**Ներածություն**

Http – hypertext transfer protocol, իրենից ներկայացնում է կոմունիկացիոն պրոտոկոլ TCP/IP – ի հիման վրա որը օգտագործվում է տվյալների փոխանակման համար (html էջեր, ֆայլեր, request – ների մշակման այլ արդյունքեր)։ Http սպեցիֆիկացիան սահմանում է թե ինչպես են web client – ի request – ները ստեղծվելու և ուղարկվելու սերվերին և ինչպես է սերվերը պատասխանելու դրանց։

Հիմնական բնութագիրը

* Http – ն չի պահանջում մշտական միացման առայություն։ Յուրաքանչյուր նոր Http request – ի ժամանակ web client – ը բացում է նոր միացում, իսկ response – ը ստանալուց հետո փակում է միացումը։ Այսինքն կլիենտը և սերվերը փոխազդում են իրար հետ միայն request – ի և response – ի ժամանակ։
* Http – ն կախված չէ միջավայրից, այսինքն Http պրոտոկոլով կարելի է ուղարկել և ստանալ ցանկացած տիպի տվյալ, եթե web client – ը և սերվերը կարող են մշակել այդ տվյալները։ Կլիենտը և սերվերը պետք է նշեն կոնտենտի տիպը (MIEM տիպը` Multipurpose Internet Mail Extensions, հանդիսանում է ստանդարտ որը բնութագրում է տվյալի տիպը։ Ունի ստանդարտների սպեցիֆիկացիա)։
* Http – ն stateless է, այսինքն սերվերը և կլիենտը փոխազդում են իրար հետ միայն request – ի և response – ի ժամանակ և իրար մասին, այդ թվում նաև request – ների մասին ինֆորմացիա չեն պահում։

Http պրոտոկոլը դա request/response – ի պրոտոկոլ է և հիմնված է client/server իրար հետ փոխազդեցության մեխանիզմի վրա։ Բրաուզերը գործում է որպես web կամ http կլիենտ։

Կլիենտը ուղարկում է request սերվերին` request – ի տիպի և url – ի տեսքով։ Request – ը իր մեջ պարունակում է նաև MIME որը պարունակում է կլիենտ, փոխացնվող տվյալների մասին ինֆորմացիա, նաև պրոտոկոլի վերսիայի մասին։

Սերվերը ուղարկում է response դրա մեջ ներառելով տվյալները, ստատուս կոդը, նաև response – ի տվյալների մասին մետաինֆորմացիա։

Կլիենտը դա միջնորդ է սերվերի և user – ի միջև։ Որպես կլիենտ դիտարկում ենք բրաուզերը։ Կլիենտը HTTP request ուղարկելւ նպատակով կապ է հաստատում սերվերի հետ։ Սերվերը իրենից ներկայացնում է որևէ ռեսուրս որը ունի մեքենայական մաս և ծրագրային մաս (web server application): Web server application – ը ընդունում է request- ը և կլիենտին ուղարկում է HTTP response:

**Connection**

**TCP/IP**

HTTP – ն կիրառական մակարդակի պրոտոկոլ է՝ այսինքն ցանցային կապի մանր դետալները իր համար կարևոր չեն։ Կապի դետալները HTTP – ն թողնում է ցանցային պրոտոկոլի հայեցողությանը։

Ինտերնետը հիմնված է TCP/IP – ի վրա որը ցանցային պրոտոկոլների հավաքածու է (ստեկ), այսինքն տվյալների փոխանցման կանոններ։

Ցնացի տեսանկյունից ամենավերևի մակարդակում աշխատում է HTTP – ն դրա տակից տեղափոխման պրոտոկոլ TCP – ն (transmision control protocol այս մակարդակում ավելանում է պորտը) դրա տակից ցանցային պրոտոկոլ IP – ն(համակարգչից համակարգիչ առանց պորտը նշելու)։

Ցանկացած համակարգիչ ինտերնետ ցանցի շրջանակում ունի ունիկալ IP(Internet protocol) հասցե: Բայց քանի որ տվյալներով փոխանակվում են ոչ թե համակարգիչները այլ դրանց վրա աշխատող application – ները մեզ ip – ից բացի պետք է նաև application – ի հասցեն (համարը / port): Օր web service – ը լսում է 80 port – ն։ ip – ն և port – ը միասին կոչվում են սոկետ:

HTTP- ն օգտագործում է URI (Uniform Resource Identifier) ռեսուրսի նույնականացման և դրա հետ կապ հաստատման համար։

URI = "http:" "//" host [ ":" port ] [ abs\_path [ "?" query ]]

Uri – ն իր մեջ ներառում է URL(uniform resource location) և URN - uniform resource name)

Սերվերի IP հասցեն ստանալու համար օգտագործվում է URL – ը(հոստը), որը նույն IP - ի թվային արտահայտությունն է կոնվերտացված տքեստի։ Հոստերի անունները կոնվերտացվում են IP հասցեների Domain Name Service կամ կրճատ DNS – ի միջոցով։

