

Адресація та передача даних в Інтернет

(Internet addressing and data transfer)

Ihor Kapatsila

softserve

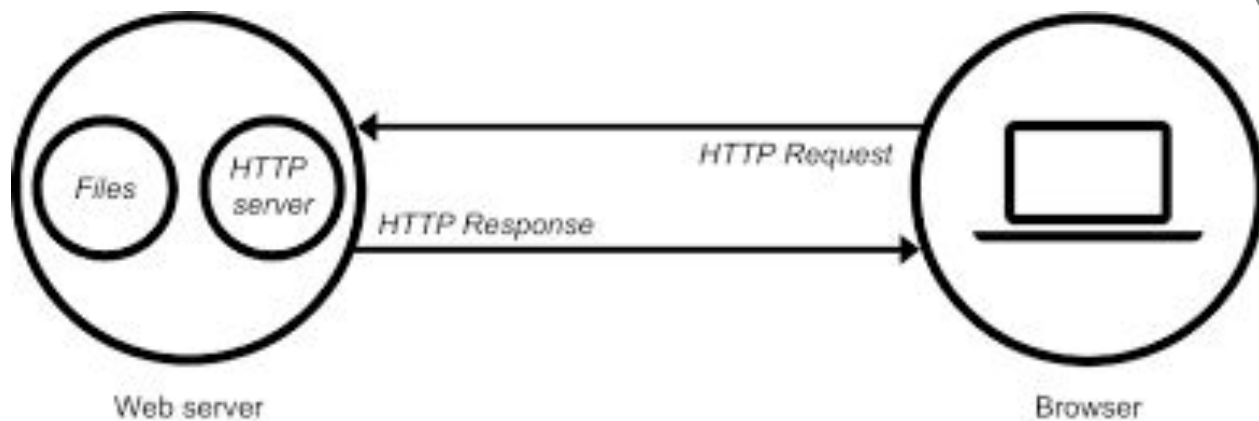


Як працює мережа ?

Базові уявлення

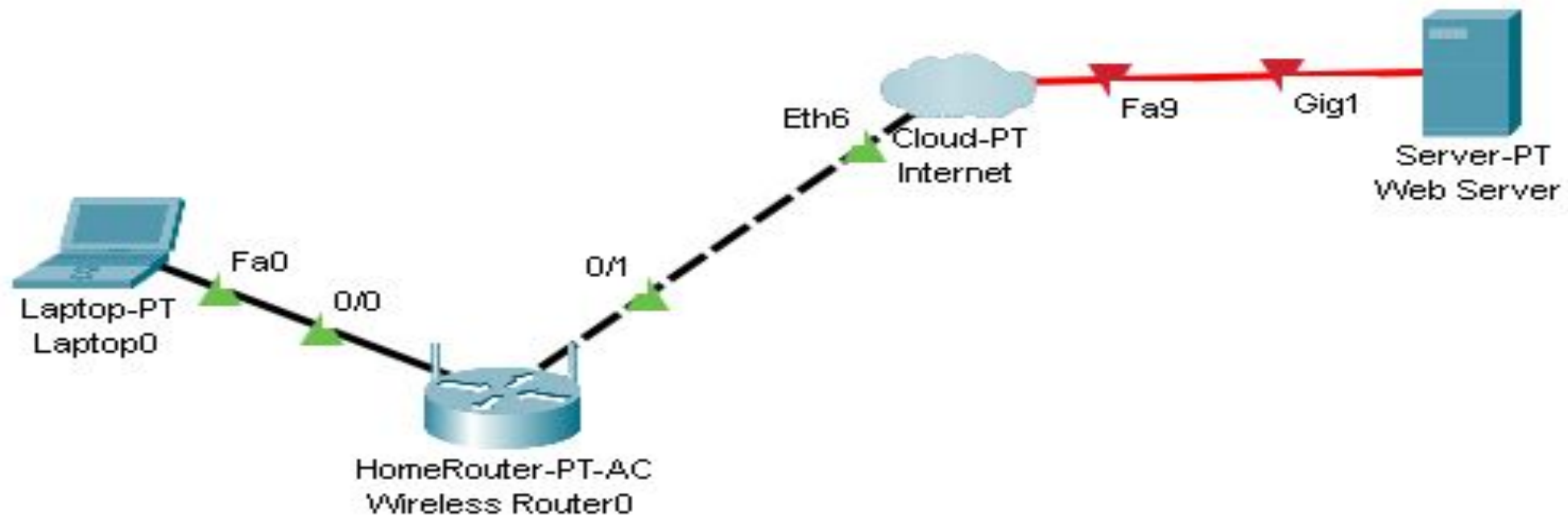
- Відкрили браузер
- ввели адресу
- ENTER
- Є веб сторінка!!!

