WEB SERVERS AND DYNAMIC CONTENT pdf

1: Dynamic Content

By combining server-side scripts with scripts that run in the browser, websites can load dynamic content over an already-loaded web page without the user being aware of it. Conclusion The current generation of web technology is built around dynamic content.

Confused about what people mean when they talk about "static" websites vs. A static website is a vending machine. A dynamic website is a restaurant. How you get your food the pageview When you look at a URL, like http: This is what people talk about when they talk about web servers: A server might be a plain old computer like the one on your desk, or it might be a rented computer in a giant data center that looks like something from WarGames. The vending machine static sites When people talk about a static server, they are talking about a server set up like a vending machine. When you ask for a URL, like http: Talking about "flat files" refers to the same thing: You punch E6, you get the Cheetos that are already in the machine, packaged, ready to eat. Like a vending machine, a static server can be: A static server has to find the file you asked for, and send back a copy. No kitchen fires, no flaky employees. Static servers rarely break. You get your order right away. Static servers are very fast. Fewer long spinning wheels waiting for content to be rendered and databases to do databasey things. Cheap - Servers cost money. But serving files statically is especially cheap, thanks to services like Amazon S3. You can serve a million pageviews for pocket change. This is a pain in the ass, because you have to go around to all these vending machines and restock the Cheetos. You have to replace them all. You only have to fix the problem in one central place. Impersonal - A vending machine has no idea who you are. It gives everyone the same food. If you want to serve people different content based on user input, this can be harder to do on a static site. Hold the mayo, medium rare, whatever you want. The restaurant dynamic sites When people talk about a dynamic server, they are talking about a server set up like a restaurant. The food is made-to-order. The kitchen has ingredients, and the cooks assemble those ingredients into the finished product only AFTER someone orders that dish. In this version, if you ask for a URL, like http: There is no such file. The server waits for your request, and when the request comes in, it uses ingredients e. When people talk about the "back end" or "server side," they are talking about the kitchen: Like a restaurant, a dynamic server can be: More personalized - Because content is being created dynamically, in response to requests, you have an opportunity to craft each one for that user. Easier to change the menu - When you need to change something that affects lots of web content, it can be easier to do this on a dynamic server. You typically only have to make the change in a central place, like your database or your template. If your site is static, you need a way to update every affected file, which is certainly possible but requires a more elaborate process and, if enough files are involved, a significant amount of time. More complex - Restaurant kitchens are frenetic, chaotic places, full of stuff that can break. There are many more ways for dynamic servers to crash or fail insidiously. Slower - It takes longer to get your dinner when it has to be made on-the-fly. Similarly, it can be slower to serve a piece of content dynamically rather than using a static file. The server has more work to do before it can send it back. Dynamic servers are vulnerable to heavy loads, when lots of people are all trying to get content at the same time think election night, or big sporting events. When you have lots of users all demanding content at the same time, serving content gets slower and slower, until the server crashes entirely. This is how many denial-of-service attacks work. They intentionally make a site too popular, flooding it with requests for content, until it buckles under the sheer volume of simultaneous requests. Prep cooks a little bit of both In the real world, most servers mix these two approaches. They have prep cooks organizing, chopping, and staging ingredients ahead of time to minimize how much of the process is dynamic. That image is probably just an actual file that already sits on the server. Part of managing dynamic servers is about having a good mise en place: Some caveats Like all analogies, this one is limited. Dynamic servers can use various kinds of caching to serve pre-cooked content but update it dynamically. There are not many absolutes. Most things that are easier on dynamic servers, like personalization and updating content in a central place, are possible on a static server. It just tends to be trickier. You have to figure out workarounds to make things seem dynamic, and have systems in place to regenerate your files in bulk when you need to make updates. You can create extremely fast and robust dynamic servers that will stand up to any amount of traffic. You have to have the sysadmin savvy to use the right tools and employ the right strategies.

2: dynamic content in other web servers - Remote Control, Monitoring and the Internet - LAVA

dispatching dynamic content plugins -- think a web server that does on-the-fly watermarking or image transcoding. I'm looking for the fastest, lowest-overhead way of dispatching this. Hmnn, a custom extension module to nginx is the lowest overhead option.

