Outline for Dynamic Content Generation Chapter

Stephen Chen
Paul Reitsma
David Tucker

March 13, 2001

1 Introduction (psar)

1.1 What is dynamic content?

1.2 Why is dynamic content important?

1. Critical for some functionality
   (a) Customization (My Yahoo)
   (b) Serving time-dependent data (stock ticker)
   (c) Retrieving data (Google search results)
2. Ubiquitous
   (a) More than 80% of the served web
   (b) 400-500 times the data available dynamically as statically

1.3 What are the types of dynamic content?

1. Three Types
   (a) Temporal Dynamism
   (b) Client-Side Dynamism
   (c) Context Dynamism
2. Later is most complex (underlying page changes)
3. Contrast to static pages

2 The Issues (psar & dbt)

2.1 Efficiency

1. static pages takes 2-10 CPU milliseconds; context dynamism may take orders of magnitude more
2. server processing
   (a) running programs
   (b) disk accesses
   (c) forking processes
3. download time
   (a) size of data
4. client processing
   (a) starting up virtual machines
   (b) running programs
   (c) additional network connections

2.2 Where the work is done

1. server-side
   (a) no special support needed by client
   (b) possibly high cost for server computation
   (c) don't need to send sensitive data to client
2. client-side
   (a) client needs to support new languages
   (b) server isn't burdened by computation
   (c) facilitates interaction
   (d) not all dynamism can be client-side

2.3 Language

1. expressiveness: how much content can we make dynamic?
   (a) any data whatsoever
   (b) only HTML documents
   (c) only data such as last-modified
2. suitability / simplicity

   (a) handling user input
   (b) support for content generation
   (c) support for interaction

3. robustness

4. efficiency

2.4 Security

1. server hacks

   (a) invalid input strings
   (b) buffer overflow attacks
   (c) denial of service

2. client hacks

   (a) read/write local filesystem
   (b) open new network connections
   (c) resource (CPU, disk, network) DOS

3 Tools (swc & dbt)

3.1 CGI

1. A protocol that can be used to communicate between Web forms and your program.

2. Many languages have library support for CGI, the most popular being Perl.

3.2 SHTML

1. Special tags in HTML are parsed by the server and replaced with dynamic content.

2. Slow: server has to parse the HTML document.

3. Limited form of dynamism.

4. Mostly secure.

3.3 Embedded scripting languages

1. PHP
   - (a) Perl hack developed during the mid-90s; eventually turned into a new programming language.
   - (b) Developed for embedding within HTML. Advantages:
     - i. quicker response time
     - ii. improved security
     - iii. transparency to the end user
2. Perl
3. TCL

3.4 Cookies

1. Stores hidden state on client-side
2. Sends to web server when requesting web page
3. Secure, but can be used for tracking user across sites

3.5 Java Servlets

1. Server-side
2. Runs in JVM in web server
3. Keeps state across HTTP requests
4. Slow

3.6 Javascript

1. Developed by Netscape Communications during the mid-90s.
2. An interpreted language aimed primarily at adding interactivity to websites.
3. Doesn't have GUI support. (?)
4. Doesn't have extensibility of Python, Perl, Tcl, etc.

3.7 Java Applets

1. Client-side
2. Uses Java, a full programming language
3. Takes a long time to download, start up JVM

3.8 Commercial tools

1. ASP
2. JSP
3. ColdFusion

4 How well do current tools support the different uses? (swc & dbt)

4.1 Temporal Dynamism (think: stock ticker)

1. server-side
   (a) can push update to client every n seconds
   (b) caching issues
2. client-side
   (a) real-time data
   (b) usually applets: heavyweight solution
   (c) frequent network accesses

4.2 Customization (think: amazon)

1. identifying the client
   (a) cookies are fairly accurate
   (b) user might have to enter name/password
2. accessing data
   (a) database integration

4.3 Interaction (think: multi-page survey)

1. server-side
   (a) very poor support
   (b) have to save state between web pages
   (c) ”back” button can create inconsistencies
2. client-side
   (a) JavaScript has limited capabilities
   (b) Java applets are not often used

5 Caching (psar)

1. Generating dynamic pages is expensive; may involve:
   (a) CPU time
   (b) Expensive IO
   (c) Additional network traffic
2. Wish to avoid generating dynamic pages more than is necessary
   (a) Aggressively propagate generated pages to caches
3. Only useful when a dynamic page will be seen many times
   (a) "Pseudo-dynamic"
   (b) Dynamic composition of stored data
   (c) Locality in request stream
4. Truly dynamic pages (single-use) should not be cached

6 New Horizons (dbt)

6.1 `<bigwig>`

1. an extensible programming language
   (a) strongly typed
   (b) syntactic macros
2. provides language support for interactive web services
   (a) notion of a session
   (b) sessions represented as threads in web server
   (c) HTML documents are first-class values
   (d) form field validation
   (e) database integration
   (f) security analysis
3. compiles to existing technologies
   (a) on the server: CGI, HTTP authentication
   (b) on the client: HTML, JavaScript, applets

6.2 Continuation-based web servers

[to be filled in...]

6.3 XML

1. separates data and presentation
   (a) data represented as XML document
   (b) presentation specified via XSL or other stylesheets
2. different implementation strategies
   (a) combine data/presentation on server-side, produce XHTML
   (b) send data and/or presentation to client
3. improves efficiency of dynamic data
   (a) if either data or presentation are static:
   (b) if server does processing, don’t need to regenerate
   (c) if client does processing, don’t need to resend

6.4 Zope

[to be filled in...]

7 Conclusions