Course Notes

EECS 485

Web Databases & Information Systems

Andrew DeOrio, H. V. Jagadish - Fall 2015

Contributors: Max Smith

Latest revision: September 21, 2015

Contents

Abstract

This course is a contemporary exploration of modern webbased information systems. It will integrate concepts from multiple computer science topics used in the design, development, and deployment of webbased applications, services, and knowledge systems. While broad in scope, it will also cover several key concepts in depth, including: web networking protocols, web databases and applications, web services, web search, webrelevant security issues, web infrastructure, and webrelevant data mining. Students will learn how to incorporate these concepts into an engineering process that includes design, analysis, development and testing, using technologies such as HTTP, XML, JavaScript, AJAX, and others.

1 Introduction and Overview

– See the syllabus.

2 Web Basics

2.1 Networking Basics

– **Circuit switched**: a dedicated channel is established for the duration of a transmission.

  – Good for real-time things like voice and teleconferences
  – Don't have to packet data
  – Better latency
  – No waittime

– **Packet switched**: netwrok in which relatively small units of data called **packets** are routed through a network for transmission.

  – Makes better use of network resources
  – Can survive destruction of a node in the network
  – Packets can get lost

2.2 Network Protocol Stack Model

Application	User interaction	HTTP, FTP, SMTP
Presentation	Data representation	XML, Cryptography
Session	DIalogue management	???
Transport	Reliable end-to-end link	TCP
Network	Routing via multiple nodes	IP
Data Link	Physical addressing	Ethernet
Physical	Metal or RF representation	802.11, Bluetooth

– IP is best-effor; therefore, packets may get dropped or delayed

– TCP is reliable because it guarantees data will get there in order

– Open System Interconnection (OSI) Reference Model

–

2.3 HTTP Structure

– Request/response protocol:

1. Client opens TCP connection to server and writes a request
2. Server response appropriately
3. COnnection is closed

– Completely stateless

  – Each request is treated as brand new

– Client requests have several possible forms:

  – GET, POST, PUT, DELETE, HEAD, TRACE, CONNECT, OPTIONS
  – Each has an associated parameter

– Even a single page can consist of dozen of HTTP requests

2.4 HTTP Client Algorithm

1. Wati for user to type into browser
2. Break the URL into host and path
3. Contact host at port 80, send GET ¡path¿ HTTP/1.1
4. Download result code and bytes
5. Send content bytes to HTML renderer for drawing onscreen

2.5 HTTP Server Algorithm

1. HTTP server process (or thread) waits for connection from client
2. Receives a GET /index.html request
3. Looks in content directory, computs name /content/index.html
4. Loads file from disk
5. Write resposne to clinet: 200 OK, followed by bytes for /content/index.html

2.6 URL Encoding

http://server:port/path#fragment?search

– Server name translated by DNS look-up: www.umich.edu→135.22.87.1

– Path is a file name relative to server root

– Fragment is identified at the client, ignored by server

– Search string is a general-purpose (set of) parameter(s) that the server can use as it pleases.

3 Content and Dynamism

3.1 Mark up Language

– Add tags as “mark up” to text

– Document still “primarily” text

– Mark up improves both structure and presentation

Hypertext

– Text with embedded links to other documents

– Anchor tag ¡a href = “link”¿atext¡/a¿

Escape Strings

– Some characters have special meaning in an application context

  – / or ? in URL

– Need an **escape string**

  – e.g., &lt; &euro; &amp;

3.2 XML

– No standard set of stags

– Define tags and use them

– Need to balanced like parentheses

– Tags form a heirarchy of objects

  – Document Object Model (DOM) tree

3.3 XHTML

– HTML permits unbalanced tags

– HTML cannot be derived from XML

– XHTML is a dialect of HTML that meets XML rules (proper containment)

3.4 HTML5

– Merges all that has happened over years related to HTML

  – XHTML
  – Browser-specific estensions of HTML
  – OTher use cases of broad interest

3.5 Layout - CSS

– Modern HTML seperates content form the way content looks (through CSS)

– Included via:

Scaling and Positioning

– Biggest problem in layout

– If display screen width changes, do you:

  – Keep your page fixed width any way
  – Adjust page width, and use more lines of text - increasing page length
  – Adjust path width and font size so everything scales evenly

