CMSC 330: Organization of Programming Languages

Markup & Query Languages

Other Language Types

- Markup languages
  - Set of annotations to text
- Query languages
  - Make queries to databases & information systems
- Used together in
  - Web interface to databases

Markup Languages

- Set of annotations (tags) added to text
  - Example – `<tag> text </tag>`
- Describe how text is
  - Structured, laid out, formatted...
- First used in publishing industry
  - Typesetting, proofreading
    - nroff, troff, TeX, LaTeX
    - Mostly replaced by WYSIWYG editors like MS Word
    - What you see is what you get
- Regained importance with advent of web
  - Used to describe format & presentation of web pages

History of Markup Languages

- GML (1960s)
  - Generalized markup language
  - Describe both structure & presentation of content
- HTML (1991)
  - Hypertext markup language
  - Flexible & simple descriptive markup for web pages
  - Hypertext links parts of document to other documents

History of Markup Languages (cont.)

- XML (1998)
  - Extensible markup language
  - Language for describing tags (meta-language)
  - User can create tags and describe their uses
  - Used to describe documents w/ structured information
  - No mechanism for displaying XML document

Markup Language – GML

- Example
  :.h1.Recipes:
  :p.Bread
  :ol
  :li.Flour
  :li.Yeast
  :li.Water
  :eol.
**Markup Language – HTML**

- Example
  ```html
  <html>
  <head><title>Bread Recipe</title></head>
  <body>
  <h1>Bread</h1>
  <ol>
  <li>Flour</li>
  <li>Yeast</li>
  <li>Water</li>
  </ol>
  </body>
  </html>
  ```

**Markup Language – XML**

- Example
  ```xml
  <recipe name="Bread">
  <title>Bread</title>
  <ingredient>Flour</ingredient>
  <ingredient>Yeast</ingredient>
  <ingredient>Water</ingredient>
  </recipe>
  ```

**HTML / XML Elements**

- **Element**
  - A start tag, an end tag, and data in between
  - Examples
    - `<director> Tyler Perry </director>`
    - `<actor> Tyler Perry </actor>`

- **Attribute**
  - A name-value pair separated by an equal sign (=)
  - Used to attach additional information to an element
  - Example
    - `<city ZIP="20742"> College Park </city>`

**HTML Elements**

- **Structural**
  - Describes purpose of text
  - Examples
    - `<h1> Level 1 heading </h1>`
    - `<ol> Ordered list </ol>`
    - `<ul> Unordered list </ul>`

**HTML Elements (cont.)**

- **Presentation**
  - Describes appearance of text
  - Examples
    - `<b> boldface </b>`
    - `<i> italics </i>`
    - `<p> line spacing </p>`

- **Hypertext**
  - Links part of document to other documents
  - Examples
    - `<a href="http://www.cs.umd.edu"> URL link </a>`

**XML Document**

- **An XML element with nested XML elements**
  - Example
    ```xml
    <movies>
    <movie year="2005">
    <title> Diary of a Mad Black Woman </title>
    <director> Tyler Perry </director>
    </movie>
    <movie year="2006">
    <title> Madea's Family Reunion </title>
    <director> Tyler Perry </director>
    </movie>
    </movies>
    ```
XML Documents (cont.)

- Guidelines
  - Elements must have an end tag (unlike HTML)
  - Elements must be cleanly nested
    - Overlapping elements are not allowed
  - Attribute values must be enclosed in quotation marks
  - Document must have unique first element (root node)

- Document Type Definition (DTD)
  - User can create set of rules to specify legal content
  - Place restrictions on XML file

Comparing HTML With XML

- HTML
  - Fixed set of tags
  - Presentation oriented
  - No data validation capabilities
  - Single presentation

- XML
  - Extensible set of tags
  - Content oriented
  - Standard Data infrastructure
  - Multiple output forms

Using Markup Languages

- Descriptive markup
  - Structure
    - How is this organized? (<chapter>, <section>)
  - Semantics
    - What is this? (<person>, <title>)

- Separate presentation from content
  - Keep presentation elsewhere (CSS, XSL)
  - Puts content in "delivery neutral format"
    - <h1> is a first level heading, but can be any font

