互联网应用开发技术

Web Application Development

第1课
WEB应用&HTTP

Episode One

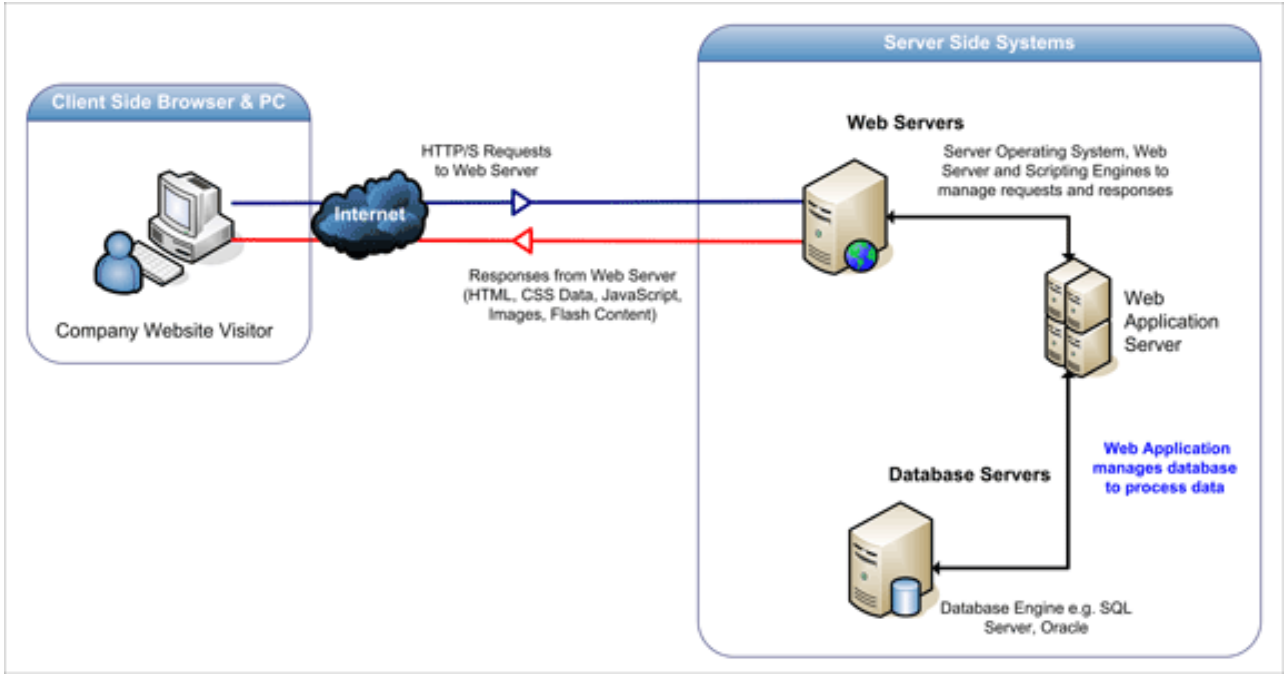
Web Application & HTTP

陈昊鹏

chen-hp@sjtu.edu.cn

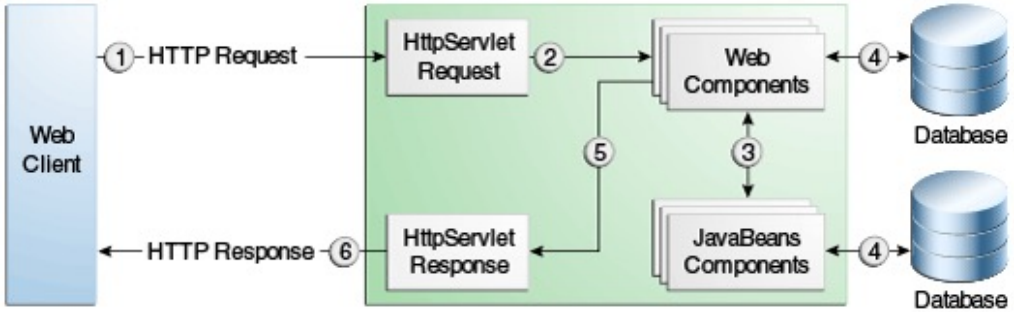
What is web application ?

- Web applications are
  - computer programs allowing website visitors to submit and retrieve data to/from a database over the Internet using their preferred web browser.



Web Applications

- Web applications are of the following types:
  - Presentation-oriented: A presentation-oriented web application generates interactive web pages containing various types of markup language (HTML, XHTML, XML, and so on) and dynamic content in response to requests.
  - Service-oriented: A service-oriented web application implements the endpoint of a web service. Presentation-oriented applications are often clients of service-oriented web applications.



Web Application Lifecycle

- A web application consists of
  - web components;
  - static resource files, such as images and cascading style sheets (CSS);
  - and helper classes and libraries.

- The web container provides many supporting services that enhance the capabilities of web components and make them easier to develop.

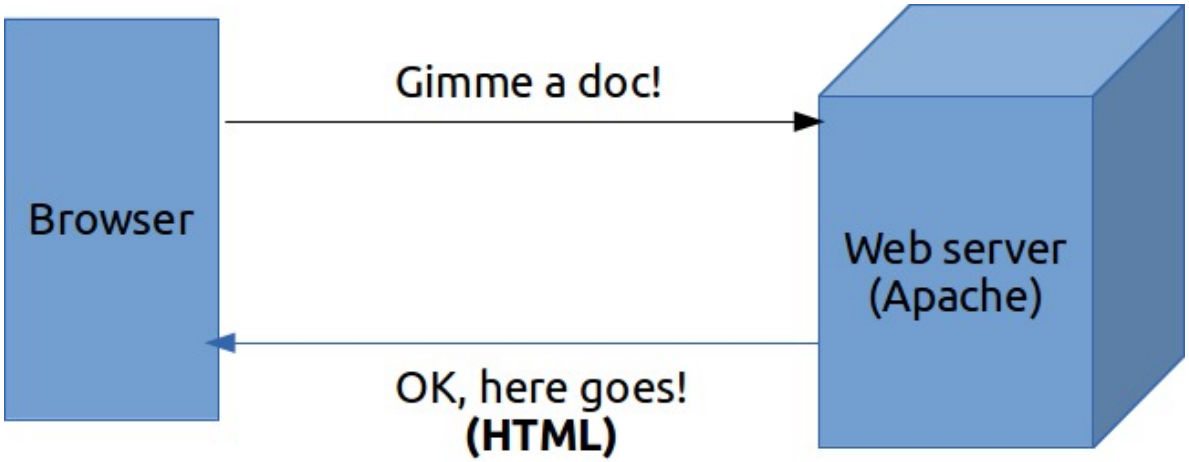
Web Application Lifecycle

- However,
  - because a web application must take these services into account, the process for creating and running a web application is different from that of traditional stand-alone Java classes.

- The process for creating, deploying, and executing a web application can be summarized as follows:
  1. Develop the web component code.
  2. Develop the web application deployment descriptor, if necessary.
  3. Compile the web application components and helper classes referenced by the components.
  4. Optionally, package the application into a deployable unit.
  5. Deploy the application into a web container.
  6. Access a URL that references the web application.

