

# Web Systems: extending the basic system

Info 341

1

---

---

---

---

---

---

---

---

---

---

## Quick Questions...

- What is a proxy?
  - Why are proxies useful?
  
- What is a proxy cache?
  - Why is a proxy cache useful?
  
- By the start of the web boom (circa 1994) what was the most popular server?
  - Who wrote that server?

2

---

---

---

---

---

---

---

---

---

---

## Objectives

- What are ways to extend the web system?
- What is WebDAV?
- What are some important features of WebDAV?
- What is TLS?
- How does TLS establish a secure connection?
- What is CGI?
- How does a server communicate with CGI code?
- What are some limitations of CGI?
- What are two methods of resolving limitations of CGI?
- What is ASP and what language is embedded in ASP pages?

3

---

---

---

---

---

---

---

---

---

---

## [ The Web System ]

- Protocols & Standards
  - HTTP, WebDAV, SSL
  - URL, URI, URN
  - HTML, XHTML, DHTML, CSS
- Software
  - Web servers - Apache, IIS
    - Server-side code
      - CGI, PHP, JSP, ASP, Apache modules
  - Web clients - IE, Netscape, Mozilla, Opera
    - Client-side code
      - JavaScript, DHTML, Java, ActiveX, Flash/ActionScript

4

---

---

---

---

---

---

---

---

---

---

## [ How do we extend the 'Web' system? ]

- What do we do to support new capabilities, new functionality?

5

---

---

---

---

---

---

---

---

---

---

## [ How do we extend the 'Web' system? ]

- What do we do to support new capabilities, new functionality?
  - Extend the protocol(s)
  - Extend the type of content

6

---

---

---

---

---

---

---

---

---

---

## Examples of extension

- Consider two examples
  - Extend the protocol(s)
    - WebDAV
    - Transport Layer Security (TLS)
  - Extend the type of content
    - CGI
    - PHP, ASP, JSP

7

---

---

---

---

---

---

---

---

---

---

## Protocol Extension

- Two examples
- Extending Up
  - WebDAV takes HTTP and adds functionality through additional commands
- Extending Down
  - TLS (SSL) takes insecure connections and defines a mechanism for creating secure connections
  - This is really done by creating a Transport Layer protocol, but the result is to solve a problem at the Application Layer

8

---

---

---

---

---

---

---

---

---

---

## WebDAV

- Web based Distributed Authoring and Versioning (WebDAV)
- rfc2518 - February 1999 (Goland, Whitehead, Faizi, Carter, Jensen) HTTP Extensions for Distributed Authoring
- Really only WebDA
  - Versioning had to be dropped out because it is such a complex problem

9

---

---

---

---

---

---

---

---

---

---

## [ Why does WebDAV matter? ]

- Solves critical problems with web based collaboration
- Microsoft adopted WebDAV for all its applications

---

---

---

---

---

---

---

---

---

---

10

## [ Important Issues ]

- Distributed, asynchronous, authoring
- Collections
- Locking shared resources
- Properties (metadata)
- WebDAV adopts XML for protocol

---

---

---

---

---

---

---

---

---

---

11

## [ Distributed Authoring ]

- A problem with the original Web
  - Berner-Lee's original browser could both edit and view pages
- What happens when many people are reading and writing the same file at the same time?

---

---

---

---

---

---

---

---

---

---

12

## [ Collections ]

- Specification for interpreting a URL as groups of files, collections
- <http://foo.com/files/>
  - <http://foo.com/files/A>
  - <http://foo.com/files/B>
  - <http://foo.com/files/C>
- Collections can be thought of as similar to a directory, but they do not have to be file system dependent - they could come from a database

---

---

---

---

---

---

---

---

---

---

13

## [ Locking ]

- What is a lock?

---

---

---

---

---

---

---

---

---

---

14

## [ Locking ]

- What is a lock?
  - A way of enforcing long term mutual exclusion
  - A way of resolving race conditions when two people want to edit and save the same file

---

---

---

---

---

---

---

---

---

---

15

## [ Locking ]

- What is a lock?
  - A way of enforcing long term mutual exclusion
  - A way of resolving race conditions when two people want to edit and save the same file
- In WebDAV a lock is implemented as a special property of a resource

16

---

---

---

---

---

---

---

---

---

---

## [ Locking in WebDAV ]

- Two types of locks
  - Exclusive
    - The ability to write the resource is granted exclusively to the lock owner
  - Shared
    - Allows multiple people to have a lock on the resource
    - People with the appropriate level of trust can write the resource as necessary
    - The intent is to make it clear who else is working on the given resource
    - Assumption is that the people will coordinate through some other means (phone, chat system, ...)

