Introduction

© Leonidas Fegaras
University of Texas at Arlington

Information

- **Class:** MW 2:30-3:30pm (NH 109)
- **Instructor:** Leonidas Fegaras
- **Office:** GACB 115 (General Academic Classroom Bldg)
- **Phone:** 817-272-3629
- **Email:** fegaras@cse.uta.edu
- **Office hours:** Monday and Wednesday 4:00-5:30pm (after class)
- **Web:** [http://lambda.uta.edu/cse6339/](http://lambda.uta.edu/cse6339/)
- **GTA:** ?

Visit the class web page often. It will contain reading assignments, project description, class notes, grades, etc.

Description

- XML has become an important standardization for data representation and information exchange among Internet co-operative applications.
- This course provides an in depth study of the area of web data management with an emphasis on XML standards and technologies.
- The course primarily covers the state of the art in designing and building web applications and services, primarily focusing on issues and challenges that revolve around the management and processing of XML data.

Prerequisites

- **Prerequisite:** CSE 3330/CSE 5330 (Database Systems I) or equivalent
- Students are expected to have a working knowledge of
  - Java
  - SQL
  - basic HTML
- Students without adequate preparation are at substantial risk of failing this course
Grading

- The final grade will be based on
  - 50% projects (10 small projects)
  - 20% midterm exam
  - 30% final exam (comprehensive)
- Final grades will be assigned according to the following scale:
  A: score >= 90, B: 80 <= score < 90, C: score < 80
- Sometimes, I use lower cutoff points, depending on the overall performance of the class
- Your grades will be available on-line on the course web page

Reading Material

- There is no required textbook but you are expected to read many online tutorials and references
- Links will be given out in class
  - Many good online tutorials, eg http://www.w3schools.com/default.asp
- Many books on XML standards and programming
  - See the syllabus for some recommended books

Exams

- Both exams are closed-book and closed-notes
- The final exam will cover the material from the first lecture up to
  and including the last lecture
- Makeup exams will be given only when the instructor (at least 3
days before the exam) has approved the request to change the
  exam time
  - Approval will be given for illness, sickness or death in the family only.

Projects

- There will be ten small weekly projects
- Each project will be done individually
- Details will be given out in class
- Late project will be marked 20%-off per day
  - No further extensions will be allowed
  - No excuses, no exceptions.
Software

- All projects will be done in Java (using JDK 6)
  - Students are expected to have a working knowledge of Java, SQL, and basic HTML.
- The software used for the projects is open-source, free, platform-independent, and well-suited for Java:
  - Java/web development platform: Sun's NetBeans 6.0 (Java Studio Creator)
  - Database connectivity: JDBC over MySQL (on omega)
  - Servlet container: Apache Tomcat
  - Web services: Apache Axis
- You can do the projects on your PC/laptop under any platform
  - Linux, MAC OS X, MS Windows, etc.
  - Directions of how to download the required software will be given out in class
- Although we will briefly talk about it, we will not use Microsoft ASP.NET (Visual Studio, C#, etc), since this framework is platform-dependent (for IIS only)

Cheating

- All work in this class must be done individually. No copying is permitted
- The punishment for cheating is a zero in the assignment and will be subject to the university’s academic dishonesty policy
- Cheating involves giving assistance to or receiving assistance from another individual on work assigned in this class
- If you have any questions regarding an assignment, see the instructor or the teaching assistant

Miscellaneous

- Distance Education Students:
  The requirements for distance education students are the same as for regular students with the possible exception of the exams
  If you are a distance ed. student and work within one hour driving distance from UTA, then you need to come and take the exams in person.
  Otherwise, you will have to find an exam proctor on site to supervise the exams. The proctor cannot be anyone equal or below your pay grade at your office, unless it is someone in HR that specializes in proctoring exams. The proctor could be someone from a local school, testing center, etc. The proctor must be approved by the instructor and a proctor agreement must be signed. The exam will be delivered to a proctor in the morning of the exam day.
- Special Accommodations:
  If you require an accommodation based on disability, I would like to meet with you in the privacy of my office, during the first week of the semester, to make sure you are properly accommodated.

Tentative Schedule

- Introduction and motivation
- Web application development
  - Dynamic web pages
  - HTTP GET/POST requests
  - HTML forms
  - Client-side programming (JavaScript)
  - XHTML and CSS stylesheets
  - The document object model (DOM) and dynamic HTML
  - Asynchronous server requests (AJAX)
  - Server-side programming: PHP scripts
  - Cookies and sessions
  - Servlets (Tomcat)
  - Java Server Pages (JSP)
  - Database connectivity
Tentative Schedule (cont.)