You can find out about how to enable dynamic compression for the IIS web-server by consulting the official documentation which is located here: You can also read about the benefits of using dynamic compression in this following blog entry: This article means to discuss the way that dynamic compression can be configured in the IIS web-server, version 7 and above, as well as some new features that have been introduced as of IIS 10 and Windows Server IIS Dynamic Compression configuration: Dynamic compression is a feature that allows the IIS web-server to compress responses coming from such handlers as the ASP. For more on handlers and the integrated pipeline, see the video series of IIS Architecture and Components: Dynamic compression addresses content that is generated dynamically, hence the name, as opposed to static compression, which deals with files that are read from disk and sent across the network to the requesting client think of PDFs, images, javascript files and CSS files as examples. Contrary to static compression, dynamic compression has to be done on each outgoing response, since there is very little chance that two dynamic responses from a web-application will be the same and the result of the compression of response A will be reused in response B. By default, the dynamic compression module is not installed on the IIS server if you are just doing a standard install via the Server Manager and choosing to install the web-server role. The configuration for this feature is done at two distinct levels on the IIS web-server. At the server level we can configure which types of content dynamic compression is enabled for, and what are the cut-off values in terms of CPU consumption for stopping compression from occurring on outgoing responses and resuming it. One very good thing about the Configuration Editor is that it allows you to inspect IIS configuration settings that are available and apply to the location in the tree view server vs site vs web-application where you are located. Configuration sections that are not present at the level you have selected cannot be configured at that level, even if you manually try to enter them in a web. From the listing above, we can see several dynamicCopmpression parameters that are of interest: This is an important change from Windows and IIS 6, where a server administrator had to list the extensions of all request urls for which the server should be performing static and compression. Note that we can enable or disable compression for each of the listed types as desired. The configuration section that contains these settings is the system. The two configuration values that can be used are: It will disable or disable the compression for the MIME types that are listed in the system. If a dynamic response is determined by IIS to be cachable will be stored in cache for the next requests , the question becomes: If the doDynamicCompressionBeforeCache is enabled, and the doDynamicCompression is also enabled, and the response is determined to be cachable, the compression will take place before the response is placed in cache. Hence, IIS will store the cached copy of the response. If doDynamicCompression is turned on but doDynamicCompressionBeforeCache is turned off, the response stored in cache will not be compressed. A second request that comes in for the same content on the IIS server will retrieve the content from cache, and the content will have to be recompressed before it is sent to the requesting client. Turning this setting on can greatly contribute to diminishing CPU consumption in some cases. Determine if dynamic compression is working for your content: You can find out more about FREB by looking at thus getting started guide here: To determine if dynamic compression is working, consider configuring FREB tracing to trace requests you are interested in: As you can see, the response has been compressed from an initial response size of bytes down to a response size of bytes. New features in Windows and IIS In IIS 10 the configuration system of the web-server has been extended to allow the definition of some of the server level configuration sections at the site and web application level as well. We can see this very easily by using the same Configuration Editor to inspect the configuration at site or web-application level. You will note that there are six dynamicTypes defined, since I have defined an extra MIME type at the level of my website. This allows a site administrator to define or override the definitions of the dynamic and static compression MIME types that were defined at a server level. Hopefully, this will be one more reason for everyone using IIS to consider moving to the version 10 of the web-server.

3: httpd - fastest web server for static, dynamic content? - Server Fault

When web servers first appeared their primary purpose was to serve up selected information from the machine on which they ran. The idea was to simply take the contents of a file and transmit them over a TCP connection in HTTP format. The inherent limitation discovered early on was that dynamic.

This attribute ensures that older clients and proxy servers do not attempt to cache compressed files. To enable this setting, you must set the sendCacheHeaders attribute to true. The WWW service must be restarted before changes to this property take effect. Specifies the directory where compressed versions of static files are temporarily stored and cached. Specifies whether a limit exists for the amount of disk space that all compressed files, which are stored in the compression directory specified by the directory attribute, can occupy. The default value is true. Specifies the maximum amount of dynamically compressed data that IIS will buffer before flushing the buffer to a client. This decreases the amount of memory that is necessary to perform dynamic compression. This attribute was added in IIS 7. The default value is Specifies the percentage of CPU utilization at which dynamic compression will be disabled. This attribute acts as an upper CPU limit at which dynamic compression is turned off. When CPU utilization falls below the value specified in the dynamicCompressionEnableCpuUsage attribute, dynamic compression will be re-enabled. Specifies the percentage of CPU utilization below which dynamic compression will be enabled. The value must be between 0 and Average CPU utilization is calculated every 30 seconds. This attribute acts as a lower CPU limit below which dynamic compression is turned on. When CPU utilization rises above the value specified in the dynamicCompressionDisableCpuUsage attribute, dynamic compression will be disabled. Specifies the content of the HTTP Expires header that is sent with all requested compressed files, together with the Cache-Control header specified in the cacheControlHeader attribute. This combination of headers ensures that older clients and proxy servers do not try to cache compressed files. The default value is Wed, 01 Jan Specifies the number of megabytes of disk space that compressed files can occupy in the compression directory. When the space used by compressed files exceeds 90 percent of the value in this attribute, IIS deletes the least recently used files until a percent usage level is reached. In IIS 7 the limit is applied per application pool. Specifies the minimum number of kilobytes a file must contain in order to use on-demand compression. The default value for IIS 7. Specifies whether compression is disabled for requests that contain an HTTP 1. You can use this setting to avoid returning a compressed file to a client that cannot decompress it. Specifies whether the HTTP 1. Some HTTP proxy servers do not handle the caching of compressed objects correctly. You can use this setting to avoid returning a compressed file to a proxy server that cannot decompress it. Some clients cannot handle range requests correctly. The default value is false. Specifies the percentage of CPU utilization at which static compression is disabled. This property acts as an upper CPU limit at which static compression is turned off. When CPU utilization falls below the value specified in the staticCompressionEnableCpuUsage attribute, static compression will be reenabled. Specifies the percentage of CPU utilization at which static compression is enabled. This property acts as a lower CPU limit below which static compression is turned on. When CPU utilization rises above the value specified in the staticCompressionDisableCpuUsage attribute, static compression will be disabled. If True, disables the behavior that a static file is compressed only if it is hit a certain number of times within a time period. You may encounter circumstances In which you want static content always to be compressed to lower bandwidth usage. For example, you may want to always compress static content when a system employs a load balancer with edge caching between a Web server and the requester, causing an uncompressed file to be cached at the edge server because subsequent requests would not reach the Web server. If the behavior is not disabled, a hit rate of greater than or equal to two hits in 10 seconds will result in compression of the static content. A lesser hit rate would result in the content not being compressed. The default value is False.