17

---

---

---

---

---

---

---

---

---

---

## [ Properties ]

- WebDAV recognized the need for attaching properties to web resources
  - Properties was just the term to indicate the use of metadata
- Why attach properties to the resource?

18

---

---

---

---

---

---

---

---

---

---

## [ Properties ]

- Why attach properties to the resource?
- Where are properties currently stored in the web model?
  - Where do you store metadata when you write a web page?
- Why is this a problem?

---

---

---

---

---

---

---

---

---

---

19

## [ WebDAV Properties ]

- creationdate
- displayname
- getcontentlanguage
- getcontentlength
- getcontenttype
- gettag
- getlastmodified
- lockdiscovery
- resourcetype
- source
- supportedlock
  
- WebDAV relied on the work in the Dublin Core metadata model for picking what would be the default set of properties

---

---

---

---

---

---

---

---

---

---

20

## [ WebDAV an HTTP Extension ]

- Supporting distributed authoring
  - One option would have been to write a completely new protocol
  - Another option is to extend an existing protocol
- What are the problems with extending an existing protocol?

---

---

---

---

---

---

---

---

---

---

21

## [ Extending HTTP ]

- WebDAV further specifies
  - GET
  - HEAD
  - POST
  - DELETE
  - PUT

---

---

---

---

---

---

---

---

---

---

22

## [ Extending HTTP ]

- Why is there a problem for
  - GET, HEAD, POST, DELETE, PUT?

---

---

---

---

---

---

---

---

---

---

23

## [ Extending HTTP ]

- Why is there a problem for
  - GET, HEAD, POST, DELETE, PUT?
  - How do these HTTP methods handle collections?
  - How do these HTTP methods handle properties?

---

---

---

---

---

---

---

---

---

---

24

## WebDAV: Existing Method Clarification

- GET
  - GET is well defined in http, the spec notes that where get is applied to a collection the server might or might not return something that is human readable
- HEAD
  - Head is also well defined, since it is defined in terms of GET
- POST
  - The semantics for POST are a problem. POST is often server dependent. In WebDAV they do not define it further.

25

---

---

---

---

---

---

---

---

---

---

## WebDAV: Existing Method Clarification

- DELETE
  - DELETE for non-collections
    - Largely the same as in HTTP
  - DELETE for collections
    - Must delete the collection and all the collection members (recursive delete)
- PUT
  - PUT for non-collections
    - Largely the same as in HTTP
  - PUT for collections
    - PUT can potentially result in the creation or deletion & recreation of a collection. In these cases PUT should fail.
    - When PUT creates a non-collection entity as a member in a collection than all ancestors must exist prior to the PUT or the PUT should fail.

26

---

---

---

---

---

---

---

---

---

---

## WebDAV Adds Methods

- PROPFIND
- PROPPATCH
- MKCOL
- COPY
- MOVE
- LOCK
- UNLOCK

27

---

---

---

---

---

---

---

---

---

---

## WebDAV: HTTP Extensions

- PROPFIND
  - Retrieve the properties stored for the indicated Request-URI
  - Can be used to find a specific property value
  - Can use 'propname' to find names of existing properties or 'allprop' to find all names and values
- PROPPATCH
  - Set or remove properties stored for the specified Request-URI

28

---

---

---

---

---

---

---

---

---

---

## WebDAV: HTTP Extensions

- MKCOL
  - Create the collection specified by the Request-URI
  - All ancestors in the Request-URI must exist prior to creation or the request must fail
- COPY
  - Create a complete copy of the URI specified in the Request-URI at the specified Destination
  - The Destination header element must be included in the request header
  - COPY on properties must create an exact copy of all live properties
  - There are some issues when copying a collection that are further specified by copy definition

29

---

---

---

---

---

---

---

---

---

---

## WebDAV: HTTP Extensions

- MOVE
  - For non-collections, MOVE is the equivalent of a COPY followed by a DELETE on the same Request-URI
  - MOVE for properties must implement the same semantics for the COPY operation
  - For collections, MOVE must copy the entire collection recursively

30

---

---

---

---

---

---

---

---

---

---

## WebDAV: HTTP Extensions

- LOCK
  - Requests a lock on the Request-URI
  - Locks are granted for specific entities and/or collections
  - When a lock is granted it applies to the entity and all of the properties of that entity
  - Locks include a timeout, a time when the lock will expire
- UNLOCK
  - Releases the lock on the specified Request-URI