Чи може все трохи складніше?





Як воно працює?



IP адресація

- IP адреса – це адреса пристрою в Інтернет.

IP v4

32 бітна адреса
10.2.3.15.25

Маска

Числова

255.255.255.0

Або бітова /24

IPv6

128 бітна адреса
2dfc:0:0:0:0217:cbff:fe8c:0

Маска бітова

Порти

- Порт є кінцевою точкою в глобальній адресації вузлів в мережі
- 127.0.0.1:8080 (127.0.0.1:3000)
- WWW – 80
- FTP – 21
- SSH – 22
- SMTP – 25
- DNS - 53
- POP3 - 110
- SSL - 443
- MySQL - 3306
- PostgreSQL - 5432
- MongoDB - 27017 i 28017

DNS

Domen Name System

Доменні імена верхнього рівня відомі всій мережі

<https://www.softserveinc.com>

Територіальний признак

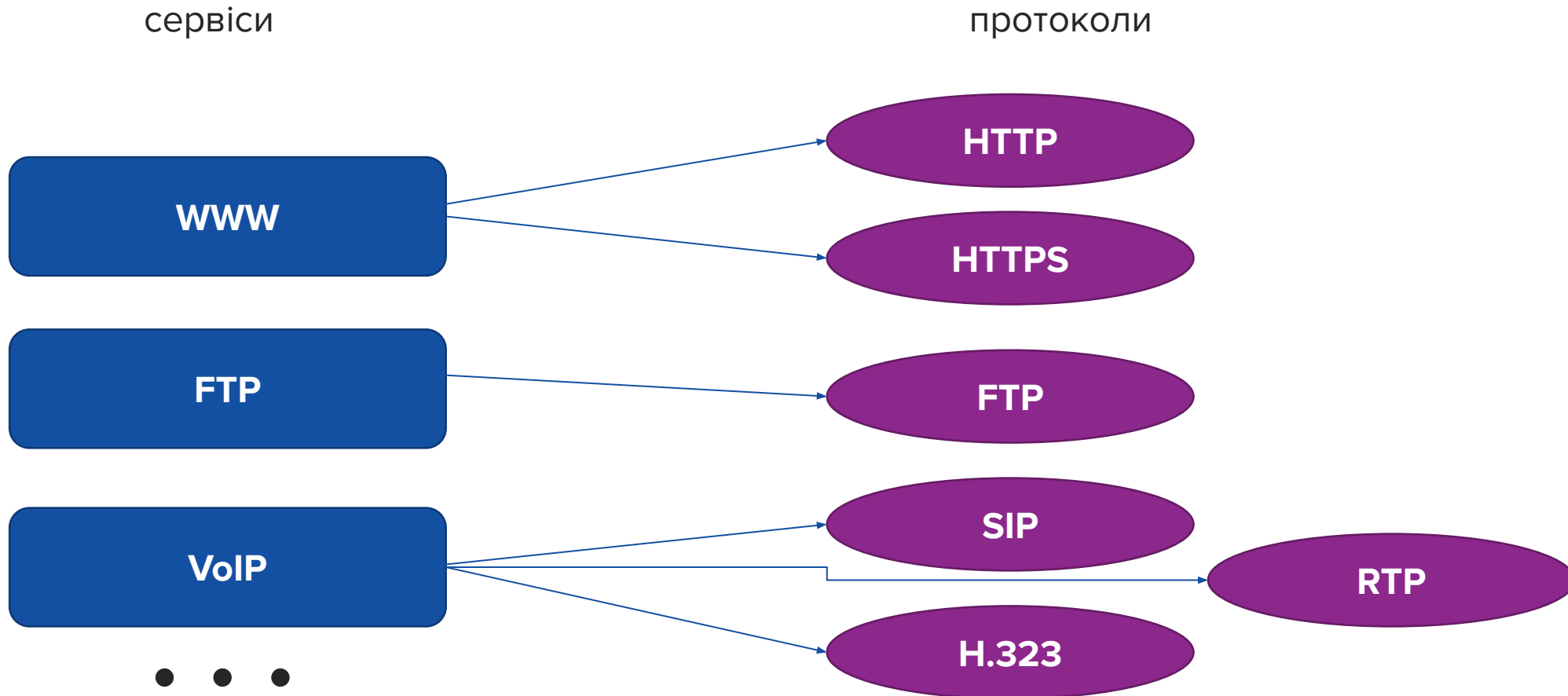
- ua
- pl
- us
- uk

Організаційний признак

- com
- org
- gov
- edu
- mil
- net

Доменні імена 1, 2, 3 рівня

Сервіси Інтернет



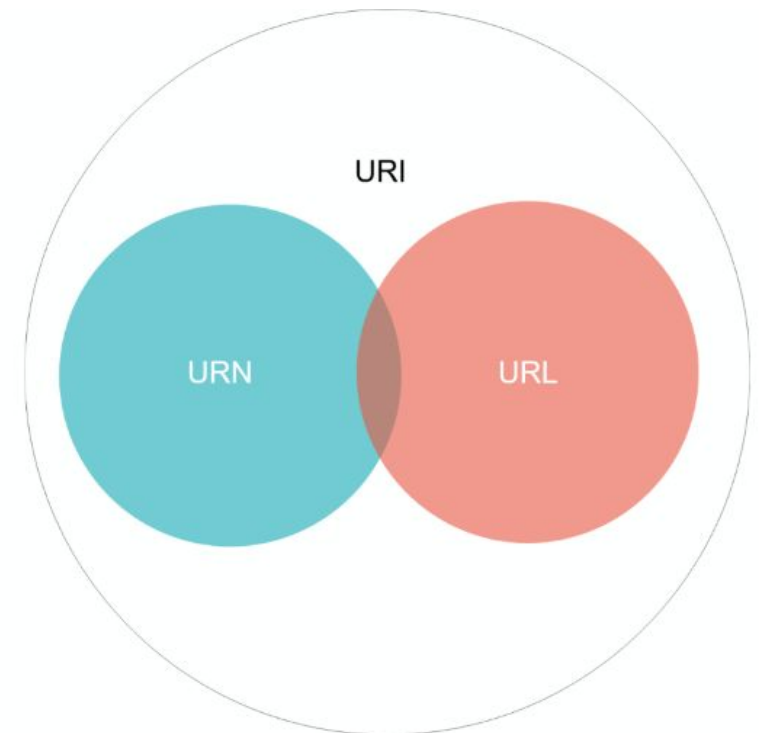
Посилання

(вони ж запити до серверу)

- **URI - Uniform Resource Identifier** (уніфікований **ідентифікатор** ресурсу)
 - **URL - Uniform Resource Locator** (уніфікований визначник **місця** знаходження ресурсу)
 - **URN - Uniform Resource Name** (уніфіковане **ім'я** ресурсу)
-
- **URL - <https://www.softserveinc.com/>**
 - **URI - <https://www.softserveinc.com/cdn/img/home/leaves-illustration.png>**
 - **URN - [cdn/img/home/leaves-illustration.png](#)**