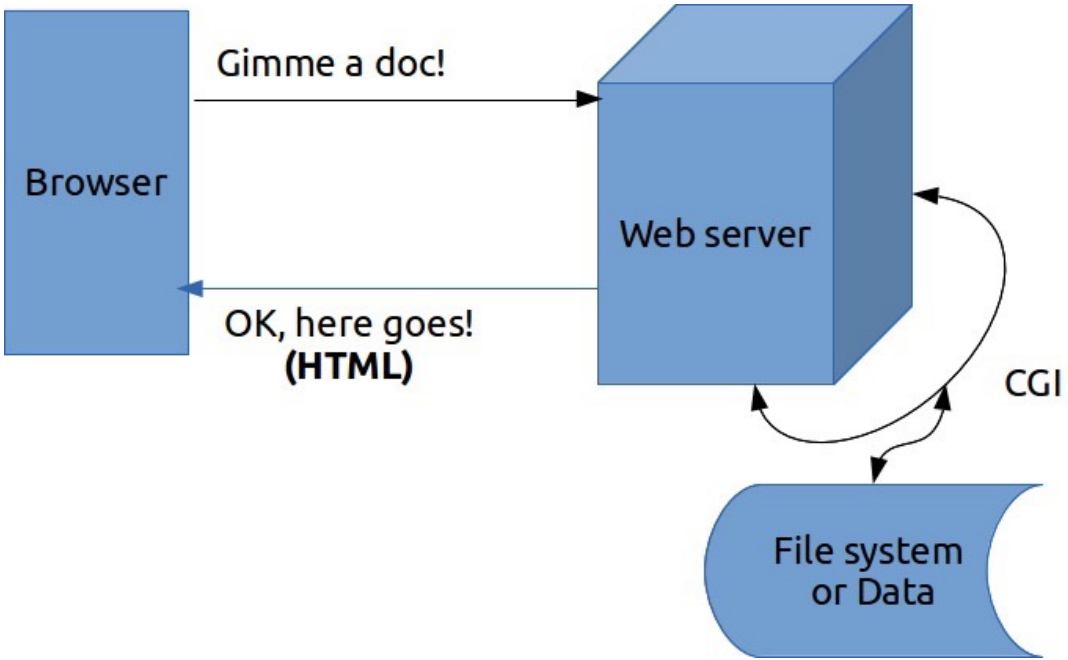
Static Websites

- All are static webpages



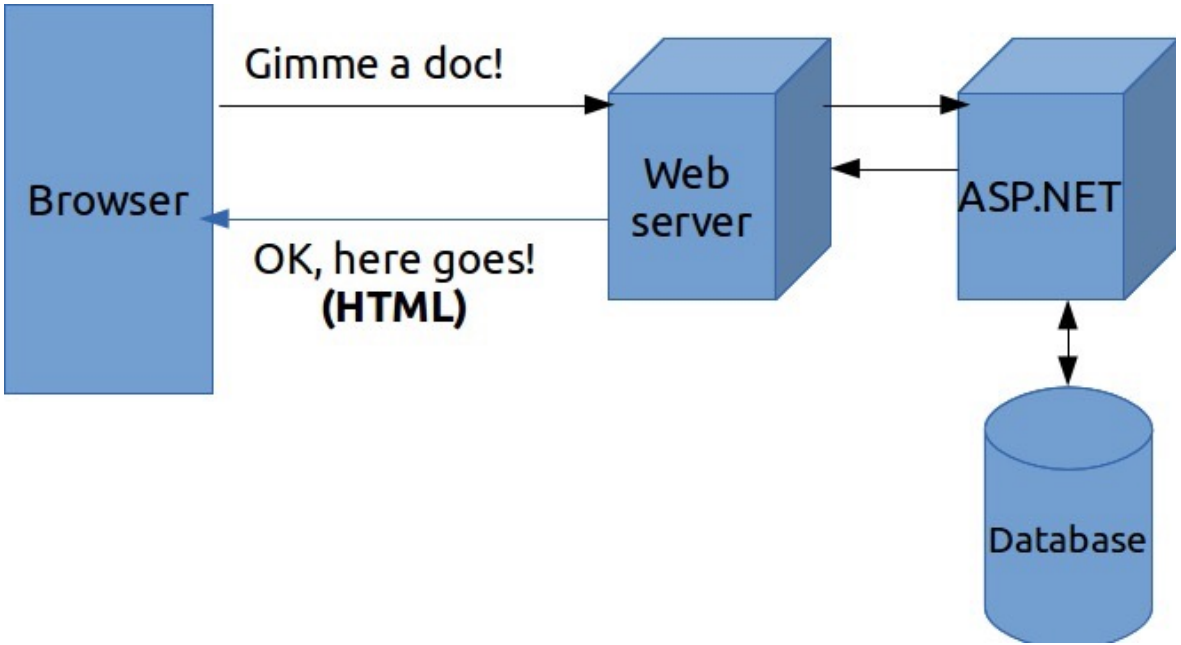
Web 1.0

- Dynamic Web application based on CGI



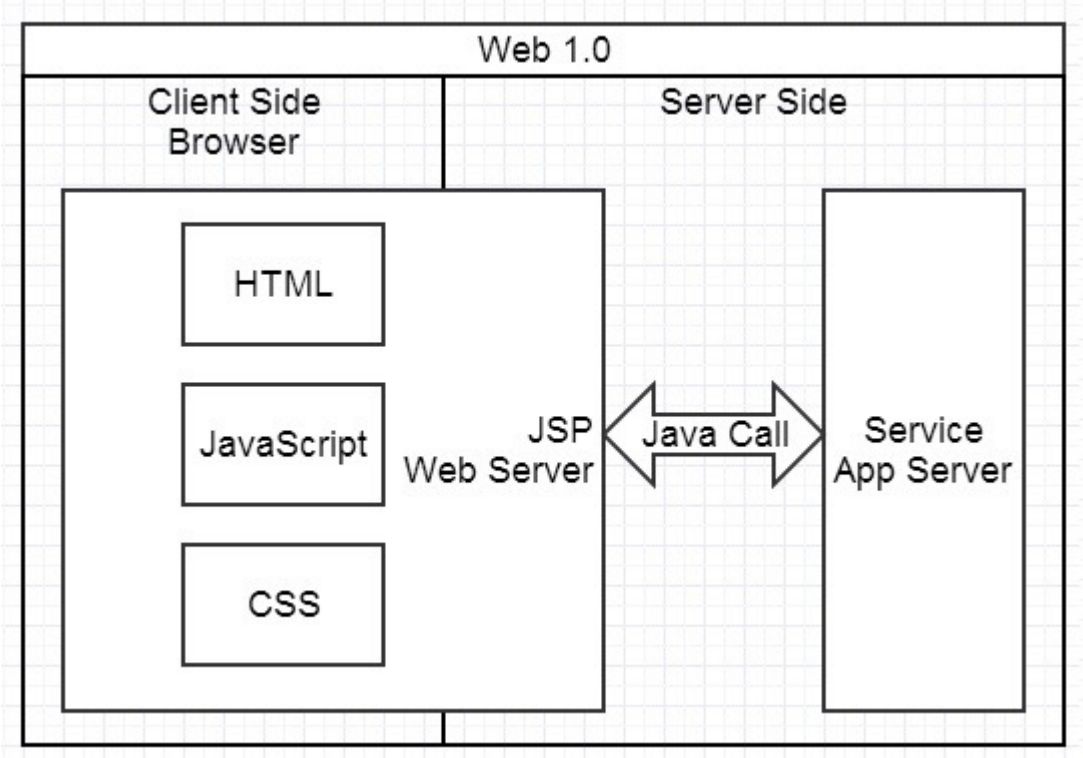
Web 1.0

- Dynamic Web application developed with ASP.NET/JSP



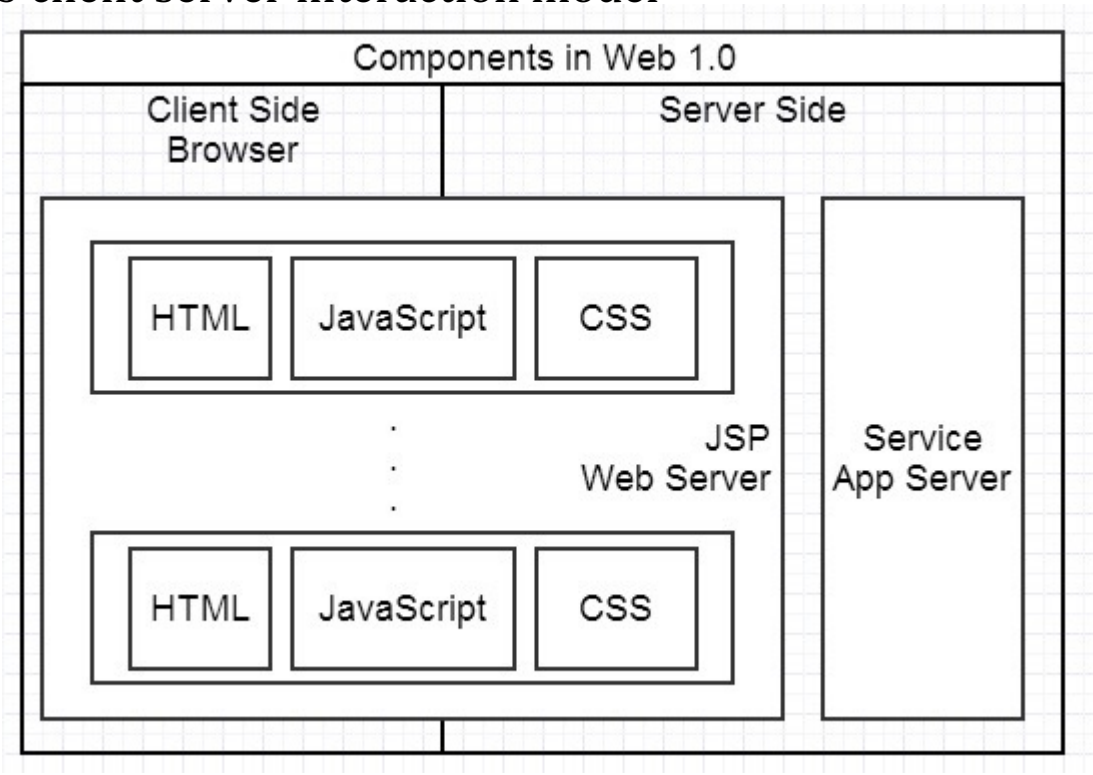
Web 1.0

- Traditional web client server interaction model



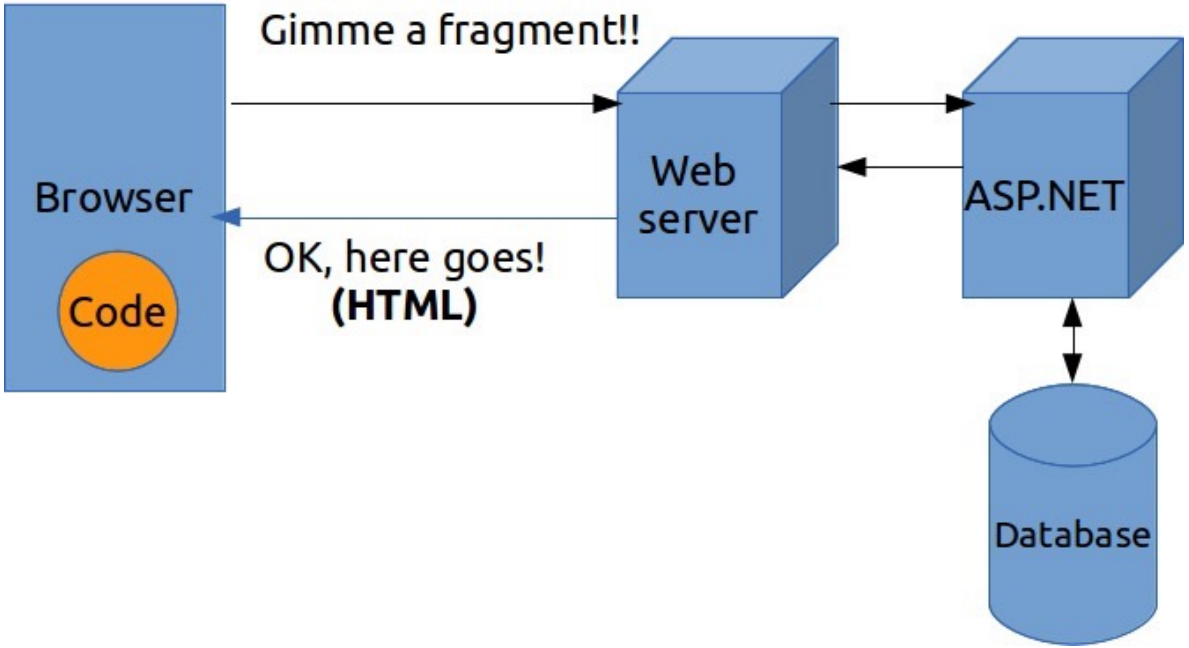
Web 1.0

- Traditional web client server interaction model

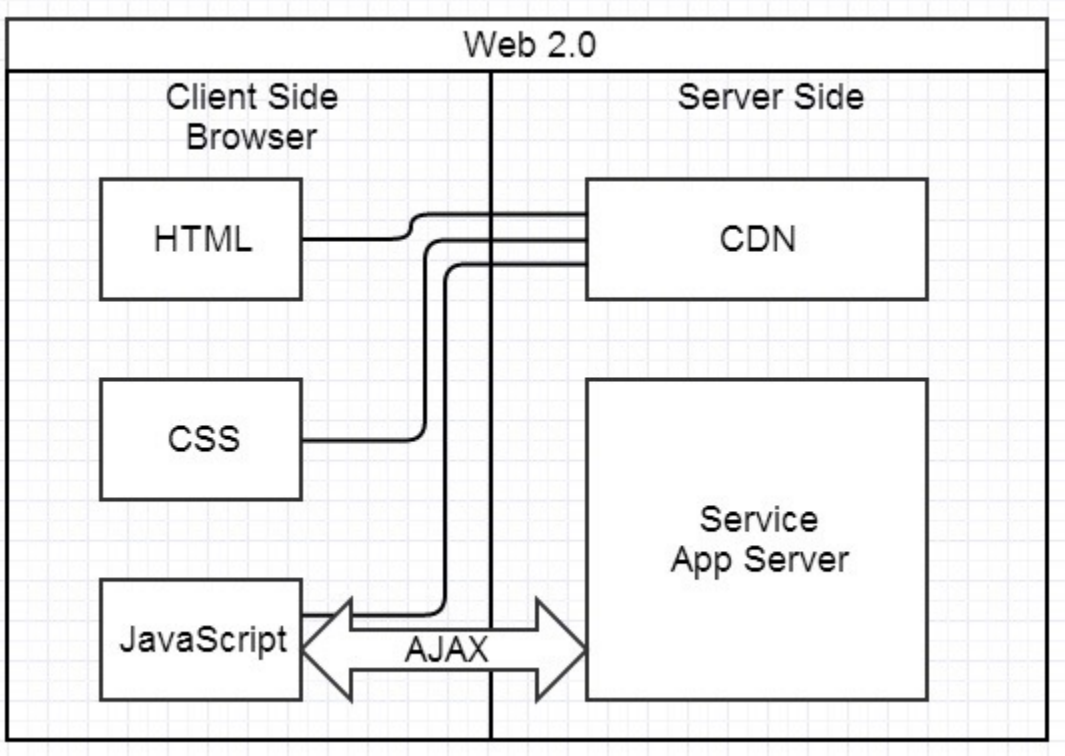


Web 2.0

- Ajax brought a new world in 2005

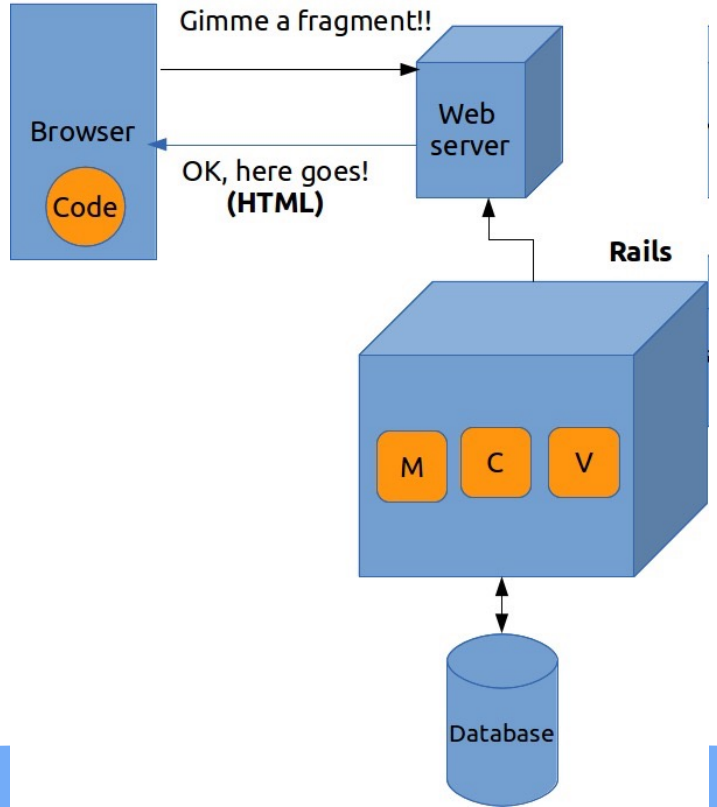


- Single Page Application



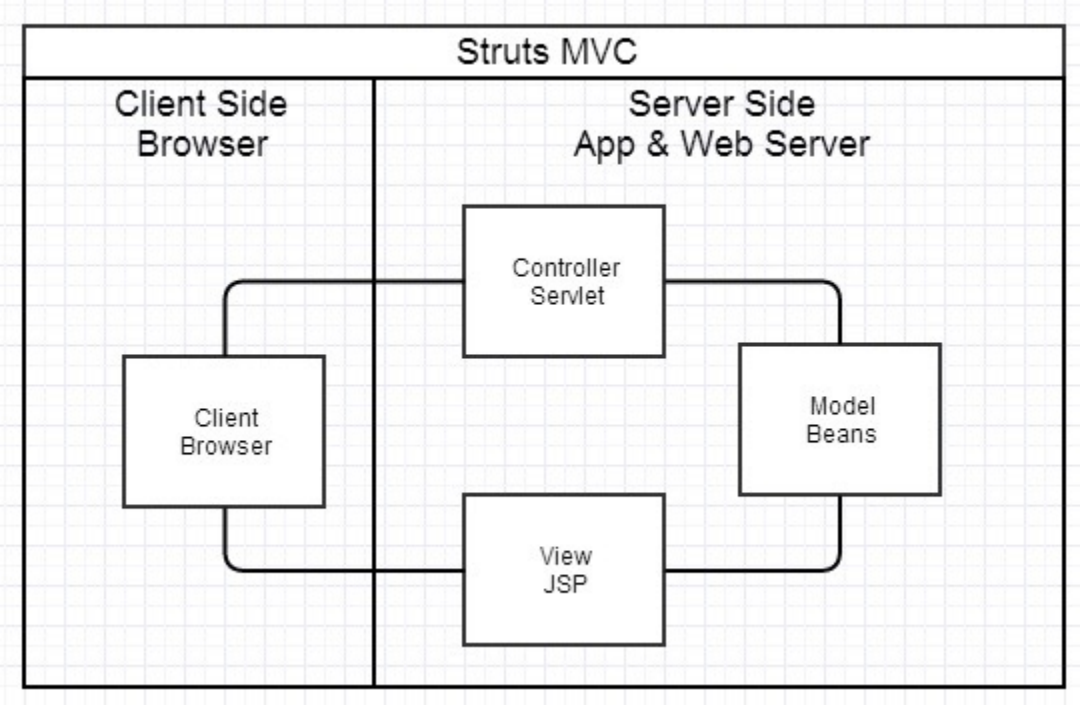
MVC Pattern

- Applying MVC patterns to improve maintainability



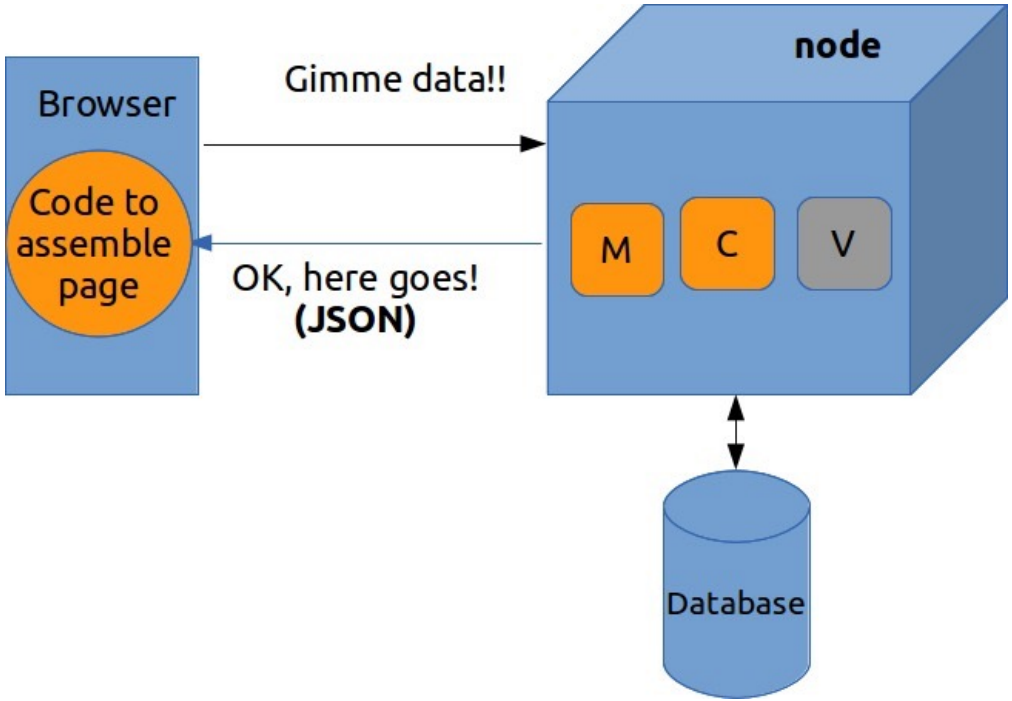
MVC Pattern

- Applying MVC patterns to improve maintainability



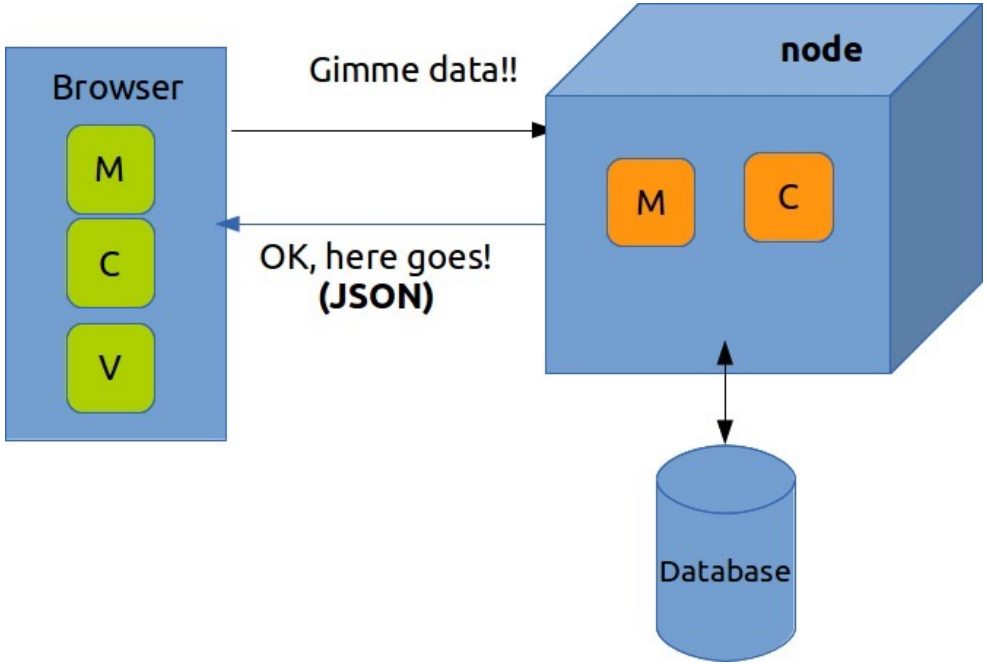
MVC & Ajax

- Assemble page with retrieved data at front-end

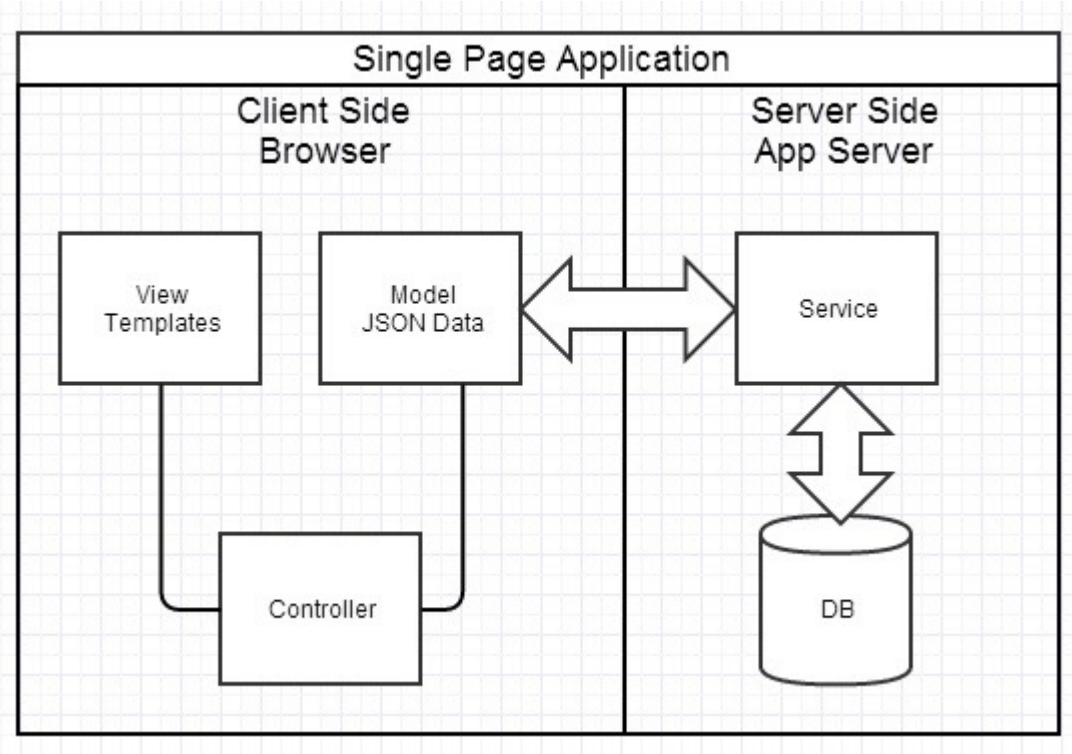


MVC & Ajax

- MVC at front-end

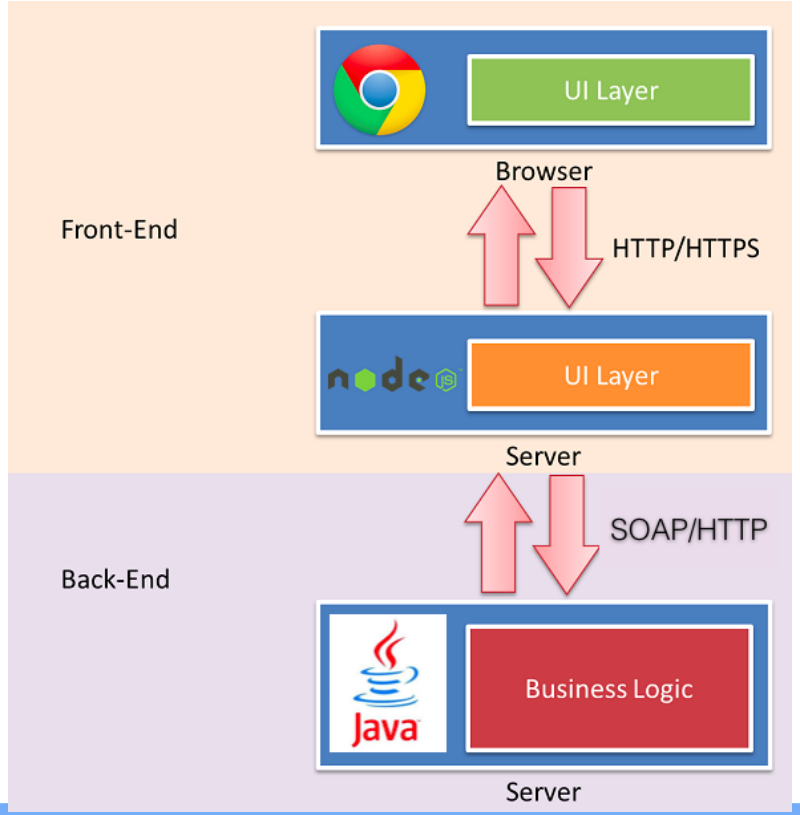


- MVC Pattern at front-end



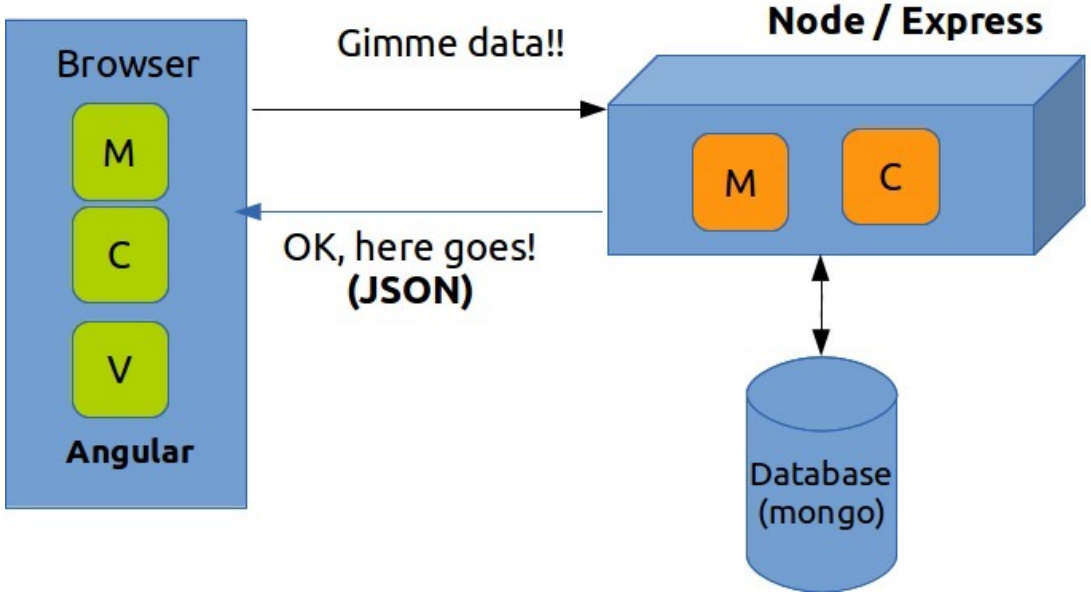