Markup Language Usage

- Started with documents
- Now also used to organize
  - Metadata
    - Data about data, used to help understand / manage data
    - Example: <LastName optional="true">Smith</LastName>
  - Transactions
    - Single unit of work for application
  - Applications
    - Helping applications interact / work together

Query Languages

- Make queries to
  - Databases
  - Information systems

- Goals
  - Data retrieval
  - Data management

- Examples
  - SQL (1970s) – Query relational databases
  - LDAP (1993) – Query directory services for TCP/IP

Databases (DB)

- A structured collection of data (records)
  - Whose content can be quickly and easily
    - Accessed, managed, updated

- Database model
  - Hierarchical
    - Records are stored in a tree
  - Network
    - Records have links to other records
  - Relational
    - Records are stored in tables (relations)
Tables (Relations)
- Each column constitutes an attribute
- Each row constitutes a record or tuple

<table>
<thead>
<tr>
<th>Attribute 1 (column 1)</th>
<th>Attribute 2 (column 2)</th>
<th>Major</th>
<th>2007 Starting Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record 1 (tuple 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record 2 (tuple 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SQL (Structured Query Language)
- Queries for relational database systems
- Allows for complete
  - Table creation, deletion, editing
  - Data extraction (queries)
  - Database management & administration

SQL – Creating Database
- Types of attributes
  - char, varchar, int, decimal, date, etc.
  - varchar is a string with varying # of chars
- Not Null
  - Each record must have value
- Primary key
  - Must be unique for each record

```sql
CREATE TABLE tableName (  
  name VARCHAR(55),  
  sex CHAR(1) NOT NULL,  
  age INT(3),  
  birthdate DATE,  
  primary key(name) );
```

SQL – Creating Database (cont.)
- Primary key
  - Can use autoincremented numbers as primary key
  - Guaranteed to be unique
  - 1st entry key = 1
  - 2nd entry key = 2, etc...

```sql
CREATE TABLE tableName (  
  name VARCHAR(55),  
  sex CHAR(1) NOT NULL,  
  age INT(3),  
  birthdate DATE,  
  id INT AUTO_INCREMENT,  
  primary key(id) );
```

SQL – Inserting Values
- Identical result
- Order of fields do not matter

```sql
INSERT INTO tableName (name, sex, age)  
VALUES ('Bob', 'M', 42);
```

```sql
INSERT INTO tableName (age, name, sex)  
VALUES (42, 'Bob', 'M');
```

SQL – Updating Values
- Operations in the form
  - Select …
  - From …
  - Where …

```sql
UPDATE tableName  
SET age = '52'  
WHERE name LIKE 'Bob'
```

- Means
  - Select a column
  - From a database
  - Where x meets y condition
Database Server

- Accepts requests to access database
  - Requests in query language (e.g., SQL)

- MySQL
  - Multithreaded
  - Multiuser
  - SQL database management system (DBMS)
  - Open source
    - Free download of Community Edition

Database Web Interface

- Requires
  - Database server (MySQL)
  - Web server (Apache)
  - Method of connecting two (scripts)
    - CGI, Javascript, PHP, Ruby on Rails

PHP – PHP: Hypertext Preprocessor

- Scripting language
  - Designed to produce web pages
  - Can also be used from command line, in GUIs

- Characteristics
  - Paradigm
    - Imperative, object-oriented
  - Type system
    - Dynamic, weak
  - Application domain
    - Server side scripting

Server-side Scripting

- Steps
  1. Browser requests PHP document from server
  2. Server reads the PHP document and
    - Runs the PHP code
    - Generates HTML document
    - Returns HTML document to browser
  3. Browser displays HTML document

- Other server-side scripting languages
  - ASP.NET, JavaServer Pages, mod_perl, eRuby

PHP Documents

- PHP document
  - Filename ends in .php or .phtml
  - PHP code enclosed in (non-html) tags
    - <?php PHP code ?>
    - <script language="php"> PHP code </script>
  - Everything outside of PHP tags is unchanged
    - Usually standard HTML
  - PHP output is standard HTML document

PHP Document Example

- test.php
  <html>
  <head><title>PHP Test</title></head>
  <body>
  <?php echo '<p>Hello World</p>'; ?>
  </body>
  </html>
PHP Document Example 2

```php
<?php
    function hello() { return 'Hello'; }  
    function world() { return 'World\n'; }

    $fn1 = 'hello';
    $fn2 = 'world';
    echo $fn1() . ' ' . $fn2();
?>
```