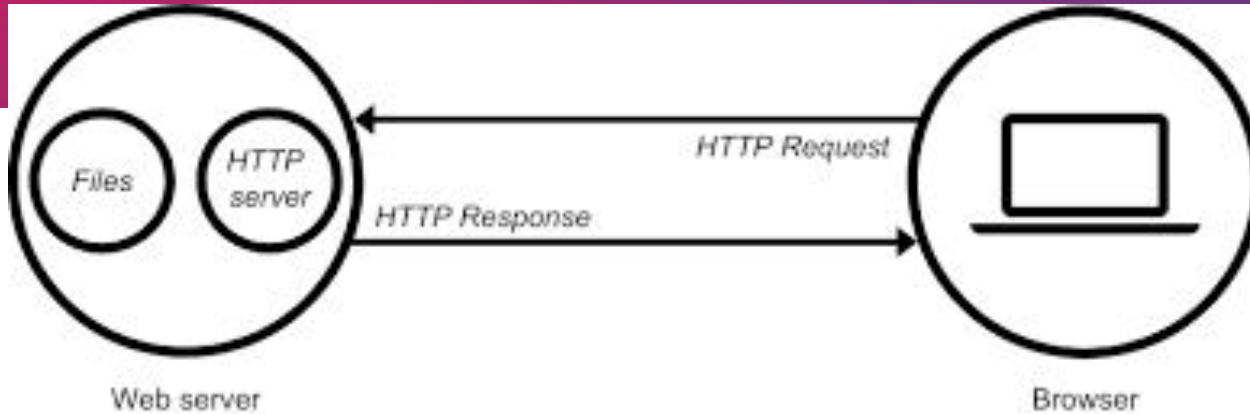
Короткі URI

<http://example.com/good-uri-design>



Як передати дані?

Простий приклад з html



1. `<body>`
2. `<div id="calcmain">`
3. `<h1 id="calcheading">CALCULATOR Body Mass Index</h1>`
4. `</div>`
5. `<form action="/calc" method="post">`
6. `<input type="text"`
7. `name="Name"`
8. `placeholder="Enter your name!" />`
9. `<button class="btn" type="submit">Get result</button>`
10. `</form>`
11. `</body>`

Методи передачі даних

GET

- Відправляє скрипту всю зібрану інформацію форми як частину URL

<https://www.example.ua/script.php?page=1&name=user>










- Обмежує об'єм інформації при передачі (*максимальна довжина URL — 1024 символи*)
- Може пагубно впливати на безпеку даних
- Можна зробити закладкою браузера
- Сторінка з параметрами може кешуватися
- Підтримувані типи кодування :
application / x-www-form-urlencoded

POST

- Користувач сайту не бачить дані, котрі передає скрипту

<https://www.example.ua/script.php>

- Дозволяє передавати файли
- Відносно безпечний
- Неможливо зробити закладкою
- Сторінка з параметрами не кешується
- Підтримувані типи кодування:
application / x-www-form-urlencoded
multipart / form-data
text / plain

Ситуація	GET	POST
Фільтр товарів	 Користувач отримає сторінку з відповідними йому товарами, зможе зберегти її в закладки, переслати посилання на сторінку з параметрами іншим і повернутися на сторінку пізніше без повторного заповнення форми фільтра.	 Для повторного відвідування сторінки користувач повинен буде повторно заповнити форму фільтра, сторінкою з параметрами можна буде поділитися, зберегти в закладки і повернутися на сторінку пізніше без повторного заповнення форми фільтра.
Форма авторизації	 Відсутній захист конфіденційної інформації. Введений пароль буде видно в адресному рядку браузера, буде збережений в історії відвіданих сайтів.	 Хоча дані можуть передаватися в незашифрованому вигляді, побачити їх можна тільки через інструменти розробника або за допомогою іншого спеціального програмного забезпечення.
Онлайн заявка Оформлення замовлення Форма зворотнього зв'язку	 При повторному зверненні за посиланням на сервері буде проведена повторна обробка, наприклад, буде створена повторна заявка, оформлене ще одне замовлення, створений ще один запит на зворотний зв'язок.	 Повторне звернення за посиланням не призведе до повторної обробки запиту з введеними раніше параметрами.
Перехід за гіперпосиланням	  Перехід за гіперпосиланням з параметрами рівнозначний відправці запиту через HTML форму.	 Немає технічної можливості помістити POST запит в гіперпосилання.
AJAX запити	Використовуються обидва методи. Вибір залежить від контексту. Принципи вибору методу такі ж, як і для HTML форм.	