4: What Is a Web Server vs. an Application Server | NGINX

On the web server hosting it, there is an application server that takes article content from a database, formats it, puts it inside some HTML templates, and sends you the results. In this case, the application server is called Kuma and is built with Python (using the Django framework).

Amazon CloudFront Dynamic Content Delivery Deliver personalized, dynamic web content at no additional cost without writing new code If you are serving dynamic content such as web applications or APIs directly from an Amazon Elastic Load Balancer ELB or Amazon EC2 instances to end users on the internet, you can improve the performance, availability, and security of your content by using Amazon CloudFront as your content delivery network. With Amazon CloudFront, your end users connections are terminated at CloudFront locations closer to them, which helps in reducing the overall round trip time required to establish a connection. These CloudFront locations are connected to the highly resilient Amazon Backbone Network that provides superior performance and availability for connection to AWS origins. In addition, various other optimization such as persistent TCP connections to the origin, SSL enhancements such as Session tickets and OCSP stapling helps in improving the performance even for non-cacheable, dynamic content. Use cases for dynamic content delivery with the CDN Application development, APIs, IoT, telemetry For inbound data and API calls from devices, responsiveness and reliability of short, bursty requests can make or break real-world interactions and device behavior. In addition, the emerging world of voice assistants, smart homes, and other applications that require a low latency connection to cloud resources to process responses or actions can benefit from transaction acceleration through the CDN, especially when devices may be deployed on transient, congested, or lossy connections. The average latency around the world to Slack. It gives them flexibility to apply various security measures at the edge. Fast discovery of products via search and browse is critical. Performance improvements for applications here translate directly into revenue and end user loyalty. Extensive options for cookie and querystring handling, cache key modification , CDN and client-side cache-control allow for maximizing what content is cached, what comes directly from the origin. Advertising Targeted ads are computed on-the-fly based on cookie or query string data, and advertisers generally need low latency in serving ads. Amazon CloudFront can help meet the performance and personalization needs for such applications either by accelerating ad targeting calls from client to ad server, caching and delivering the ad creatives, or optimizing the reporting beacon calls. News, sports, local, weather Web applications of this type often have a geographic focus with customized content for end users. Content can be cached at edge locations for varying lengths of time depending on type of content. For example, hourly updates can be cached for up to an hour, while urgent alerts may only be cached for a few seconds so end users always have the most up to date information available to them. A content delivery network is a great platform for serving common types of experiences for news and weather such as articles, dynamic map tiles, overlays, forecasts, breaking news or alert tickers, and video. Earth Networks uses a CDN so that they can provide dynamic and personalized web based content quickly to their users with very low latency and high performing response times. Specifically, they need to be able to provide local information to the end user, in near real time, and need a CDN that allows them to adjust things like time to live, query strings, and cookie information so that they can pass all that information back to the origin to pull just what the user needs. From performing device detection, to caching variants by device characteristics, to working with Lambda Edge to perform image optimization, Cloudfront can improve responsiveness and save money by reducing bytes delivered while retaining visual experience.

5: Static vs. Dynamic

The act of dispatching either dynamic or static content places a certain load on a web server, with dynamic content placing a heavier load than static content -- a distinction that was described at length in the web servers section of the chapter on key technologies.