PHP Document Example 3

```html
<?php
    <form method="post" action="email.php">
        Email: <input name="email" type="text" />
        Message:<br />
        <textarea name="message" rows="15" cols="40">
            Message:
            Email: <input name="email" type="text" />
        </textarea><br />
        <input type="submit" />
    </form>
```

PHP Document Example 3 (cont.)

```php
<?php
    function world() {  return "World!\n"; }
    function hello() {  return 'Hello'; }

    $fn2 = 'world';
    $fn1 = 'hello';

    echo $fn1() . ' ' . $fn2();
?>
```

PHP Functions

- Connect to database server
  - `mysql_connect($hostName, $userName, $password)` or die("Unable to connect to host $hostName");
- Modify database
  - `mysql_select_db($dbName)` or die("Unable to select database $dbName");
- Disconnect from database server
  - `mysql_close();`

Manage Tables Through Queries

- Basic information searches
  - `$SQL = "SELECT FirstName, LastName, DOB, Gender FROM Patients WHERE Gender = ":$Gender" ORDER BY FirstName DESC";`  
  - `$Patients = mysql_query($SQL);`
- Editing, adding, and deleting records and tables
  - `$SQL = "INSERT INTO Patients (FirstName, LastName) VALUES('$firstName', '$lastName');`  
  - `$Patients = mysql_query($SQL);`
- Potential problem…

SQL Injection

- Users may inject malicious commands to query
  - Through intentionally misformed fields
- Example
  - Query code
    - `$SQL = "SELECT ... WHERE Gender = "$Gender" ...";`  
    - `$Patients = mysql_query($SQL);`
  - User enters for Gender
    - "M; DROP TABLE Patients;" instead of "M"
  - Query becomes
    - `mysql_query("SELECT...WHERE Gender = 'M'; DROP TABLE patients;...";`  
  - Causing patient database to be deleted!
- Prevention
  - User input must be filtered / escaped / parameterized
**Ruby On Rails**

- Web application development framework
  - Written in Ruby
  - Supports web database applications
  - Uses Javascript libraries, AJAX for GUI
- Model-view-controller model
  - Used to organize web DB applications
  - Separates database from GUI
- Generates “scaffolding” code
  - Scripts generate code from specifications
  - Gets web database up and running quickly

---

**AJAX**

- Asynchronous JavaScript and XML
- Group of interrelated web development techniques
  - Used for creating interactive web application
  - Can update portions of page without browser refresh
  - Retrieves data using XMLHttpRequest from browser
- Examples
  - Google Maps
  - Gmail
  - Flickr

---

**eRuby**

- Rails uses eRuby
  - Template system to embed Ruby in text document
  - Needs interpreter to process eRuby and output html
  - Filename ends in .rhtml or .erb
- eRuby tags
  - `<%= Ruby code %>`
  - `% Ruby code`
  - `<%=%> Ruby expression %>`
  - Evaluates expression and replaces with result
  - Example: `<%= 2+3 %>` → 5

---

**Rails 2.0 Demo – Build a TODO list**

- Install Rails (or use InstantRails → Ruby+Rails+Apache+MySQL)
- Create Rails application
  - `rails todo`
    - Creates directory structure & files for todo application
  - `cd todo`
- Generate database & scaffolding
  - `ruby script/generate scaffold Todo task:string desc:text`
  - Creates model-view-controller scaffold code for todo list
  - Specifies SQL database named todo with 2 columns (task & desc)
  - `rake db:migrate`
  - Creates Table todo in database described in todo/config/database.yml
- Start built-in Rails web server
  - `ruby /script/server`

---

**eRuby Examples**

- Generate 3 list items
  ```ruby
  <% 3.times do
  %>
  <li>list item</li>
  <% end %>
  </ul>
  ```
- Alternative version
  ```ruby
  <ul>
  % 3.times do
  %>
  <li>list item</li>
  % end
  </ul>
  ```
- Return current time
  ```ruby
  <%= Time.now %>
  ```
  ```ruby
  <p>Date: <%= Time.now %></p>
  ```

---