3.6 Dynamic Content

– So far, HTTP servers are fileservers

  – And browsers are HTML renderers

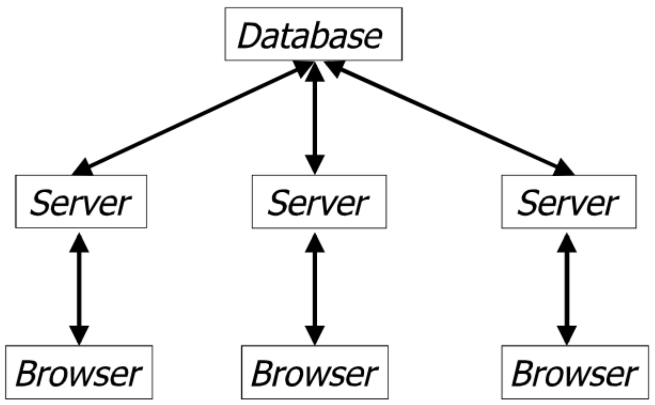
– Think of the things that are impossible with simple static pages

– In the old days, almost all requests were just disk loads

3.7 N-tier model

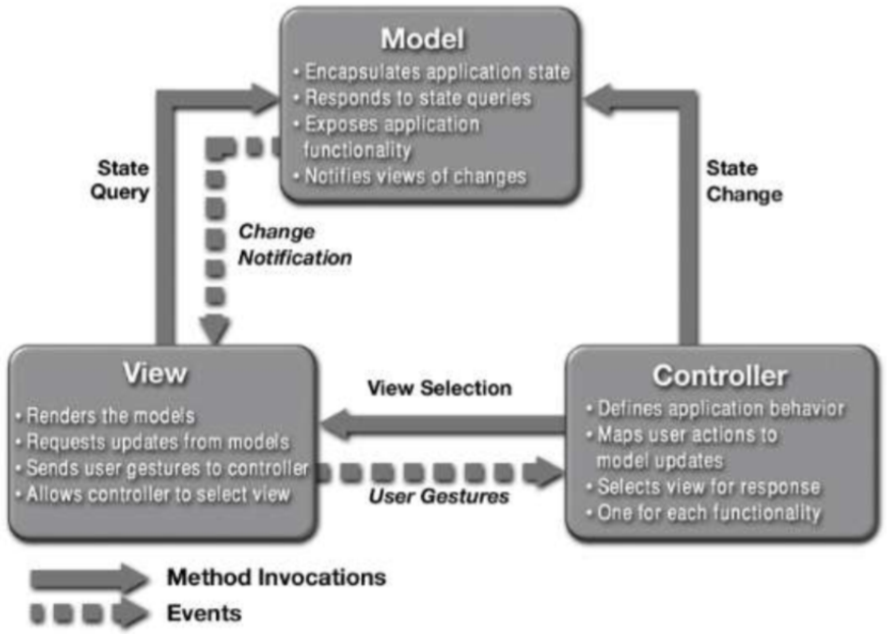
– We've seperated persistent storage from user interactions

– Web apps often break down pieces of code functionality into machines



Model-View-Controller

– System to sort out who does what


Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
Change Notification
State Change
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

– HTML is the model

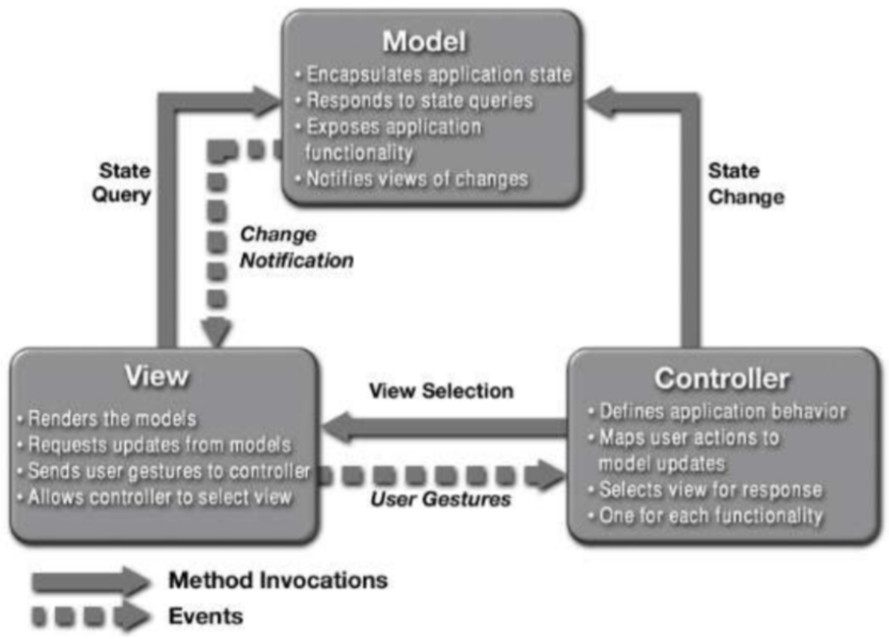
– CSS is the view

– Browser is the controller

– Model exists in the database/filesystem

– View is the computed HTML++

– Controller is processing of user input


Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
Change Notification
State Change
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

