

Hypermedia and the Web

Servlet Architecture

Why have code in the server?

A basic question in any client-server system design is the apportioning of functionality between the client and the server.

Consider an application like Amazon.com

Need to provide access into the inventory, customer information, and shipping database. Want to ensure access to these databases is secure.

- Ideally want to ensure databases are not accessible from the open Internet. Do not want to make client side software responsible for updates to these databases, since it opens the possibility of fraud.

Need to provide customized pages of information based on user queries.

- Need to have the ability to construct custom pages based on a database query. In the absence of client-side intelligence, must perform this on the server. Even with client-side intelligence, versioning issues with JavaScript make it much easier to provide a consistent user experience with server-side code.

Want to ensure the client can be used for this purpose, as well as for accessing other sites as well.

- Means you can't have specialized code hard-wired into the browser just for a specific site (although things like the Google taskbar bend this – only possible for sites that are very important to a specific user).

Both the client and the server have the need for dramatic customization, without altering the core capabilities of the client or server.

On the server side, one could take a server, like Apache, where you have source code access, and modify it to add functionality for a specific Web site. This has several drawbacks.

- It forks your code from the rest of the project, making it more difficult to incorporate future changes (such as security patches and bug fixes) to the core server
- It is more time consuming to directly modify the server's source code than to use a technology like servlets, which do not modify the server's source code.
- Errors in source code modifications can crash the entire server, while errors in servlets typically only affect a single servlet execution instance.
- It requires more knowledge to successfully modify the server's source code, than to work inside a particular extension framework (like servlets).

For adding functionality on the server side, there are a few choices

- Plug-in approach: (Apache modules, ISAPI) have a well-defined API for external code to use when interacting with the rest of the (core) pieces of the server. Code can register to be called at different points in the processing lifecycle. Code is executed inside the same executable image as the server process.
 - Benefits: fast, more scalable
 - Drawbacks: very server-specific, hard to move to a different server architecture, more difficult to program against these interfaces, errors can affect entire server

- Code sandbox approach: (Servlets) have an execution environment for a language inside the server. Each request is handled by code running in the execution environment. For servlets, this execution environment is known as a servlet container.
 - Benefits: less complex to program in sandbox environment, errors are generally limited to the sandbox, code is portable across server vendors (and libraries calls are too, assuming J2EE compliance)
 - Drawbacks: less performance (but with modern JVMs, not as big a performance hit as in the past), can be tedious to create large, complex pages since servlet code involves large #s of println statements.
- CGI (Common Gateway Interface): One of the first server-side extension mechanisms. The server sets a series of environment variables, including information such as the HTTP method, the URL, the part of the URL after the "?", etc., then forks an external process and executes an external program, which then constructs a Web page and sends it back.
 - Benefits: relatively easy for programmers to create, errors generally do not affect server, most servers support CGI
 - Drawbacks: poor scalability due to performance hit of creating new process, poor security since CGI developers often fail to make basic security checks
- Page-embedded scripting languages (SSI, PHP): A language that is embedded inside a Web page, and is interpreted on the server side to create the HTML page that is sent back to the requesting client.
 - Benefits: relatively easy to create, even for non-programmers, errors generally do not affect server
 - Drawbacks: not as scalable as other approaches, can be difficult to modify presentation separate from logic, since logic and presentation are intertwined

Why you want database accesses to be on the server side

- When databases are accessed from code on the server side, it's possible to put the databases behind a firewall, and only allow accesses from the server machines. Dramatically reduces the number of machines that can be used to break into the databases.
- Invariably, for production systems you want to have the databases on separate machines from the web server. This allows load sharing between the web server and the database. Typically you require fewer database machines than you require web server machines. Increased site volume can be handled with more web server machines. Only for very low volume web sites would you want both to be on the same machine.