- XML standards
  - DTD
  - XML Schema
  - XPath
  - XML programming (DOM, SAX, StAX)
  - XSLT
  - XQuery
  - Java/XML data binding (JAXB)
- XML data modeling
- Native XML storage management
  - Indexing techniques
  - Xindice and Berkeley DB XML
- Relational databases and XML
  - XML shredding
  - XML publishing
- XML on commercial databases (Oracle XML DB, SQL Server SQLXML)

Web Databases and XML  L1: Introduction  13

Tentative Schedule (cont.)

- XML data management
  - Query processing and optimization
  - Updates
  - View maintenance
  - Integrity constraints
- XML search engines
  - Information retrieval
  - Web search engines
  - XML ranking
- Web services
  - Standards: SOAP, WSDL, UDDI
  - Axis and JAX-WS
  - Mashups
- Special topics
  - Metadata management with RDF, Data integration, Semantic Web, etc

Web Databases and XML  L1: Introduction  14

Traditional DB Applications

- Typically business oriented
- Large amount of data
- Data is well-structured, normalized, with predefined schema
- Large number of concurrent users (transactions)
- Simple data, simple queries, and simple updates
- Typically update intensive
- Small transactions
- High performance, high availability, scalability
- Data integrity and security are of major importance
- Good administrative support, nice GUIs

Web Databases and XML  L1: Introduction  15

Document Applications

- Human friendly: what-you-see-is-what-you-get paradigm
- Focus on presentation
- Information is divided into multiple small documents
- Mostly static
- Implicit structure: section, subsection, paragraph, etc
- Meta-data: title, author, date, indexing keywords, etc
- Content structure: form/layout, inter-relationships, references
- Tagging: eg. <p> for new paragraph
- Operations: retrieving, editing, spell-checking, printing, etc
- Information retrieval: keyword queries
  - most successful in web search engines (eg, Google)
Internet Applications

- Internet applications
  - use heterogeneous, complex, hierarchical, fast-evolving, unstructured/semistructured data
  - access mostly read-only data
  - need 100% availability
  - manage millions of users world-wide
  - have high-performance requirements
  - are concerned with security (encryption)
  - like to customize data in a personalized manner
  - expect to gain user’s trust for business-to-consumer transactions.

Internet users choose speed and availability over correctness

Electronic Commerce

- Currently, mostly business-to-business (B2B) rather than business-to-consumer (B2C) interactions
- Focus on selling and buying:
  - Order management
  - Product catalogs
  - Product configuration
  - Sales and marketing
  - Education and training
  - Web services
  - Web communities

Other Web Applications

- Web services
  - Many standards: SOAP, WSDL, UDDI
- Web integration
  - Heterogeneous data sources and types
  - Thousands of web-accessible data sources
  - Dynamic data
  - Data warehouses
- Web publishing
  - Access different types of content from browsers (PDF, HTML, XML)
  - Structured, dynamic, customized/personalized content
  - Integration with application
  - Accessible via major gateways and search engines
- Application integration
  - Transformation between different data formats (eg, XML, HTML)
  - Integration of multiple applications

Current Internet Application Architectures

- Architecture:
  - **Server-Tier**: relational databases and gateways to diverse data sources, such as, files, OLE/DB etc. Use of enterprise servers
  - **Middle-Tier**: provides data integration & distribution, query, etc. Consists of a web server and an application server
  - **Client-Tier**: mostly a web browser, may use CGI scripts or Java

- Characteristics:
  - Customization is achieved at the server site (customer data in a database) with some data at the client site (cookies)
  - Load balancing is typically hardware based (multiple servers, DNS routers)
**HTML**

```html
<html>
<head>
<title>My Web Page</title>
</head>
<body>
<hi>Introduction</hi>
look at <a href="http://lambdascape.uta.edu/index.html">this document</a>
<img src="image.jpg" width="100" height="50">
</body>
</html>
```

- It's a *markup* language: text (content) + tags (control marks)
- It is very simple: human readable, can be edited by any editor
- It reflects document presentation, not the semantics or structure of data
- Universal: portable to any platform
- HTML pages are connected through hypertext links
- HTML pages can be located using web search engines