Basic concepts[edit] Classical hypertext navigation, with HTML or XHTML alone, provides "static" content, meaning that the user requests a web page and simply views the page and the information on that page. However, a web page can also provide a "live", "dynamic", or "interactive" user experience. Content text, images, form fields, etc. There are two ways to create this kind of effect: Using client-side scripting to change interface behaviors within a specific web page , in response to mouse or keyboard actions or at specified timing events. In this case the dynamic behavior occurs within the presentation. Using server-side scripting to change the supplied page source between pages, adjusting the sequence or reload of the web pages or web content supplied to the browser. Server responses may be determined by such conditions as data in a posted HTML form , parameters in the URL , the type of browser being used, the passage of time, or a database or server state. History[edit] It is difficult to be precise about "dynamic web page beginnings" or chronology, because the precise concept makes sense only after the "widespread development of web pages": It is obvious, however, that the concept of dynamically driven websites predates the internet, and in fact HTML. The introduction of JavaScript then known as LiveScript [2] enabled the production of dynamic web pages. Execusite introduced the first dynamic website solution for the professional marketplace in June Execusite was acquired by Website Pros now Web. During the bust cycle of the Dot-com bubble , the original Execusite founders bought back the company from Website Pros December Server-side scripting[edit] A dynamic web page needs a support-server, an application server to process its server-side language. A program running on a web server server-side scripting is used to generate the web content on various web pages, manage user sessions, and control workflow. Two notable exceptions are ASP. Dynamic web pages are often cached when there are few or no changes expected and the page is anticipated to receive considerable amount of web traffic that would create slow load times for the server if it had to generate the pages on the fly for each request. Client-side scripting[edit] Client-side scripting is changing interface behaviors within a specific web page in response to mouse or keyboard actions, or at specified timing events. In this case, the dynamic behavior occurs within the presentation. Client-side scripting also allows the use of remote scripting , a technique by which the DHTML page requests additional information from a server, using a hidden frame , XMLHttpRequests , or a Web service. The innerHTML property or write command can illustrate the client-side dynamic page generation: Combination technologies[edit] All of the client and server components that collectively build a dynamic web page are called a web application. Web applications manage user interactions, state, security, and performance. It is a web application development technique for dynamically interchanging content, and it sends requests to the server for data in order to do so. The server returns the requested data which is then processed by a client-side script. Google Maps is an example of a web application that uses Ajax techniques. HTTP supports uploading documents from the client back to the server. There are several HTTP methods for doing this.

6: Web servers for static content - Web application performance and scalability

The main difference between Web server and application server is that web server is meant to serve static pages e.g. HTML and CSS, while Application Server is responsible for generating dynamic content by executing server side code e.g. JSP, Servlet or EJB.