Full-stack JavaScript & REST API

- Node.js



Full-stack JavaScript & REST API

- MongoDB - JSON



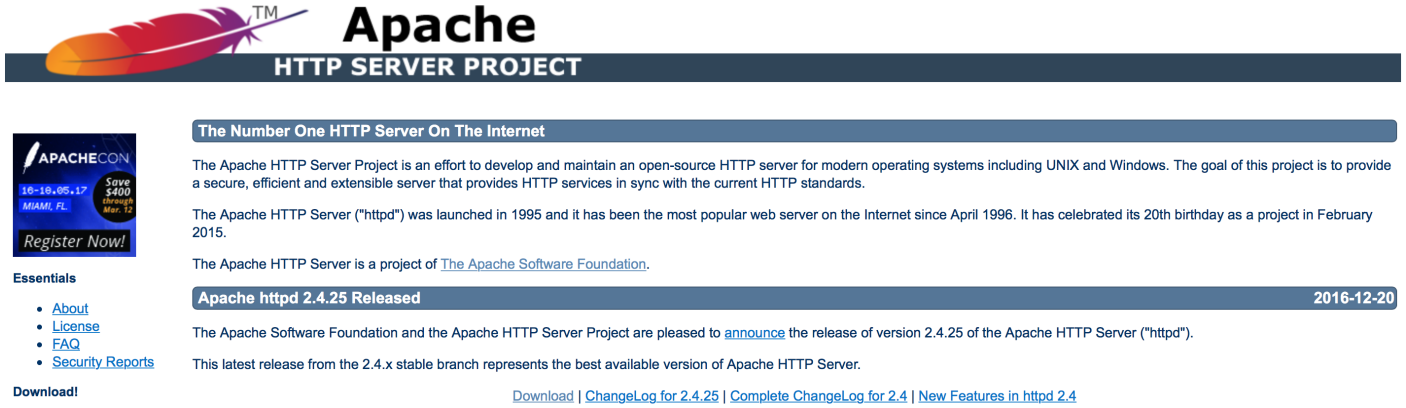
Web 3.0

- Web3 is being touted as the future of the internet.
  - The vision for this new, blockchain-based web includes cryptocurrencies, NFTs, DAOs, decentralized finance, and more.
  - It offers a read/write/own version of the web, in which users have a financial stake in and more control over the web communities they belong to.
  - Web3 promises to transform the experience of being online as dramatically as PCs and smartphones did.
  - It is not, however, without risk.

HTML 5 + CSS 3

Apache HTTP Server

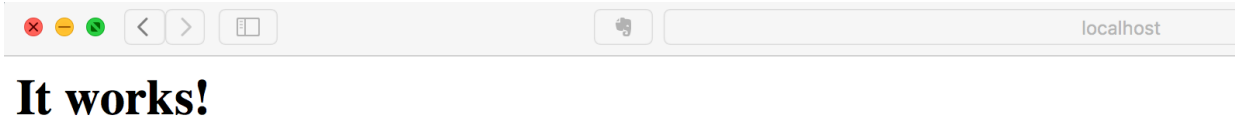
- The **Apache HTTP Server**
  - commonly referred to as **Apache** (/əˈpætʃiː/ *ə-**PA**-chee*)
  - is a web server application notable for playing a key role in the initial growth of the World Wide Web User enters interacts with a web page.



Testing

- Run `bin/httpd.exe` (on Windows)
  - The Apache HTTP Server will be running in a command window.
- Run `sudo apachectl start` (on Linux | Mac OS)
  - Mac OS has a built-in Apache HTTP Server.

- Browse `http://localhost`
