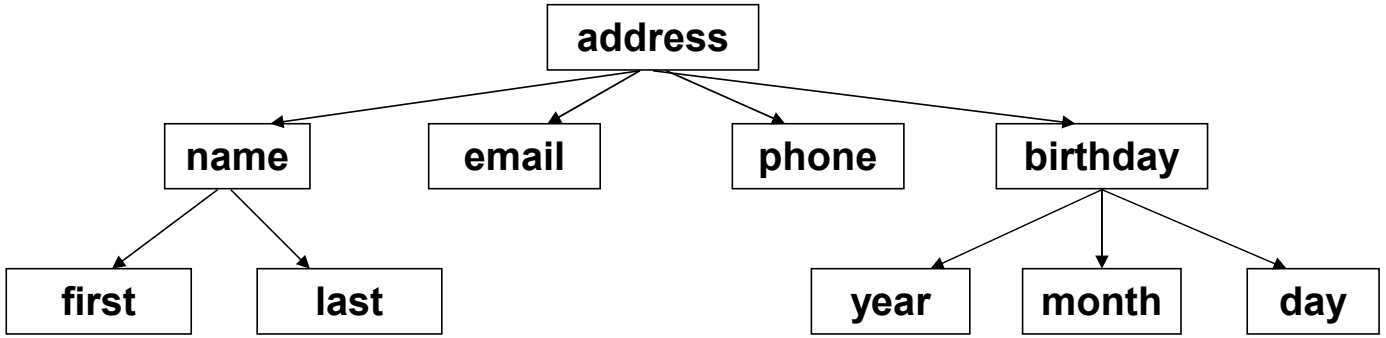
XML Rules

- Tags are enclosed in angle brackets.
- Tags come in pairs with start-tags and end-tags.
- Tags must be properly nested.
  - **<name><email>…</name></email> is not allowed.**
  - **<name><email>…</email><name> is.**

More XML Rules

- Tags are case sensitive.
  - **<address> is not the same as <Address>**
- XML in any combination of cases is not allowed as part of a tag.
- Tags may not contain '<' or '&'.
- Documents must have a single *root* tag that begins the document.

XML Files are Trees



XML Trees

- An XML document has a single root node.
- The tree is a general ordered tree.
  - A parent node may have any number of children.
  - Child nodes are ordered, and may have siblings.
- Preorder traversals are usually used for getting information out of the tree.

Steps of creating an XML file

- Discover (or establish) the structure of your data.
  - Use DTD or XML schema
- Build the XML file that holds the data.
- Apply a formatting style to the xml file.

Document Type Definitions

Document type definitions

- DTDs (**D**ocument **T**ype **D**efinitions) contain a list of element , tags, attributes and entity references contained in an XML document and describes their relationships to each other.

- Simply, DTD
  - Specifies a list of tags
  - Defines the relationships between these tags .

Document Type Definitions

- A DTD describes the tree structure of a document and something about its data.
- There are two data types, PCDATA and CDATA.
  - PCDATA is parsed character data.
  - CDATA is character data, used about text data not be parsed.
- A DTD determines how many times a node may appear, and how child nodes are ordered.

DTD for address Example

<!ELEMENT address (name, email, phone, birthday)>

<!ELEMENT name (first, last)>

<!ELEMENT first (#PCDATA)>

<!ELEMENT last (#PCDATA)>

<!ELEMENT email (#PCDATA)>

<!ELEMENT phone (#PCDATA)>

<!ELEMENT birthday (year, month, day)>

<!ELEMENT year (#PCDATA)>

<!ELEMENT month (#PCDATA)>

<!ELEMENT day (#PCDATA)>

Structure of a DTD

- A DTD always starts with <! DOCTYPE and always ends with]>
- Directly after the <! DOCTYPE comes the name of the document element followed by a [
- Then comes a list of all elements and attributes contained in the XML file, including the document element

Example of DTD

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

XML: Example of DTD

```
<!DOCTYPE employees [
<!ELEMENT employees (name,email,tel,fax)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
]>
```

Example: XML Structure

<employees>

<name>Karim</name>

<email>karim@yahoo.com</email>

<tel>00202352</tel>

<fax>00202536</fax>

</ employees>

Where do I get a DTD?

- Industry announcements
- Some recent examples
  - Chemical Markup Language(chemical modelling)
  - Math Markup Language
  - Etc.

Assignment

- What is XML Schema Definition (XSD)
- Compare it with DTD