Requests for Microsoft ASP. NET Web pages or Web servicesâ€"files with extensions. Requests for files with the extension. Once the engine has successfully rendered the markup for the requested resource, this markup is returned to IIS, which then sends the markup back to the client that requested the resource. This model of serving contentâ€"having IIS directly serve only static content, while delegating the rendering of dynamic content to separate enginesâ€"has two distinct advantages: It provides a nice division of labor. IIS can focus on excelling at serving static content, and can leave the details for serving dynamic content to an external program. NET Web pages and Web services. It allows for new dynamic server-side technologies to be added to IIS in a pluggable manner. NET Web pages itself, rather than relying on an external engine. In that case, each time a new version of ASP. NET came outâ€"or any dynamic server-side technology, for that matterâ€"a new version of IIS would need to be created that supported this new version. Those who wanted to use the latest version would have to update the version of IIS. To have this model of serving content work, IIS needs a mapping of file extensions to programs. When a request comes into IIS, then, this mapping is consulted to determine where the request should be routed. NET engine performs a similar series of steps to determine how to properly render the requested file. For example, there is a different HTTP handler class in the. NET Web pages than there is for rendering Web services. NET engine relies on different classes to render the content for a certain type of content. By having the ASP. Of particular interest is the fact that this model allows for developers to create new HTTP handler classes, and plug them into the ASP. NET have some sort of directory, mapping file extensions to external programs. IIS stores this information in its metabase, which is able to be edited through the Internet Services Manager. Each provided extension maps to a specific executable path. Figure 1 shows some of the file extensions that are mapped to the ASP. An ISAPI extension is an unmanaged, compiled class that handles an incoming Web request, whose task is to generate the content for the requested resource. NET engine, however, is a set of managed classes in the. In both the machine. Finally, the type attribute specifies the type of the HTTP handler that is responsible for rendering this content. The following is a snippet of default HTTP handler assignments in the machine. The second maps all requests to Web services. This HTTP handler, then, is invoked if a user attempts to browse to a. The HttpForbiddenHandler simply emits a message indicating that files of that type are not served, as shown in Figure 2. For example, if you run a Web hosting company you could configure IIS to route requests to. NET engine map all. The mappings, however, can be customized on a Web-application by Web-application basis using the Web. In order for the ASP. NET Web application requires two steps. HttpContext instance which contains information about the request. The HttpContext class that is passed into the ProcessRequest HttpContext method exposes many of the same vital properties that the System. The IsReusable property indicates if one instance of the HTTP handler can be used for other requests to the same file type, or if each request requires an individual instance of the HTTP handler class. Sadly, there is little information in the official documentation as to what are the best practices for using IsReusable. NET Team at Microsoft, sheds some light on the subject: Allocating memory is cheap, so you only need to mark a handler as reusable if one-time initialization cost is high. This will create a new project with a default Class. In this file, add the following code your namespace might differ: We can leave the IsReusable property as-is since it returns false by default , meaning that all we have left to do is write the code for the ProcessRequest HttpContext method. Write statements, spitting out the precise HTML markup that should be sent back to the requesting Web browser. A simple way to achieve this is through Response. What About Web Controls? There are two techniques that can be employed: Create a separate ASP. The HTTP handler, then, marries the template and the dynamic content to display. The first approach is the ideal one as it provides a clean separation of code and content. NET Web page rather than mucking around with Response. Write statements in the HTTP handler. It takes a bit of work to get this technique working properly, though. The second technique involves programmatically creating instances of the Web control classes that you want to have rendered in the ProcessRequest method ofyour HTTP handler. This technique proves a bit challenging since if you want to add Web controls nested inside of other ones, you have to manually build the Web control hierarchy yourself. Once the control hierarchy has been constructed, you need to generate the HTML for the control hierarchy using the RenderControl method. The following code snippet illustrates how Web controls can be rendered programmatically in an HTTP handler: Add lbl1 ; p. Add new LiteralControl " - " ; p. WebControls namespaces will need to be included via Imports or using statements. Clearly building and rendering a control hierarchy by hand is not nearly as easy as adding Web controls by dragging and dropping, or using the declarative Web control syntax. Realize that each time an ASP. NET Web application to use the handler for a particular file extension. NET Web application project to the Solution. Right-click on the References folder and choose Add Reference. From the Add Reference dialog box, select the Projects tab and pick the Class Library project created in the previous section. If you are not using Visual Studio. We could make up our own extension, like. NET Framework is installed on a Web server the extension. Realize that this file, HelloWorld. What happens is the following: NET engine consults the Web. NET engine, then, creates an instance of this class and calls its ProcessRequest method, passing in the current HttpContext. These seven steps would proceed in the same manner had the visitor requested ASPisNeat. NET Web applications today. The remainder of this article walks through three such HTTP handlers: The handler only displays the code file if the request comes through localhost. The handler adds a watermark to the image and checks to ensure that the image is not being "lifted" from your site that is, that someone from another site is not linking to your image from their Web site. NET Web page to display information about a specific employee from an employee database. The complete source for all three handler examplesâ€”along with the source code for the ASP. NET Web application demonstrating their useâ€"is available to download from this article. NET and open the associated project? NET is usually the last thing I want to do. Ideally, it would be nice to be able to visit the code-behind class that resides on the server to view its source. That is, to see the code for the page WebForm1. Ideally, though, when moving the ASP. On your development server, though, you might want to be able to view the code-behind source code through a browser. One option would be to just remove the mapping from. This could be done for the entire development server by modifying the machine. When this works, the source code is displayed as plain text, in an unformatted manner see Figure 4. Displaying code with an HTTP handler While this definitely works, the source code display is anything but ideal. Fortunately there are free. With just a couple of lines of code, squishySyntaxHighlighter takes in a string containing Visual Basic. To accomplish this we will use an HTTP handler. This is, after all, what HTTP handlers are designed to doâ€"to render a specific type of content.

7: Dynamic web page - Wikipedia

Instead of simply delivering static content, it generates dynamic content and delivers it to the user's web browser. Application servers, like Apache Tomcat, power the interactive parts of a website and those parts that can appear differently depending on the context of the request.