Типи даних

- *application/x-www-form-urlencoded*: Значення кодуються в кортежі з ключем, розділених символом '&', з '=' між ключем і значенням. Не буквено-цифрові символи кодуються відсотковим кодуванням (# - %23): це причина, по якій цей тип не підходить для використання з двійковими даними (замість цього використовуйте multipart/form-data)
- *multipart/form-data*: Кожне значення посилається як блок даних ("body part"), з певним призначенням клієнтом користувача роздільником ("boundary"), що розділяє кожну частину. Ці ключі даються в заголовки Content-Disposition кожній частини.
content-type: multipart/form-data; boundary=-----590299136414163472038474
- *text/plain*
- Raw json
Content-Type: application/json
{"key1":"value1","key2":"value2"}

Коди відповідей

Код відповіді (стану) HTTP показує, чи був успішно виконаний певний HTTP запит.

Коди згруповані в 5 класів:

1. Інформаційні 100 - 199
2. Успішні 200 – 299
3. Перенаправлення 300 - 399
4. Клієнтські помилки 400 - 499
5. Серверні помилки 500 – 599

Найбільш часто відображаємі:

Код	
200	OK – Успішна операція
202	Accepted - Прийнято
400	Bad Request
403	Forbidden
404	Not Found
500	Internal Server Error

REST

REST означає REpresentational State Transfer (Вікіпедія: «передача стану представлення»). Це популярний архітектурний підхід для створення API в сучасному світі.

- Метод, який використовується в HTTP-запиті, вказує, які дії ви хочете виконати з цим запитом. Важливі приклади:

- **GET** : отримати детальну інформацію про ресурс
- **POST** : створити новий ресурс
- **PUT** : оновити існуючий ресурс
- **DELETE** : видалити ресурс
- **HEAD** : запит заголовку
- **OPTIONS** : опис параметрів з'єднань з цільовим ресурсом.
- **PATCH** : аналог UPDATE в CRUD
- **TRACE** : виконує перевірку зворотнього зв'язку до цільового ресурсу

Код стану завжди присутній у відповіді HTTP. Типові приклади:

200 - успіх

404 - сторінку не знайдено

Детальніше: <https://developer.mozilla.org/ru/docs/Web/HTTP/Methods>

Розширена інформація про помилку при REST підході

IETF розробив [RFC 7807](#), який описує узагальнену схему обробки помилок

схема складається з п'яти частин:

1. *тип* - ідентифікатор URI, який класифікує помилку
2. *заголовок* - коротке, зрозуміле для читання повідомлення про помилку
3. *status* - код відповіді HTTP (необов'язково)
4. *деталь* - зручне для читання пояснення помилки
5. *екземпляр* - URI, який ідентифікує конкретне виникнення помилки

Наприклад:

```
{ "type": "/errors/incorrect-user-pass",  
  "title": "Incorrect username or password.",  
  "status": 401,  
  "detail": "Authentication failed due to incorrect username or password.",  
  "instance": "/login/log/abc123"  
}
```

Або

```
curl -X GET https://api.twitter.com/1.1/statuses/update.json?include_entities=true
```


Запитання?



**FOR
YOUR
FUTURE**

Дякую увагу!

softserve