31

## Additional WebDAV Issues

- What happens when a person already has a lock and another lock is requested?
- WebDAV specifies several new response codes

32

## Transport Layer Security (TLS)

- The TLS Protocol 1.0
- Designed based on SSL 3.0
  - SSL 3.0 is owned by Netscape and is patented
- rfc2246 - January 1999 (Dierks & Allen)  
The TLS Protocol Version 1.0

33

## Why does TLS matter?

- Internet & TCP connections are inherently insecure
- Transport Layer Security facilitates open, generalized security for ecommerce on the Internet

34

---

---

---

---

---

---

---

---

---

---

## TLS Basics

- Two parts
  - TLS Record Protocol
    - Used to encapsulate other protocols
    - Reliable & private message delivery
  - TLS Handshake Protocol
    - Establish peer identity (authentication)
    - Negotiate a shared secret that will be used to encrypt a connection

35

---

---

---

---

---

---

---

---

---

---

## TLS Record Protocol

- Messages may include fields for length, description, and content
- Sender
  - accepts message from higher level
  - fragments message into blocks
  - compresses each block
  - encrypts, and transmits the result
- Receiver
  - decrypts, verifies
  - decompresses
  - reassembles
  - Delivers message to higher level

36

---

---

---

---

---

---

---

---

---

---

# TL S Handshake Protocol

- The TLS Handshake Protocol involves the following steps:
  - Exchange hello messages to agree on algorithms, exchange random values, and check for session resumption.
  - Exchange the necessary cryptographic parameters to allow the client and server to agree on a premaster secret.
  - Exchange certificates and cryptographic information to allow the client and server to authenticate themselves.
  - Generate a master secret from the premaster secret and exchanged random values.
  - Provide security parameters to the record layer.
  - Allow the client and server to verify that their peer has calculated the same security parameters and that the handshake occurred without tampering by an attacker.

37

---

---

---

---

---

---

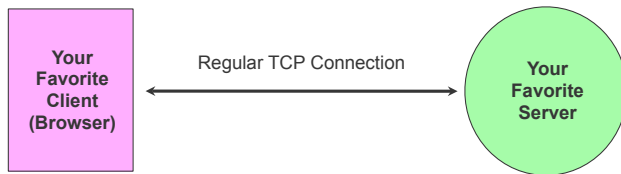
---

---

---

---

# How does this really work?



38

---

---

---

---

---

---

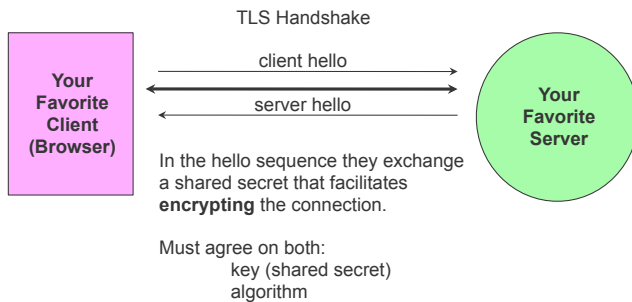
---

---

---

---

# How does this really work?



39

---

---

---

---

---

---

---

---

---

---

## How does this really work?



40

---

---

---

---

---

---

---

---

---

---

## Quick Overview

- Historical background
- Web as a system
- Protocols
  - HTTP
    - Software that implements the protocol, browsers, servers, robots, web crawlers, proxies
  - WebDAV, STL
    - Extensions to the protocol
- Extensions through active content
  - CGI ...

41

---

---

---

---

---

---

---

---

---

---

## CGI: Common Gateway Interface

- CGI was a specific feature of the NCSA web server
  - The CERN server did not originally support it.
- CGI is not an RFC
  - CGI can be implemented differently on different servers

42

---

---

---

---

---

---

---

---

---

---

## [ What does CGI do? ]

- CGI is a mechanism that allows a web server to create dynamic content by using code
  - The Request-URI is treated as code
- How does it know that the Request-URI should be executed?