This directive is used to indicate the number of threads created by each web server child process. More threads of course means an ability to handle more user requests concurrently. Apache on Window OS is only capable of creating a single child process, hence it needs to have a higher thread count to handle the entire load of the web server. This directive is used to limit the number of requests a web server child process can handle. In case the number of maximum requests is reached for a particular web server child process, the process is killed. By having a value of 0 this indicates there is no maximum number of requests. Having the entire load of a web server absorbed by a single process is a risky proposition. The StartServers directive is used to control the number of web server child processes created when Apache is started. However, bearing in mind Apache dynamically controls the number of processes depending on load, there is usually little reason to adjust this parameter. The next parameter is MaxClients, which sets the limit on the number of concurrent requests attended by Apache. In case the limit is reached, Apache simply puts the additional requests on hold until a child process becomes available. In case you decide to change this value, you must also explicitly declare and raise the value of the ServerLimit directive. You should be careful about setting either of these directives to a large value, since setting values higher than a system can handle in terms of resources, can lead to Apache not being able to start or becoming unstable. If performance concerns force you to raise these values, you should consider other alternatives like vertical or horizontal scaling. This process is described in a previous chapter: In case you decide to change this value, you must also explicitly declare and raise the value of the ServerLimit directive in combination with the ThreadsPerChild directive. In this case, changes are required in two directives because request processing is determined by ServerLimit multiplied by ThreadsPerChild. Though as in the previous case, you should be careful about modifying these directives to large values, vertical or horizontal scaling should be considered whenever possible before modifying such values. The side-bar contains additional details on why this can change. It depends on whether you require running a particular type of Apache module. Therefore, each thread can handle one request, which makes one process capable of handling multiple requests. Therefore, each process can only handle one request at a time. However, as easy as this choice sounds, there can be other circumstances influencing this choice. This situation can be common when using Apache to serve dynamic content e. By default, a series of HTTP headers are set by browsers on each request made to a web server, as well as a series of HTTP headers are set by a web server on each response with content sent to a browser. One is for modifying or adding standard HTTP headers -- like those used for caching, described in detail in the next chapter. Another scenario is adding non-standard HTTP headers for the purpose of debugging or some advanced functionality required by a web application. This module also allows you to alter, remove or add outgoing HTTP headers prior to sending static content to a client, therefore you can change the default standard HTTP headers values set by Apache for static resources. Both these HTTP headers tell a client i. This module can represent substantial savings in terms of bandwidth consumption, as well as a reduction in overall latency. As advisable as compressing static content is, there are certain limitations to applying compression by a web server. To begin with, there is the type of content that can be compressed, which is basically limited to text e. CPU and memory and have limited impact. Another factor to consider is that certain clients i. Aditionally, there is the issue of static content being of a sufficient size to merit compression, if a web server is compressing a 5 KiB or 10 KiB static content file, the CPU and memory consumption on both server and client may not be worth the process. An upcoming chapter dedicated to compression describes several of these techniques. This module has since been deprecated and is only available on the 1. This can impact performance because a request made to Apache can be fulfilled from its cache, instead of taking up resources by re-reading a file from the file system or doing some other task e. There are three storage management modules offered by Apache. It should be noted that up until version 2. Since all these modules are related to caching, more details about their usage and configuration are given in the next chapter focused on caching. This Firefox browser add-on executes several tests for improving client-side performance e. Though sources and installation instructions are readily available for all modules. In addition, most Apache OS distributions also include utilities like a2enmod and a2dismod for enabling and disabling installed modules. Is it better to compile add-on modules into Apache or do dynamically loaded modules work the same? If you can compile add-on modules into Apache you should, since they can show slightly better performance than using dynamically loaded modules or DSO. In addition, if you compile an add-on module into Apache, there is no way to deactivate if you change your mind later, the module will take up resources whether you require it or not. In the end it will come down to having the ability to build your own Apache web server, as well as being practical. For more on the advantages and disadvantages of using modules as Dynamically Shared Objects see: By default, most parameters default toward favoring peformance over functionality, but verifying the following parameters should be done when doing performance analysis: Set AllowOverride to None. If you must rely on configuration parameters in. While Apache does understand symbolic links if placed on a directory where it serves files from, for optimum performance you need to be careful with the SymLinksIfOwnerMatch parameter. Ensure HostnameLookups is Off. P address, date and user-agent i. Apache has the ability to translate these I. P addresses to domain names by setting HostnameLookups to On, something that makes identifying visitor patterns easier. Disable logging if possible. You can disable logging in Apache by removing or commenting out logging related configuration parameters e. If a request is made for a URL with no file extension e. You can avoid content negotation by using URLs with explicit file extensions to avoid scaning or further inspection of HTTP request headers. The KeepAliveTimeout parameter allows you to specify the time in seconds to keep a TCP connection open -- the default is 15 seconds. Though setting a higher KeepAliveTimeout can be beneficial, the higher this value the more web server processes are kept busy waiting for requests from idle clients. A balance needs to be reached for this value. This is a rather crude approach for a web server, since blocking requests is best done prior to them hitting a web server, using a firewall or a reverse proxy. One approach to blocking requests is on the basis of I. P addresses, you can define a list and let Apache block requests originating on these I. The first step is to define a list with the problematic I. P addresses, as illustrated in listing Listing List of blocked I. P addresses for Apache Listing illustrates this syntax. P addresses RewriteEngine on RewriteMap blockmap txt: The RewriteMap defines the reference name and location of the file containing the I. Since we only want to rewrite requests for unwanted I. P addresses, the RewriteCond is used to define a conditional for applying a rewrite rule. P addresses -- referenced through blockmap. P address, the comparison evaluates to OK. P addresses are to be rewritten. The second part of the rule http: However, you may want to think how you define this rewrite rule for other cases. You could apply the rewrite rule only to requests involving large static content e. P address are no longer being processed due to abuse. All this depends on how aggressive you want to be against unwelcomed requests. A large list of I. To create a hash file for Apache you can use the httxt2dbm utility. Executing the following instructions httxt2dbm -i blockedlist. P addresses into a hash file -- blockedlist. P addresses and no logging for requests from unwanted I. Even though the same reference name is used -- blockmap -- notice how the file declaration is preceded with dbm: The second difference is the RewriteRule value. Again the logic and syntax may seem a little contrived, but this how Apache configuration works. Another approach to blocking requests is on the basis of HTTP headers. P addresses allow you to block requests originating from a particular location, HTTP headers allow you to block requests originating from a particular group of clients. Since every request made by clients contains a series of HTTP headers, the values or lack thereof for some of these HTTP headers can also serve to block undesirable client requests. A basic example of using HTTP headers for blocking undesirable client requests are those lacking a User-Agent header value. A second example of using HTTP headers for blocking undesirable client requests are those related to hotlinking. Figure illustrating referenced content requests in a web page best demonstrates how hotlinking can be blocked. The first request made for a web page is for the bulk of its content, but subsequent requests are often made for referenced content consisting primarily of static content e.