  - If the Apache HTTP Server is correctly deployed, you will see:



- Shutdown `sudo apachectl stop` (on Linux | Mac OS)

PHP

- PHP
  - Hypertext Preprocessor
  - PHP is a popular general-purpose scripting language that is especially suited to web development.
  - Fast, flexible and pragmatic, PHP powers everything from your blog to the most popular websites in the world.

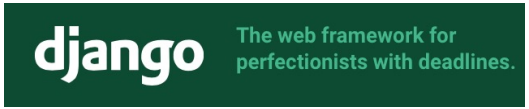
From http://www.php.net/

  - LAMP
    - Linux + Apache + MySQL + PHP/Python/Perl
    - Facebook, Yahoo, Weibo

- Ruby on Rails
  - Rails is a web application development framework written in the Ruby language.
  - It is designed to make programming web applications easier by making assumptions about what every developer needs to get started.
  - It allows you to write less code while accomplishing more than many other languages and frameworks.
  - http://rubyonrails.org
  - If you run bin/rails routes, you'll see that it has defined routes for all the standard RESTful actions.
  - Github, Airbnb

Web Frameworks for Python

- Popular Python Full-Stack Frameworks
- Diango
  - Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.



- TurboGears 2
  - TurboGears 2 is built on top of the experience of several next generation web frameworks including TurboGears 1 (of course), Django, and Rails.

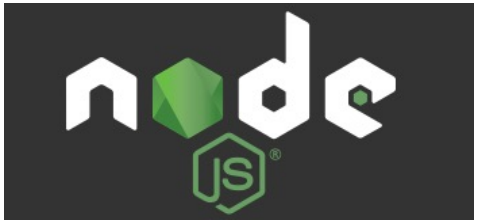
- Web2py
  - Free open source full-stack framework for rapid development of fast, scalable, secure and portable database-driven web-based applications. Written and programmable in Python.

- YouTube

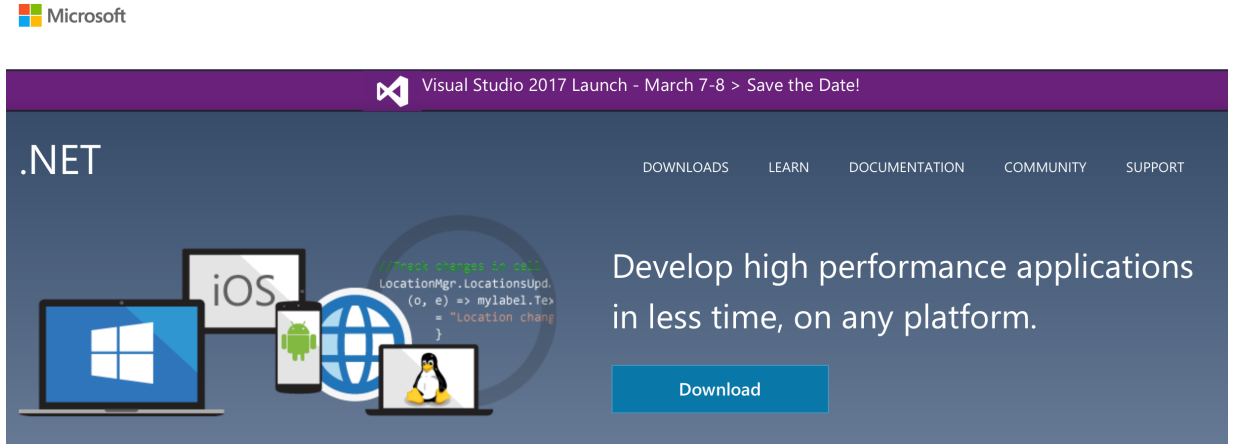
Node.js

- Node.js
  - Node.js ® is a JavaScript runtime built on Chrome's V8 JavaScript engine.
  - Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.
  - Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

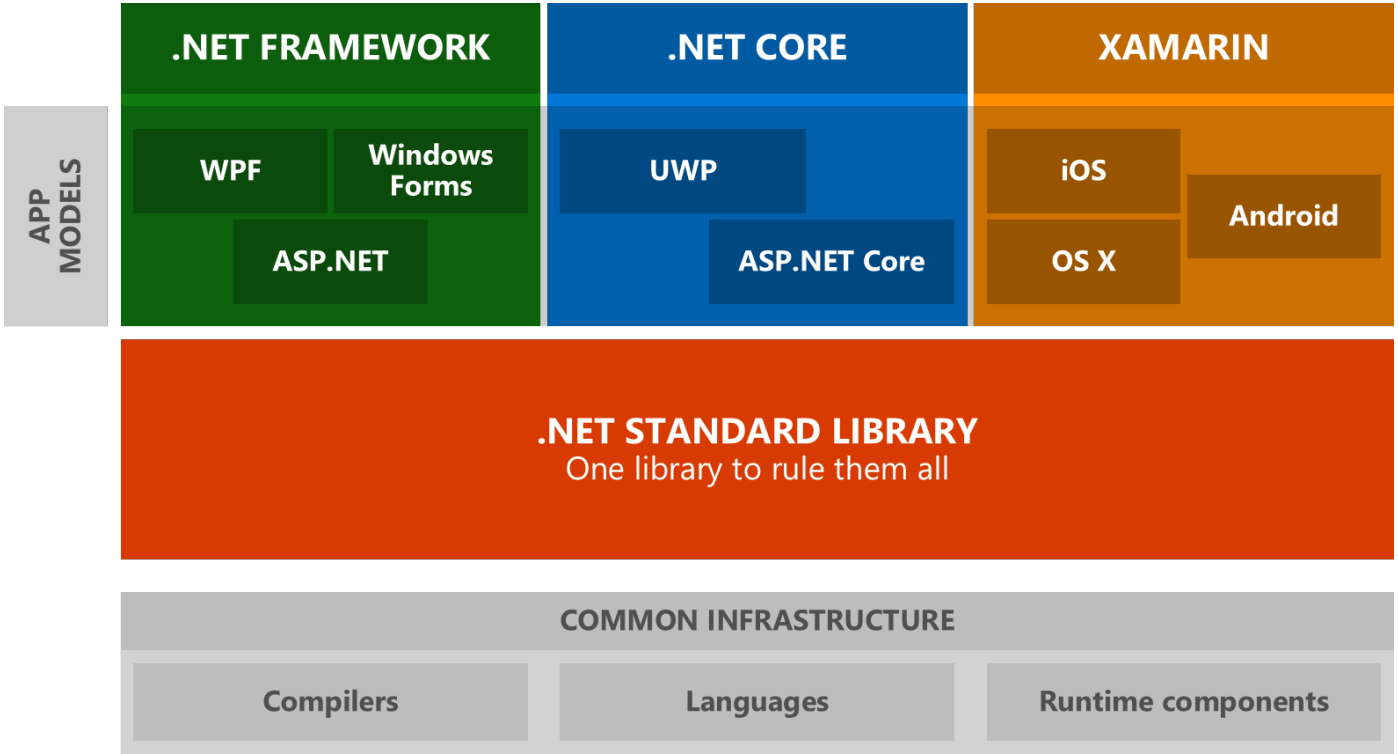
  - Emerging Technology



- Another heavy-weight framework
  - An alternative of Java EE
  - Stackoverflow



Windows .NET


APP MODELS
.NET FRAMEWORK
WPF
Windows Forms
ASP.NET
.NET CORE
UWP
ASP.NET Core
XAMARIN
iOS
Android
OS X
.NET STANDARD LIBRARY
One library to rule them all
COMMON INFRASTRUCTURE
Compilers
Languages
Runtime components

Other Options

- Erlang
  - Erlang is a programming language used to build massively scalable soft real-time systems with requirements on high availability.