**XML**

XML (eXtensible Markup Language) is a textual language for representing and exchanging data on the web

- It is designed to improve the functionality of the Web by providing more flexible and adaptable information identification
- Based on SGML
- It was developed around 1996
- It is called *extensible* because
  - it is not a fixed format like HTML (a single, predefined markup language)
  - it is actually a *metalanguage* (a language for describing other languages) which lets you design your own customized markup languages for limitless different types of documents

**XML (cont.)**

- XML can be untyped (semistructured), but there are standards for schema conformance
  - DTD
  - XML Schema
- Without structure, an XML document is *well-formed* if it satisfies simple syntactic constraints:
  - proper nesting of start and end tags
- With a schema, an XML document is *valid* if its structure conforms to a DTD or an XML Schema

**Example**

```xml
<people>
  <person>
    <name>Leonidas Fegaras</name>
    <tel>(817) 272-3629</tel>
    <email>fegaras@uta.edu</email>
  </person>
  <person>
    <name>Ramez Elmasri</name>
    <tel>(817) 272-2348</tel>
    <email>elmasri@uta.edu</email>
  </person>
</people>
```
**Why XML is so Popular?**

- It looks like HTML
  - simple, human-readable, easy to learn, universal
- Flexible & extensible, since you can represent any kind of data
  - unlike HTML
- HTML describes *presentation* while XML describes *content*
- Precise
  - well-formed: properly nested XML tags
  - valid: its structure may conform to a DTD or an XML Schema
- Supported by the W3C
  - trusted and adopted by industry
- Many standards around XML: schemas, query languages, etc

**What XML has to do with Databases?**

- XML is an important standardization for data representation and exchange, but we still need
  - to store and query large repositories of XML data
  - data models and schema representations
  - query languages, data indexing, query optimizers
  - updates, view maintenance
  - concurrency, distribution, security, etc
- Need both
  - databases at the server-side for storing data, and
  - the XML format for exchanging data between applications

**XML Syntax**

- XML consists of tags and text
- Text is bounded by tags. PCDATA: *parsed character data.*
  ```xml
  <title> The Big Sleep </title>
  <year> 1935 </year>
  ```
- Tags come in pairs:
  ```xml
  <date>8/25/2004</date>
  ```
- For each opening tag there must be a matching closing tag
- Tags must be properly nested:
  - valid nesting:
    ```xml
    <person> <name> ... </name> ... </person>
    ```
  - invalid nesting:
    ```xml
    <person> <name> ... </person> ... </name>
    ```

**XML Elements**

- An *element* is a segment of an XML document between an opening and the matching closing tags
  ```xml
  <person>
    <name> Ramez Elmasri </name>
    <tel> (817) 272-2348 </tel>
    <email> elmasri@cse.uta.edu </email>
  </person>
  ```
- An element may contain a mixture of sub-elements and PCDATA
  ```xml
  <title>An <em>element</em> is a segment</title>
  ```
- An abbreviation: for an element with empty content, we can use:
  ```xml
  <tagname ... />
  ```
  instead of:
  ```xml
  <tagname ... /></tagname>
  ```
Representing Data Using XML

- Nesting tags can be used to express various structures, such as a record:
  <person>
    <name>Ramez Elmasri</name>
    <tel>(817) 272-2348</tel>
    <email>elmasri@cse.uta.edu</email>
  </person>

- We can represent a list by using the same tag repeatedly:
  <addresses>
    <person>...</person>
    <person>...</person>
    <person>...</person>
    ...
  </addresses>

XML structure

XML:
<person>
  <name>Ramez Elmasri</name>
  <tel>(817) 272-2348</tel>
  <email>elmasri@cse.uta.edu</email>
</person>

in Lisp:
(person (name "Ramez Elmasri")
  (tel "(817) 272-2348")
  (email "elmasri@cse.uta.edu"))

as a tree data structure:

Referencing Elements Using IDs/IDrefs

- An opening tag may contain attributes
  - typically used to describe the content of an element
    <author ssn="2787901">
    <name>Ramez Elmasri</name>
    <email>elmasri@cse.uta.edu</email>
    </author>

- It's not always clear when to use attributes
  <author>
    <ssn>2787901</ssn>
    <name>Ramez Elmasri</name>
    <email>elmasri@cse.uta.edu</email>
  </author>

- ID attributes are special: must be unique within the document
  - An IDref attribute must refer to an existing ID in the same doc

Attributes