8: Dynamic Web Server Controls and View State

Using legacy languages like C and Fortran can aid computationally complex web applications.

The author has to change or update these documents manually. As the Internet and also the demand for high-level multimedia content grew, the need for dynamic web pages arose. As this document is focused on the Apache Web Server, client-side scripting like Java Script will not be covered. Basically, web clients are designed to display pages that they received as reply to a request. Alternatively, extending the functionality of Apache to implement dynamic Web Applications would include the need to develop additional modules, which is covered later in this document. Writing a separate module for each web application allows for good performance, but can be rather complicated and expensive. As each new module would either need a server restart or even a complete recompilation of the server source code, that solution is rather inconvenient, especially in ISP environments where multiple parties share one server. Therefore most web applications are usually implemented using a scripting language which will be executed by a interpreter called via CGI or by an interpreter module. As mentioned above one of the first technologies to enable the server to provide dynamic content was CGI G. To enable scripting using CGI, the web server executes an external program, which is able to interpret scripts that return HTML code to the server which then forwards the output to the client. The module supplies an execution environment for the script and offers API functions to enable the script to access external data sources and data received from the client. Basically the document includes HTML code like a static page. Within certain tags that are recognized by the script-executing module, the developer can insert scripting instructions that will output HTML code replacing the scripting instructions. The output these instructions generate can be based on external data sources such as a database or on input received by the client with the request. One common use for SSI is to include another file into a document. See below for information on CGI. SSI commands can be included in HTML pages using special commands within comment tags, which a browser will ignore in case the server is accidentally unable to interpret the script: Therefore compiled programs or scripts are used that output complete HTML documents. Files containing scripting commands may not include static HTML, which relieves the server from having to parse the whole document for scripting instructions. Another reason for these script languages to be a lot faster than languages allowing simple HTML code in their files is that some of them can be compiled and therefore run a lot faster than scripts that have to be interpreted. Usually such technologies are a lot more powerful and therefore allow for more flexibility. Communication with external programs is easier and some even start processes that run in the background and are responsible for keeping state information. CGI is used to start external programs like directory info, e. Scripting technologies employing modules to interpret scripts overcome these limitations as scripts are interpreted in the server context. With that technology, a request arriving at the web server will trigger the execution or interpretation of the script and forward all output to the client. The complexity of a script application is usually not limited by the script language capability but by the servers performance. For very complex applications it might be worth implementing an add-on module compiled into the web server and therefore usually gains performance as no interpretation nor a context switch is needed Additionally the possibility to use the complete server API without the restrictions a interpreting module may imply allows for more powerful applications.

9: javascript - What is static content in relation to web server? - Stack Overflow

Dynamic content in the context of HTML and the World Wide Web refers to website content that constantly or regularly changes based on user interactions, timing and other parameters that determine what content is delivered to the user.