- Scala
  - Construct elegant class hierarchies for maximum code reuse and extensibility, implement their behavior using higher-order functions. Or anything in-between.

- Go
  - Go is an open source programming language that makes it easy to build simple, reliable, and efficient software.

HTTP Protocol

- HTTP functions as a request–response protocol in the client–server computing model.
  - A web browser, for example, may be the *client* and an application running on a computer hosting a website may be the *server*.
  - The client submits an HTTP *request* message to the server.
  - The server, which provides *resources* such as HTML files and other content, or performs other functions on behalf of the client, returns a *response* message to the client.
    - The response contains completion status information about the request and may also contain requested content in its message body.
  - A web browser is an example of a *user agent* (UA).
    - Other types of user agent include the indexing software used by search providers (web crawlers), voice browsers, mobile apps, and other software that accesses, consumes, or displays web content.

HTTP Protocol

- HTTP resources are identified and located on the network by Uniform Resource Locators (URLs),
  - using the Uniform Resource Identifiers (URI's) schemes *http* and *https*.
  - URIs and hyperlinks in HTML documents form inter-linked hypertext documents.

- Every HTTP URL conforms to the syntax of a generic URI. A generic URI is of the form:
  - scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]

  - For example:
  - https://en.wikipedia.org/wiki/Uniform_Resource_Locator

Uniform Resource Locator

- **scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]**
  - **scheme**,
    - consisting of a sequence of characters beginning with a letter and followed by any combination of letters, digits, plus (+), period (.), or hyphen (-).
    - It is followed by a colon (:).
    - Examples of popular schemes include http, ftp, mailto, file, data, and irc.
  - Two slashes (//)
    - This is required by some schemes and not required by some others.
  - An **authority part**, comprising:
    - An optional authentication section of a user name and password, separated by a colon, followed by an at symbol (@)
  - A "**host**",
    - consisting of either a registered name (including but not limited to a hostname), or an IP address.
    - An optional port number, separated from the hostname by a colon

Uniform Resource Locator

- **scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]**
  - A **path**,
    - which contains data, usually organized in hierarchical form, that appears as a sequence of segments separated by slashes.
  - An optional **query**,
    - separated from the preceding part by a question mark (?), containing a query string of non-hierarchical data.
    - Its syntax is not well defined, but by convention is most often a sequence of attribute–value pairs separated by a delimiter.

Query delimiter	Example
Ampersand (&)	key1=value1&key2=value2
Semicolon (;)	key1=value1;key2=value2

  - An optional **fragment**,
    - separated from the preceding part by a hash (#). The fragment contains

A Request & A Response

- http://www.example.com/index.html

- **Client request**

```
GET / HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Request Syntax

- A client sends *request messages* to the server, which consist of:
  - a **request line**, consisting of the case-sensitive request method, a space, the requested URL, another space, the protocol version, a carriage return, and a line feed, e.g.:

    GET /images/logo.png HTTP/1.1

  - zero or more request header fields (at least 1 or more headers in case of HTTP/1.1), each consisting of the case-insensitive field name, a colon, optional leading whitespace, the field value, an optional trailing whitespace and ending with a carriage return and a line feed, e.g.:

    Host: www.example.com

    Accept-Language: en

  - an empty line, consisting of a carriage return and a line feed;
  - an optional message body.

  - In the HTTP/1.1 protocol, all header fields except Host: hostname are optional.

Request Syntax

- A client sends *request messages* to the server, which consist of:
  - a **request line**, consisting of the case-sensitive request method, a space, the requested URL, another space, the protocol version, a carriage return, and a line feed, e.g.:

    GET /images/logo.png HTTP/1.1

  - zero or more request header fields (at least 1 or more headers in case of HTTP/1.1), each consisting of the case-insensitive field name, a colon, optional leading whitespace, the field value, an optional trailing whitespace and ending with a carriage return and a line feed, e.g.:

    Host: www.example.com

    Accept-Language: en

  - an empty line, consisting of a carriage return and a line feed;
  - an optional message body.

  - In the HTTP/1.1 protocol, all header fields except Host: hostname are optional.

Request methods

- HTTP defines methods to indicate the desired action to be performed on the identified resource.
  - GET
    - The GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect. (idempotent)
  - HEAD
    - The HEAD method asks for a response identical to that of a GET request, but without the response body. (idempotent)
  - POST
    - The POST method requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI.
  - PUT
    - The PUT method requests that the enclosed entity be stored under the supplied URI. (idempotent)
  - DELETE
    - The DELETE method deletes the specified resource. (idempotent)

Request methods

- HTTP defines methods to indicate the desired action to be performed on the identified resource.
  - TRACE
    - The TRACE method echoes the received request so that a client can see what (if any) changes or additions have been made by intermediate servers. (idempotent)
  - OPTIONS
    - The OPTIONS method returns the HTTP methods that the server supports for the specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource. (idempotent)
  - CONNECT
    - The CONNECT method converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.
  - PATCH
    - The PATCH method applies partial modifications to a resource.
  - All general-purpose HTTP servers are required to implement at least the GET and HEAD methods, and, whenever possible, also the OPTIONS method.

Safe methods & Idempotent methods

- Some of the methods (for example, HEAD, GET, OPTIONS and TRACE) are, by convention, defined as *safe*
  - which means they are intended only for information retrieval and should not change the state of the server.

- Methods PUT and DELETE are defined to be idempotent,
  - meaning that multiple identical requests should have the same effect as a single request
  - or the response code it returns may be different on subsequent requests, the system state will be the same every time.
  - Methods GET, HEAD, OPTIONS and TRACE, being prescribed as safe, should also be idempotent, as HTTP is a stateless protocol

Safe methods & Idempotent methods

HTTP Method	RFC	Request Has Body	Response Has Body	Safe	Idempotent	Cacheable
GET	RFC 7231	No	Yes	Yes	Yes	Yes
HEAD	RFC 7231	No	No	Yes	Yes	Yes
POST	RFC 7231	Yes	Yes	No	No	Yes
PUT	RFC 7231	Yes	Yes	No	Yes	No
DELETE	RFC 7231	No	Yes	No	Yes	No
CONNECT	RFC 7231	Yes	Yes	No	No	No
OPTIONS	RFC 7231	Optional	Yes	Yes	Yes	No
TRACE	RFC 7231	No	Yes	Yes	Yes	No
PATCH	RFC 5789	Yes	Yes	No	No	Yes

Request Header Fields

Name	Description	Example
Accept	Media type(s) that is/are acceptable for the response. See Content negotiation.	Accept: text/html
Accept-Charset	Character sets that are acceptable.	Accept-Charset: utf-8
Accept-Datetime	Acceptable version in time.	Accept-Datetime: Thu, 31 May 2007 20:35:00 GMT
Accept-Encoding	List of acceptable encodings. See HTTP compression.	Accept-Encoding: gzip, deflate
Accept-Language	List of acceptable human languages for response. See Content negotiation.	Accept-Language: en-US
Cache-Control	Used to specify directives that *must* be obeyed by all caching mechanisms along the request-response chain.	Cache-Control: no-cache
Connection	Control options for the current connection and list of hop-by-hop request fields.[14]Must not be used with HTTP/2.[15]	Connection: keep-aliveConnection: Upgrade
Content-Encoding	The type of encoding used on the data. See HTTP compression.	Content-Encoding: gzip
Content-Length	The length of the request body in octets (8-bit bytes).	Content-Length: 348
Content-Type	The Media type of the body of the request (used with POST and PUT requests).	Content-Type: application/x-www-form-urlencoded
Cookie	An HTTP cookie previously sent by the server with Set-Cookie (below).	Cookie: $Version=1; Skin=new;

A Request & A Response

- http://www.example.com/index.html

- **Server response**

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 155
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close
```

```
<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>
      Hello World, this is a very simple HTML document.
    </p>
  </body>
</html>
```

Response Syntax

- A server sends *response messages* to the client, which consist of:[47]
  - a **status line**, consisting of the protocol version, a space, the response status code, another space, a possibly empty reason phrase, a carriage return and a line feed, e.g.:
    HTTP/1.1 200 OK
  - zero or more response header fields, each consisting of the case-insensitive field name, a colon, optional leading whitespace, the field value, an optional trailing whitespace and ending with a carriage return and a line feed, e.g.:
    Content-Type: text/html
  - an empty line, consisting of a carriage return and a line feed;
  - an optional message body.

Response Status

- In HTTP/1.0 and since,
  - the first line of the HTTP response is called the *status line* and includes a numeric *status code* (such as "404") and a textual *reason phrase* (such as "Not Found").

- The first digit of the status code defines its class:
  - 1XX (informational) The request was received, continuing process.
  - 2XX (successful) The request was successfully received, understood, and accepted.
  - 3XX (redirection) Further action needs to be taken in order to complete the request.
  - 4XX (client error) The request contains bad syntax or cannot be fulfilled.
  - 5XX (server error) The server failed to fulfill an apparently valid request.