43

---

---

---

---

---

---

---

---

---

---

## [ What does CGI do? ]

- How does it know that the Request-URI should be executed?
  - Put the script/code in a special place so that the incoming Request-URI can be identified
  - Special place: /cgi-bin/

44

---

---

---

---

---

---

---

---

---

---

## [ CGI APIs ]

- CGI is not a Standard
  - There is no RFC (but there have been attempts)
  - CGI can be implemented differently on different servers
  - Different environment variables can be exposed
  - Different mechanisms for returning the content
- There are several different ways to make 'CGI' work
  - NSAPI (Netscape Server API)
  - ISAPI (Internet Server API, Microsoft)
  - SAPI (Spyglass Server)
  - Apache API

45

---

---

---

---

---

---

---

---

---

---

## [ CGI Assumptions ]

- Original Assumptions
  - Coding/Scripting is somewhat specialized and probably not something every user does
  - Webmasters will have control over the code
  - Security can be maintained by the Webmaster
  - Server is running on a Unix OS ...

---

---

---

---

---

---

---

---

---

---

46

## [ CGI: getting information into the script or code ]

- Making CGI work
  - Communication to the CGI code is performed through environment variables
    - What are environment variables?

---

---

---

---

---

---

---

---

---

---

47

## [ CGI: getting information into the script or code ]

- Making CGI work
  - Communication to the CGI code is performed through environment variables
    - What are environment variables?
      - These are parameters or characteristics of the shell/ command line environment
  - Standard input
    - Request message body
  - Standard output
    - Response - body or header and body in many cases

---

---

---

---

---

---

---

---

---

---

48

## [ CGI: environment variables ]

- SERVER\_SOFTWARE
- SERVER\_NAME
- GATEWAY\_INTERFACE
- SERVER\_PROTOCOL
- SERVER\_PORT
- REQUEST\_METHOD
- PATH\_INFO
- PATH\_TRANSLATED
- SCRIPT\_NAME
- QUERY\_STRING
- REMOTE\_HOST
- REMOTE\_ADDR
- AUTH\_TYPE
- REMOTE\_USER
- REMOTE\_IDENT
- CONTENT\_TYPE
- CONTENT\_LENGTH

49

---

---

---

---

---

---

---

---

---

---

## [ Configuring CGI in Apache ]

- Example
  - Configuring httpd.conf
    - ScriptAlias
    - <Directory ...>
  - Setting up scripts
    - chmod a+x <scriptname>
  - The Request-URI (the URL)

50

---

---

---

---

---

---

---

---

---

---

## [ CGI: Limitations ]

- CGI is good, but it has some limitations
  - Must generate a full page each execution
  - CGI code executes in the same 'user' space as the web server (security issues)
  - No easy, general solution if all users want to write CGI

51

---

---

---

---

---

---

---

---

---

---

## [ CGI: Limitations ]

- Attempts to fix some CGI limitations
  - Apache can use URL (mod\_rewrite) matching to allow each user to have a /cgi-bin/ directory
    - Administrative headaches
  - Special script (CGIWrap) that wrappers a CGI call so that users writing their own CGI code only get limited permissions
    - Different administrative headaches
- None of these solve some basic problems with CGI

52

---

---

---

---

---

---

---

---

---

---

## [ Partial Dynamic Content ]

- Basic idea
  - Most dynamic pages are a bunch of static content (formatting) that surrounds some dynamic data
  - Instead of embedding the 'page' into the 'code' (like CGI) embed the code into the page
  - Interpret these pages that have special name extensions (ones that are not just .html or .htm).

53

---

---

---

---

---

---

---

---

---

---

## [ Embedded Scripting Languages ]

- Server-Side scripting languages
  - ASP - Active Server Pages
  - JSP - Java Server Pages
  - PHP - PHP Hypertext Preprocessor

54

---

---

---

---

---

---

---

---

---

---

## [ ASP ]

- Microsoft's embedded scripting approach
- Embed Visual Basic programming constructs in HTML pages
- Tightly integrated with IIS
- Easy to get going
- Proprietary

55

---

---

---

---

---

---

---

---

---

---

## [ JSP ]

- Sun Microsystem's embedded scripting approach
- Embed Java code in the HTML page
- Multiple implementations
  - Tomcat seems to be winning
  - Reasonable integration with Apache
- More complex to set up

56

---

---

---

---

---

---

---

---

---

---

## [ PHP ]

- PHP Hypertext Preprocessor
- The Perl community's embedded scripting approach
- Perl and Perl like commands are embedded into HTML pages
- Reasonable integration with both IIS and Apache
- Many, many external sources for help and code
- Thousands of diehard programmers work on PHP and Perl
- Very robust system, multi-platform

57

---

---

---

---

---

---

---

---

---

---