Overview[edit] The primary function of a web server is to store, process and deliver web pages to clients. Pages delivered are most frequently HTML documents , which may include images , style sheets and scripts in addition to the text content. Multiple web servers may be used for a high traffic website; here, Dell servers are installed together being used for the Wikimedia Foundation. A user agent , commonly a web browser or web crawler , initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so. While the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for submitting web forms , including uploading of files. This means that the behaviour of the web server can be scripted in separate files, while the actual server software remains unchanged. Usually, this function is used to generate HTML documents dynamically "on-the-fly" as opposed to returning static documents. The former is primarily used for retrieving or modifying information from databases. The latter is typically much faster and more easily cached but cannot deliver dynamic content. Web servers are not only used for serving the World Wide Web. They can also be found embedded in devices such as printers , routers , webcams and serving only a local network. The web server may then be used as a part of a system for monitoring or administering the device in question. This usually means that no additional software has to be installed on the client computer since only a web browser is required which now is included with most operating systems. The case label reads: The project resulted in Berners-Lee writing two programs in Consider the following URL as it would be requested by a client: The result is the local file system resource: The response will describe the content of the file and contain the file itself or an error message will return saying that the file does not exist or is unavailable. Kernel-mode and user-mode web servers[edit] A web server can be either incorporated into the OS kernel , or in user space like other regular applications. Web servers that run in user-mode have to ask the system for permission to use more memory or more CPU resources. Not only do these requests to the kernel take time, but they are not always satisfied because the system reserves resources for its own usage and has the responsibility to share hardware resources with all the other running applications. Executing in user mode can also mean useless buffer copies which are another handicap for user-mode web servers. Load limits[edit] A web server program has defined load limits, because it can handle only a limited number of concurrent client connections usually between 2 and 80,, by default between and 1, per IP address and TCP port and it can serve only a certain maximum number of requests per second RPS, also known as queries per second or QPS depending on: When a web server is near to or over its limit, it becomes unresponsive. Causes of overload[edit] At any time web servers can be overloaded due to: Excess legitimate web traffic. Thousands or even millions of clients connecting to the web site in a short interval, e. This can happen because of required or urgent maintenance or upgrade, hardware or software failures, back-end e. Symptoms of overload[edit] The symptoms of an overloaded web server are: Requests are served with possibly long delays from 1 second to a few hundred seconds. The web server returns an HTTP error code , such as , , , , , or even , which is inappropriate for an overload condition. The web server refuses or resets interrupts TCP connections before it returns any content. In very rare cases, the web server returns only a part of the requested content. This behavior can be considered a bug , even if it usually arises as a symptom of overload. Anti-overload techniques[edit] To partially overcome above average load limits and to prevent overload, most popular web sites use common techniques like: Managing network traffic, by using: Firewalls to block unwanted traffic coming from bad IP sources or having bad patterns HTTP traffic managers to drop, redirect or rewrite requests having bad HTTP patterns Bandwidth management and traffic shaping , in order to smooth down peaks in network usage Using different domain names to serve different static and dynamic content by separate web servers, e. RAM , disks to each computer Tuning OS parameters for hardware capabilities and usage Using more efficient computer programs for web servers, etc. Using other workarounds , especially if dynamic content is involved Market share[edit] The LAMP software bundle here additionally with Squid , composed entirely of free and open-source software , is a high performance and high-availability heavy duty solution for a hostile environment Chart:

WEB SERVERS AND DYNAMIC CONTENT pdf

Breeding objectives Finding romantic fulfillment : discovering the ultimate bridegroom Untitled Valdemar #1 (Valdemar) Breakfast : the real American cuisine First thunderstorm. Individual and social narcissism, by E. Fromm. The American drink book. First Aid for the USMLE Step 3 (First Aid) 30 days to a perfect life The craft of crime Shadow of the Badger The river of grass Graphing population An essential guide to Celtic sites and their saints Residents guide to ambulatory care 6th ed The friendly air. Teaching language and literacy Second grade ing packet Web page design using html and css codes Basic bash shell scripting A humument full Friend of tax collectors and sinners Under the Queen Annes Lace What are fifty and one hundred dollars? Infant non-accidental trauma (Julia Faslon) Our brown passenger. Carbines of the U.S. Cavalry, 1861-1905 Tunnels of Terror Off The Wall Upside Down Mad Libs Look at Mother Hippo swim! she is strong and fast Places and spaces of fashion, 1800-2007 Complete Solutions Guide to Accompany Chemistry Reclaiming democratic literacy, cautiously The International Directory of Little Magazines and Small Presses, 40th Edition (2005) Ken Schultzs concise fishing encyclopedia The inclinations of nature: the anthropology of the person Learn jquery from scratch The origins of public broadcasting in Germany The works of William Cowper, Esp.comprising his poems, correspondence and translations 2-year extension of automation deadline