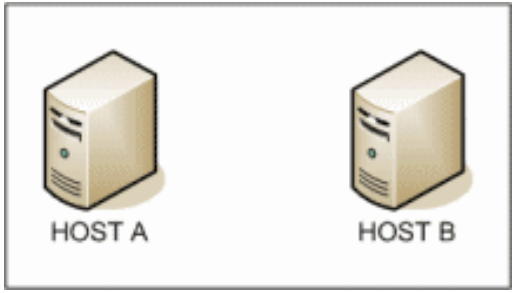
Response Header Fields

REliable, INtelligent & Scalable Systems

Field name	Description	Example
Accept-CH	Requests HTTP Client Hints	Accept-CH: UA, Platform
Access-Control-Allow-Origin, Access-Control-Allow-Credentials, Access-Control-Expose-Headers, Access-Control-Max-Age, Access-Control-Allow-Methods, Access-Control-Allow-Headers[13]	Specifying which web sites can participate in cross-origin resource sharing	Access-Control-Allow-Origin: *
Age	The age the object has been in a proxy cache in seconds	Age: 12
Allow	Valid methods for a specified resource. To be used for a *405 Method not allowed*	Allow: GET, HEAD
Content-Encoding	The type of encoding used on the data. See HTTP compression.	Content-Encoding: gzip
Content-Language	The natural language or languages of the intended audience for the enclosed content[51]	Content-Language: da
Content-Length	The length of the response body in octets (8-bit bytes)	Content-Length: 348
Content-Location	An alternate location for the returned data	Content-Location: /index.htm
Content-Type	The MIME type of this content	Content-Type: text/html; charset=utf-8

HTTP session

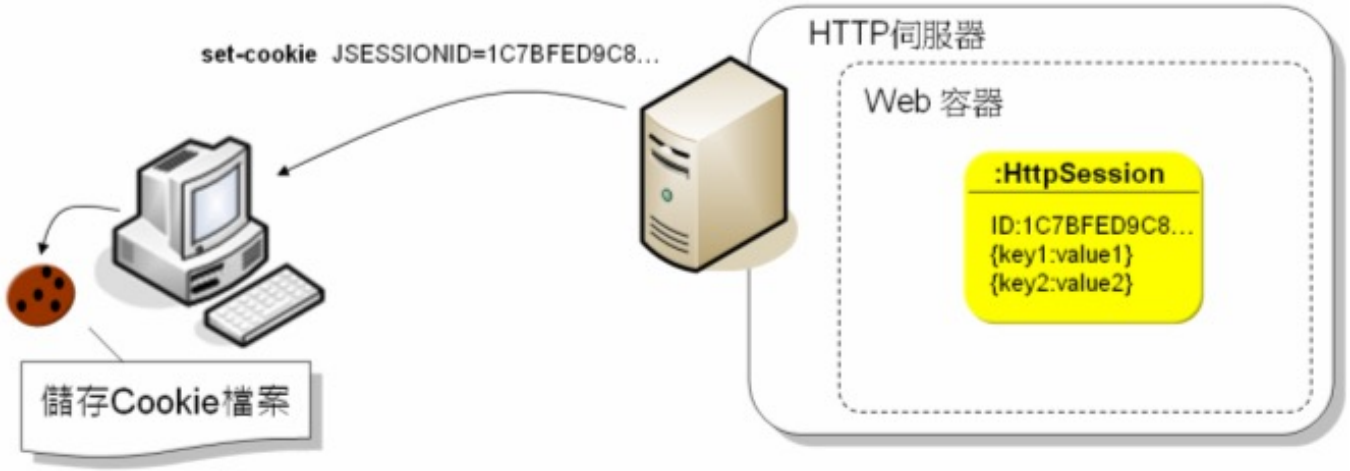
- An HTTP session is a sequence of network request-response transactions.
  - TCP 3-Way Handshake

EVENT	DIAGRAM
Host A **sends** a TCP **SYN**chronize packet to Host B Host B receives A's **SYN** Host B **sends** a **SYN**chronize-ACKnowledgement Host A receives B's **SYN-ACK** Host A **sends** **ACK**nowledge Host B receives **ACK**. ***TCP socket connection is ESTABLISHED.***	 TCP Three Way Handshake (SYN, SYN-ACK, ACK)

  - *from: http://blog.csdn.net/wangshihui512/article/details/9051819*

HTTP session state

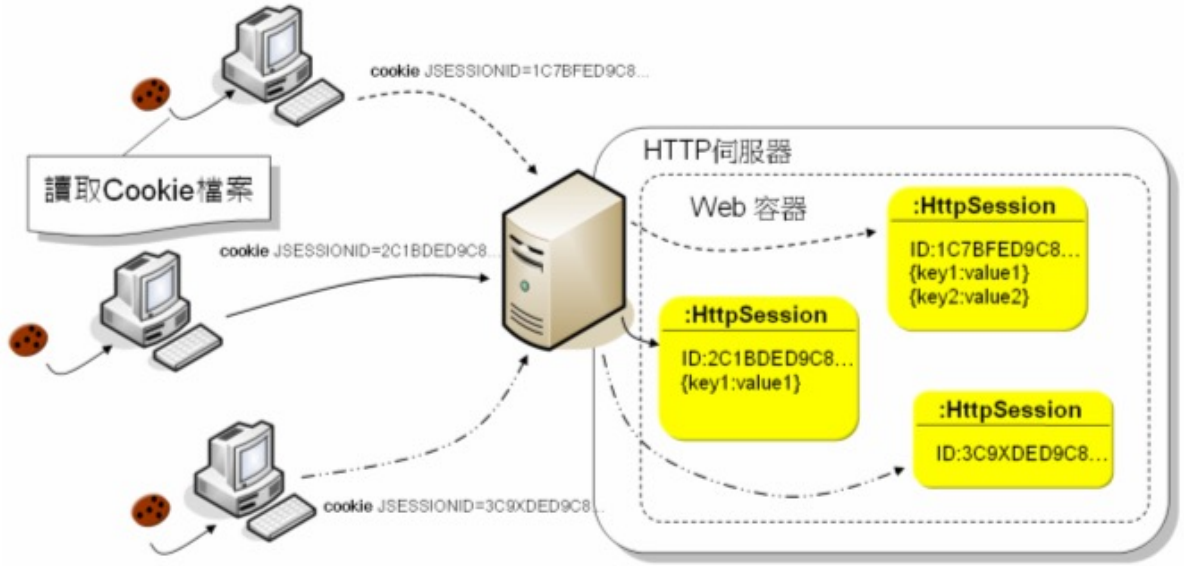
- HTTP is a stateless protocol
  - A stateless protocol does not require the HTTP server to retain information or status about each user for the duration of multiple requests.



  - *from: https://www.openhome.cc/Gossip/ServletJSP/BehindHttpSession.html*

HTTP session state

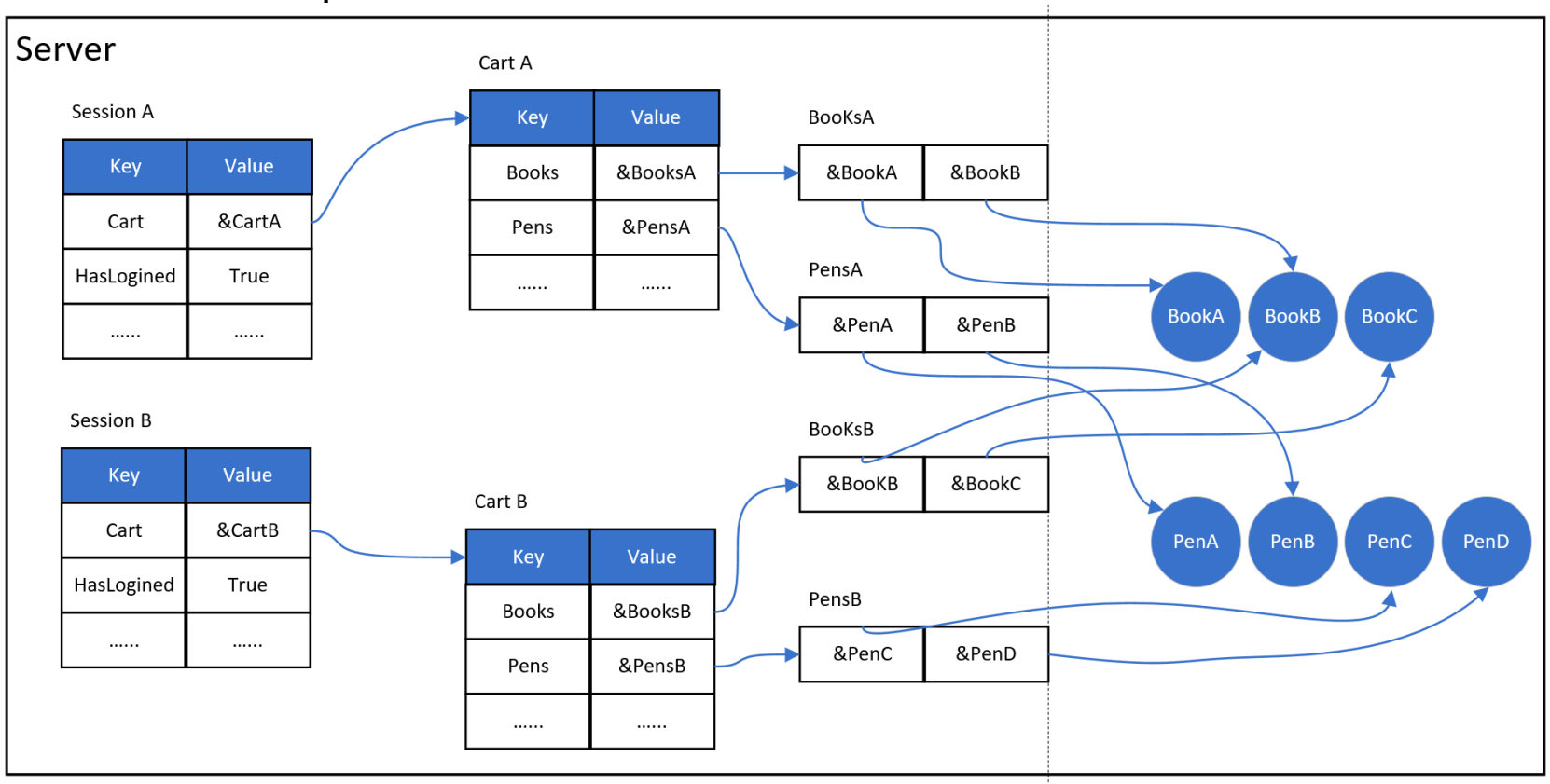
- HTTP is a stateless protocol
  - A stateless protocol does not require the HTTP server to retain information or status about each user for the duration of multiple requests.