UDP

**HTTP messages**

Http client Http server փոխազդեցությունը՝ տվյալների փոխանցումը, տեղի է ունենում HTTP – ում պարունակվող տվյալների բլոկներով որոնք կոչվում են Http message – ներ։

Http message – ը իրենից ներկայացնում է սիմվոլային հաջորդականություն այսինքն տեքստ: Կան 2 տիպի message – ներ request message և response message

HTTP message – ների միջոցով տվյալներ են փոխանցվում կլիենտի և սերվերի միջև։ HTTP message – ները և request–ի և response-ի դեպքերում փոխանցվում են մեկ ընդհանուր ֆորմատով (RFC 822), որը իր մեջ պարունակում է 4 բաղադրիչ:

* Message – ի սկզբնական տողը որը բնութագրում է request – ը կամ response – ի ստատուսը` այսինքն թե ինչ է պետք անել request – ի հետ, կամ ինչ է տեղի ունեցել response – ի հետ։
* Header fields որոնք կարող են լինել 0 և ավելի քանակով և իրենցից ներկայացնում են name value առանձնացված ։ - ով: Header – ներով ավելացվում են լրացուցիչ տվյալներ request – ին կամ response – ին։
* Դատարկ տող որով նշվում է header – ների ավարտը
* Body (optional է) որը HTTP – ի օգտակար բեռն է կարող է պարունակել ցանկացած տվյալ, request – ների body – ն փոխանցում է տվյալ սերվերին իսկ սերվերը body – ով կլիենտին է հետ վերադարձնում տվյալներ։ Ի տարբերություն սկզբնական տողի և headr – ների որոնք տեքստեր են body – ն տեքստից բացի կարող է պարունակել նաև բինար data(նկար, վիդեո, ծրագրային app - ներ և այլն)

**Request messages**

Start line (request line) – ը իր մեջ պարունակում է մեթոդը URI և վերսիան

Մեթոդի անունը պետք է լինի մեծատառերով։

Uri – ն կարող է լինել հետևյալ ձևերի՝

«\*» գտագործվում է երբ request – ը ուղղված է ոչ թե կոնկրետ ռեսուրսի այլ հենց սերվերին։ Թույլատրված է եթե մեթոդը ռեսուրսին չի վերաբերում օր Options \* HTTP/1.1

absoluteURI երբ ուղարկվում է ամբողջ URI – ն հոստի հետ միասին

abs\_path հասցեն առանց հոստի

**Respone messages**

Start line (Status-Line) = HTTP-Version Status-Code Reason-Phrase

**HTTP version**

Http – ի վերսիայի նումերացիան կատարվում է <major><minor> սխեմայով

0.9 – support է անում միայն get, support չի անում MIME

1 1.0, 1.1 վերսիաներում կատարվել են օպտիմիզացիաներ, սեմանտիկ բարելավումներ: Լայն տարածում ունի 1․1 բայց կա արդեն 2․0 և 3․0 – ի պրոտոտիպը։

**HTTP methods**

Հարկ է նշեն որ բոլոր մեթոդների անունները զգայուն են ռեգիստրին և պետք է լինեն միայն մեծատառերով։

Առանձնացնում են այսպես կոչված safe մեթոդներ GET HEAD OPTIONS որոնց արդյունքում սերվերի վրա ոչինչ չի փոխվում։

GET – օգտագործվում է նշված URL - ով սերվերից տվյալներ ստանալու համար։ Պարամետրերը փոխանցվում են URL – ի միջոցով GET request – ները պետք է միայն տվյալներ ստանան և որևէ կերպ չպետք է ազդեն տվյալների վրա։

HEAD – մեթոդը աշխատում է GET – ի նման այն տարբերությամբ որ վերադարձնում է միայն status line – ը և header- ները, առանց body - ի։ Կարող է օգտագործվել ռեսուրսի մասին

PUT – օգտագործվում է սերվերի վրա տվյալներ ավելացնելու / գրելու համար։ Սերվերը վերցնում է request – ի body – ն և օգտագործում է այն նոր data ստեծելու կամ արդեն գոյություն ուեցող data - ն փոխելու համար։

POST – օգտագործվում է տվյալները սերվերին ուղարկելու ժամանակ, օրինակ վերցված HTML ֆորմաներից։

TRACE – օգտագործվում է HTTP request – ի պարունակությունը տեսնելու համար։ HTTP request – ը մինչև սերվերին հասնելը կարող է անցնել պրոքսի սերվերներով, շլյուզներով և այլ։ Դրանցից յուրաքանչյուրը կարող է փոխել HTTP request – ը։ TRACE մեթոդը կլիենտին թույլ է տալիս տեսնել request – ի վերջնական տեքը, երբ այն հասնում է սերվերին։

OPTIONS – օգտագործվում է սերվերի կողմից support արվող պարամետրերի մասին տվյալներ ստանալու համար։ Օրինակ option - ով կարելի է ստանալ թե սերվերը ինչ մեթոդներ է support անում։