3.8 Object-Relational Mapping

– Addresses the "impedance mismatch" between relational and object worlds

– Map object classes to database tables

3.9 Dynamic Client

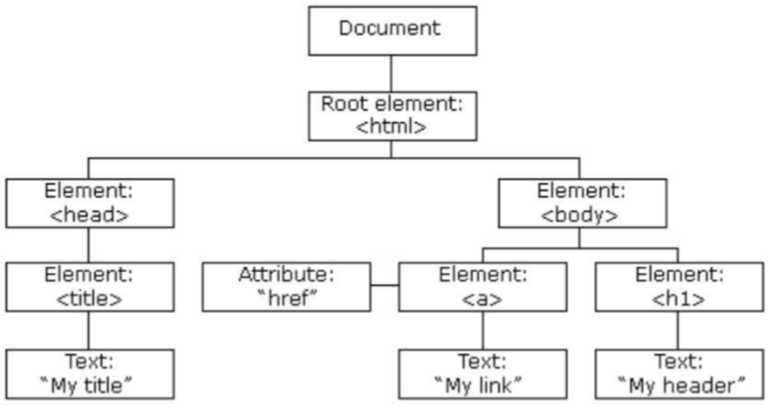
– Lots of different options for client-side content, but most people nowadays conerging on AJAX (Asynchronous JavaScript and XML)

  – JavaScript for running in the browser
  – XML for data exchange with server, via HTTP
  – After intiial page load, similar to client/server program, not traditional web page loads

– JS code runs when triggered by

– Has read/write access to the page's DOM

– e.g.,



– Client-side code will display code instead of end result. Server-side code would only show result.

3.10 Browser Security

– JS is potential security nightmare, so in-browser programs are sandboxed

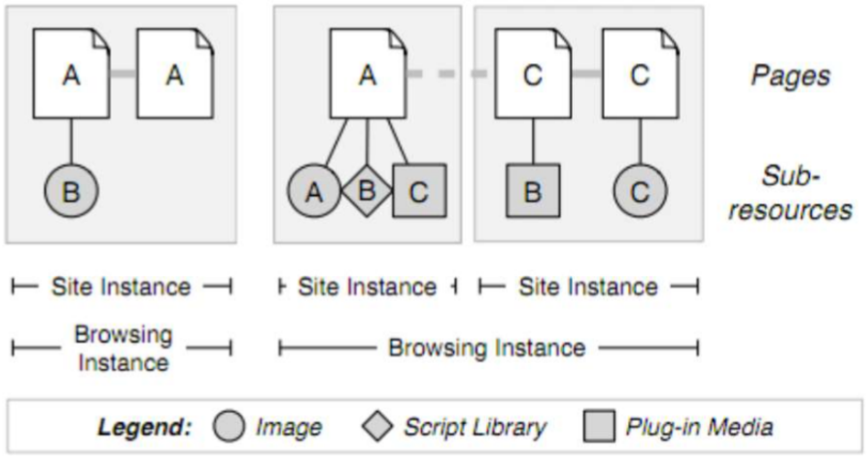
  – **Same-origin policy**: scripts from the same origin can access each others' data + methods
  – scripts form different origins cannot see each other

Attacks

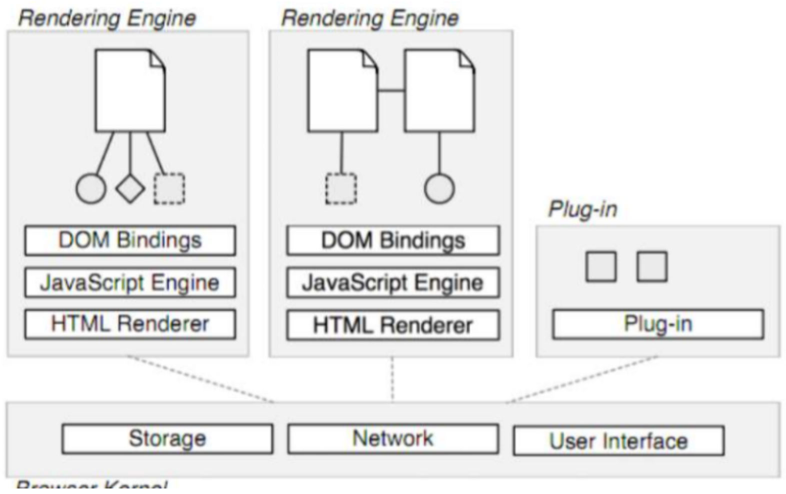
– Cross-site scripting

– SQL Injection

3.11 Browser Internals



– Chome added more break-down of functionality allowign you to kill tabs-by-process


Rendering Engine
Rendering Engine
DOM Bindings
JavaScript Engine
HTML Renderer
DOM Bindings
JavaScript Engine
HTML Renderer
Plug-in
Plug-in
Storage
Network
User Interface