  - *from: https://www.openhome.cc/Gossip/ServletJSP/BehindHttpSession.html*

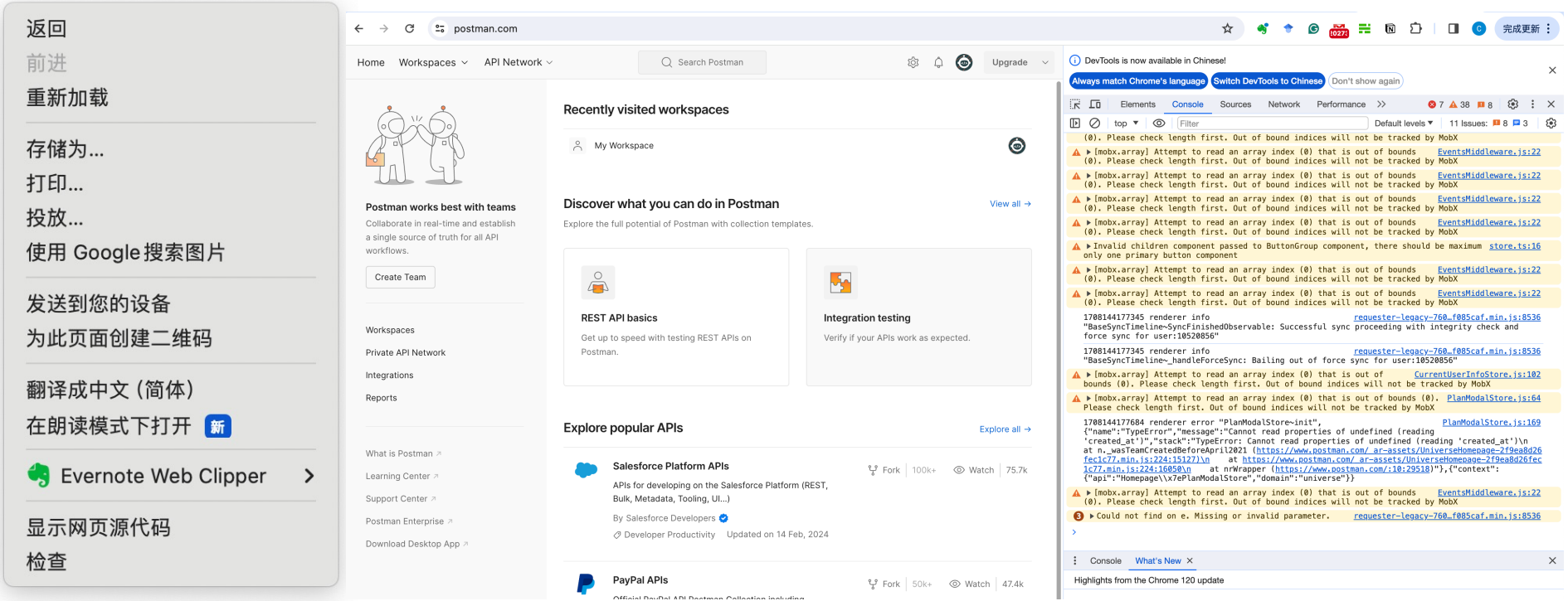
HTTP session state

- HTTP is a stateless protocol



Chrome Debugging

REin
REliable, INtelligent & Scalable Systems

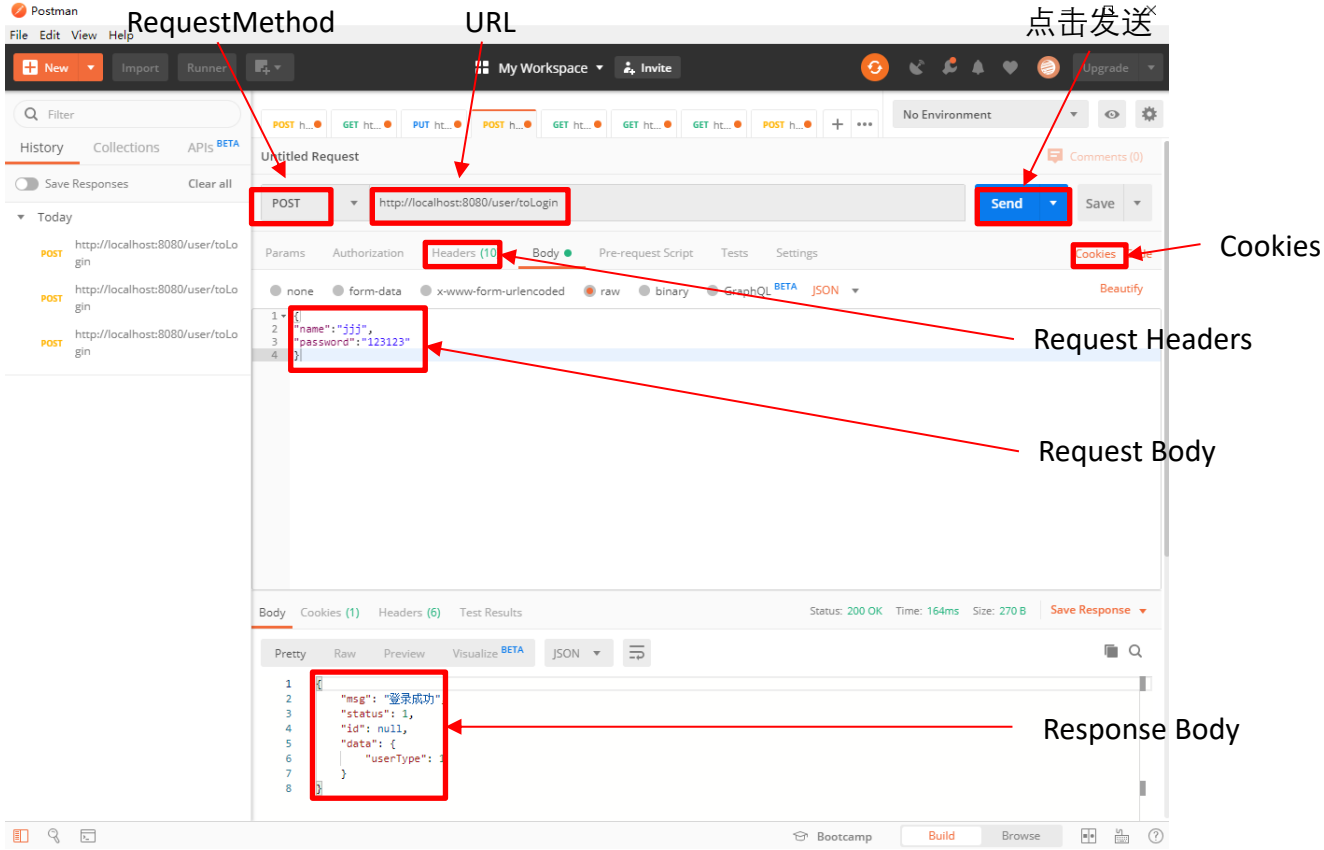


Postman

- Postman is an API platform for building and using APIs.
  - Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

- Dowload
  - https://www.postman.com/
- Docs
  - https://learning.postman.com/docs/introduction/overview/

Postman


RequestMethod
URL
点击发送
Cookies
Request Headers
Request Body
Response Body
Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
No Environment
History
Collections
APIs BETA
Save Responses
Clear all
Today
http://localhost:8080/user/toLogin
Untitled Request
POST
http://localhost:8080/user/toLogin
Send
Save
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL BETA
JSON
Beautify
{
"name":"jjj",
"password":"123123"
}
Body
Cookies (1)
Headers (6)
Test Results
Status: 200 OK
Time: 164ms
Size: 270 B
Save Response
Pretty
Raw
Preview
Visualize BETA
JSON
{
"msg": "登录成功",
"status": 1,
"id": null,
"data": {
"userType": 1
}
}
Bootcamp
Build
Browse

References

- Web开发的发展史
  - https://linux.cn/article-3166-1.html
- JAVE SOFTWARE SOLUTIONS: FOUNDATIONS OF PROGRAM DESIGN (Eighth Edition)
  - JOHN LEWIS(Virginia Tech), WILLIAM LOFTUS (Accenture)
- The Java EE 8 Tutorial – Web Applicaitons
  - https://javaee.github.io/tutorial/webapp001.html
- Web 3.0
  - https://en.wikipedia.org/wiki/Web3
- What is Web 3..0
  - https://hbr.org/2022/05/what-is-web3

References

- Django
  - https://www.djangoproject.com
- Turbogears
  - http://www.turbogears.org
- Web2py
  - http://www.web2py.com/init/default/documentation
- Node.js
  - https://nodejs.org/
- Windows .NET
  - http://www.microsoft.com/net
- Erlang
  - http://www.erlang.org
- Scala
  - http://www.scala-lang.org
- Go
  - https://golang.org

References

- Hypertext Transfer Protocol
  - https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- Uniform Resource Locator
  - https://en.wikipedia.org/wiki/Uniform_Resource_Locator
- List of HTTP header fields
  - https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#Request_fields
- HttpSession 原理
  - https://www.openhome.cc/Gossip/ServletJSP/BehindHttpSession.html

- *Web开发技术*
- *Web Application Development*

Thank You!