DELETE – օգտագործվում է սերվերից որևէ տվյալ ջնջելու համար։

CONNECT – օգտագործվում է սերվերի հետ կապ հաստատելու համար

Իդեմպոտենտություն։ HTTP մեթոդը համարվում է իդեմպոտենտ երբ նույն request – ը որքան անգամ էլ կատարվի միշտ ունենում է միևնույն արդյունքը և չի փոխում սերվերի վիճակը։ Այսինքն իդեմպոտենտ մեթոդը չպետք է ունենա կողմնակի ազդեցություններ։ Այս տեսանկյունից իդեմպոտենտ են բոլոր մեթոդները բացի post – ից։ Իդեմպոտենտությունը կախված է չէ վերադարձվող ստատուս կոդից նույնիսկ եթե դրանք տարբեր են օր մեկ անգամ delete – ից հետո մյուս նույն delete –ի request ները կվերադարձնեն 404 բայց DELETE – ը համարվում է իդեմպոտենտ

**HTTP Status Codes**

Սերվերի response – ում status code էլեմենտը իրենից ներկայացնում է եռանիշ թիվ որտեղ առաջին թիվը սահմանում է status – code – ի տեսակը։ Բուլոր ստատուս կոդերը ունեն նաև տեքստային նկարագրություն։

1 – ով սկսվող կոդերը ինֆորմատիվ ստատուս կոդերն են։

2 – ով սկսվող կոդերը նշանակում են որ request – ը հաջողությամբ մշակվել է։

**HTTP header fields**

Header field – երը նախատեսված են request – ին կամ response – ին լրացուցիչ տվյալներ ավելացնելու համար։

Գոյութոյւն ունեն header – ների 4 տեսակներ

General-header – որոնք ունեն նույն կիրառությունը և request-ների և response – ների համար

Client request-header – որոնք կիրառելի են միայն request – ների համար

Server response-header - որոնք կիրառելի են միայն response – ների համար

Entity-header – որոնք սահմանում են entity – ի body – ի մասին մետաինֆորմացիա, իսկ եթե body – ն բացակայում է ապա ռեսուրսի մասին

**General headers**

Cache-Control – օգտագործվում է դիրեկտիվների սահմանման համար, որնց պետք է ենթարկվի ամբողջ քեշավորման մեխանիզմը։ և սերվերը և կլիենտը կարող են սահմանել cache-control header – ը և դրան տալ պարամետրեր։ քեշավորման դիրեկտիվները նշվում են հաջորդաբար և առանձնացվում են ստորակետներով։

**Cache-Control: Request headers**

No-cache – քեշը չպետ է օգտագործի response - ը հաջորդ request – ի ժամանակ առանց սերվերի վրա ստումից հետո։

No-store – քեշը request – ի կամ response - ի մասին ոչինչ չպետք է պահի։

max-age = seconds կլիենտը պատրաստ է ընդունել response որի age – ը չի գերազանցում նշված արժեքը վայրկյաններով։

max-stale [=seconds] կլիենտը պատրաստ է ընդունել response – ը որը արդեն expired է եղել, բայց որի expired – ի ժամկետը չի անցնում նշված արժեքը վայրկյաններով։

min-fresh=seconds կլիենտը պատրաստ է ընդունել response որի կյանքի տևողությունը (expired) փոքր չէ նշված արժեքից վայրկյաններով։

no-transform չի կոնվերտացնում entity – ի body – ն։

only-if-cached նոր տվյալներ չի դուրս բերում, տվյալը ստացվում է միայն եթե այն քեշավորված է և նոր վերսիայի ստուգման համար ելակետային սերվերի հետ կապ հաստատելու կարիք նույնպես չկա։

**Cache-Control: Response headers**

Public - ցույց է տալիս որ response – ը կարող է քեշավորվել ցանկացած քեշի կողմից

Private - ցույց է տալիս որ response – ը կամ դրա մի մասը նախատեսված է միայն մեկ կլիենտի համար և չպետք է քեշավորվեն ընդհանուր քեշով

No-cache – քեշը չպետ է օգտագործի response - ը հաջորդ request – ի ժամանակ առանց սերվերի վրա ստումից հետո։

No-store – քեշը request – ի կամ response - ի մասին ոչինչ չպետք է պահի։

no-transform չի կոնվերտացնում entity – ի body – ն։

must-revalidate – քեշը պետք է ստուգի հնացած տվյալների ստատուսը դրանց օգտագործումից առաջ, իսկ expire – ից հետո տվյալը չպետք է օգտագործվի։

proxy-revalidate նույնն է ինչ որ must-revalidate – ը այն տարբերությամբ որ ․․․․․․․

max-age = seconds կլիենտը պատրաստ է ընդունել response որի age – ը չի գերազանցում նշված արժեքը վայրկյաններով։

s-maxage = seconds օգտագործվում է max-age – ով կամ expires – ով սահմանված maxage – ը override անելու համար

**Connection**

Կլիենտին թույլ է տալիս սահմանել կապի կարգավորումնր

Connection: close – նշանակում է որ կապը կփակվի response – ը ստանալուց հետո

Connection: keep-alive – դեֆոլտով HTTP 1․1 – ն օգտագործում է մշտական կապի պահպանում, իսկ 1․0 – ի դեպքում պետք է նշել keep-alive header – ը

**Date**

**Pragma**