Enterprise - distribuerad data
Kallas inte distributed programming för att Enterprise låter “häftigare/mäktigare/större”.

Arbeta med distribuerad data, (P2P, Client/Server...)
Databashanterare till java applikation är en form av client/server kommunikation.
När en anslutning är bryggd mellan en client och en server så skickas en förfrågan till servern som processeras av servern för att sedan skicka tillbaks ett svar. När svaret kommer tillbaks så stängs anslutningen. I en client/server relation så vill klienten hämta en specifik resurs från servern, detta görs med hjälp av en URL.
ex. https://google.com?q=enterprise

En bekanta HTTP adressen
https://support.mozilla.org/en-US/questions/

URL (Uniform Resource Locator)
Letar upp en specifik resurs på nätet

- https://support.mozilla.org/ är mozillas hjälp hem hemsida
- http, url Scheme, berättar hur vi kommer åt resursen. Det finns ett flertal olika protokoll, ex.
- ftp, mailto, stmp, https, I vårt fall är det hypertext transfer protocol
- support.mozilla.org är vilken värd (host) som håller i resursen. Datorn använder sedan DNS (domain name system) för att leta upp adressen till support.mozilla.org Prova: gå till kommandotolken och skriv “ping google.com” för att få ut adressen till google.
- /en-US/questions/ är vår URL path, dvs. där resursen finns

OBS att göra en http förfrågan till någon resurs innebär att den resursen sedan anropar andra resurser för att hämta olika css, javascript, bilder.. filer

I vissa fall behövs även portnumret specificeras, ex. https://google.com:80/ säger att vi ska försöka ansluta till google.com på port 80 med protokollet https. 80 är dock default porten och brukar vanligtvis utelämnas.

Finns två delar till på en URL, först kommer query
https://google.com?q=hej skickar med en “query” till google med ett key-value referens. Q är parametern och hej är värdet.

Andra delen är fragments
https://en.wikipedia.org/wiki/Enterprise_JavaBeans#History
Det som kommer efter # är ett fragment. Fragment processeras inte på server sidan, däremot görs det hos klienten. Den går dit där DOM “id:et” History finns.

URL Encoding
“Safe characters”:

- Alfabetet
- Nummer

- $-_.+*()

ex. så blir space %20

Content type/media type (Datan)

Med varje URL så skickas även en content type. Detta är för att undvika att ex. en bild inte försöker att laddas som ren text eller tvärtom.

En lista med MIMIE (Multi-purpose internet mail extensions) typer:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Complete_list_of_MIME_types

HTTP (språket)

HTTP/1.1 specifikationen sätter upp ett universellt språk för all kommunikation via HTTP protokollet.

Klient och Server

I en klient -> server kommunikation med HTTP protokollet skickas en HTTPRequest till servern som svarar med en HTTPResponse. Eftersom HTTP är en nät standard så förstår klienten och servern svaren, detta kallas "A single HTTPTransaction". Detta löser problem såsom att servern inte förstår klientens fråga. Eventuella fel-meddelande kan då skickas
http://www.yegor256.com/2015/11/16/json-vs-xml.html
http://www.restapitutorial.com/httpstatuscodes.html

HTTP metoder

Namn	Beskrivning
GET	Hämtar resurs
POST	Uppdaterar resurs
PUT	Lagrar resurs
DELETE	Tar bort resurs
Head	Hämtar "header" för resursen

GET och POST kallas primär och hanteras i HTML specifikationen

POST, PUT och DELETE anses vara "unsafe" därför att dessa är menade att göra ändringar på servern som klienten kan hållas ansvarig för, ex. göra ett köp online eller lagra en vara i en online kundvagn.

POST vs. GET

GET	POST
Cachas	Cachas INTE
Sparas i historik	Sparas INTE
Bokmärken	Ej bokmärkbar
Okänslig data	Känslig data
Längdbegränsing (default 8124 byte)	“ingen” begränsning
Hämtar resurs	Uppdaterar resurs

Status codes
Används för att snabbare ge en idé om hur förfrågan hanterades

Interval	Kategori
100-199	Informativ
200-299	Lyckade
300-399	Omdirigeringar
400-499	Klient fel
500-599	Server fel

XML och Json
Två vanliga data format på nätet.

XML (MIME: { text/xml, application/xml }, Filtyp: .xml)
JSON (MIME: { text/json, application/json }, Filtyp: .json)

JSON är snabbare, enklare, kan använda arrays.
XML är mer komplext, kan hålla meta-data (attribut-värden), har xsd scheman, XSTL transformationer..

Både XML och JSON är självbeskrivande, innehåller hierarkisk data, kan parsas av inprincip alla programmeringsspråk.