Робота з формами.
Відправка даних форми.

Сьогодні ми розглянемо, що відбувається, коли користувач відправляє форму – куди передаються дані і як ми їх обробляємо, коли вони туди потрапляють? Ми також розглянемо деякі проблеми безпеки, пов'язані з відправкою даних форми. Для цього нам треба пригадати, що таке HTML, HTTP і програмування з боку сервера.

HTML - за своєю суттю це досить проста мова, що складається з елементів, які можна застосовувати до фрагментів тексту, щоб надати їм різного вигляду в документі (*це абзац? Це маркований список? Це частина таблиці?*); структурувати документ за логічними розділами (чи є у нього заголовок? Три стовпці контенту? Меню навігації?); додавати контент (наприклад зображення або відео) на сторінку.

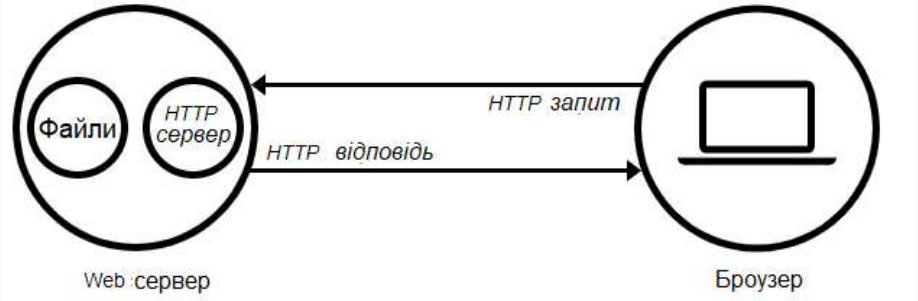
HTTP - це розширюваний протокол, заснований на таких поняттях, як ресурси і уніфіковані ідентифікатори ресурсів (URI), проста структура повідомлень і потік зв'язку клієнт-сервер.

Програмування з боку сервера - Термін «веб-сервер» може відноситися як до апаратної начинки, так і до програмного забезпечення. Або навіть до обох частин, які працюють спільно.

1. З точки зору "заліза", "веб-сервер» - це комп'ютер, який зберігає файли сайту (HTML-документи, CSS-стилі, JavaScript-файли, картинки та інші) і доставляє їх на пристрій кінцевого користувача (веб-браузер і т . д.). Він підключений до мережі Інтернет і може бути доступний через доменне ім'я.

2. З точки зору програмного забезпечення, веб-сервер включає в себе кілька компонентів, які контролюють **доступ** веб-користувачів до розміщених на сервері файлів, як мінімум - це HTTP-сервер. HTTP-сервер - це частина програмного забезпечення, яка розуміє URL'і (веб-адреси) і HTTP (протокол, який ваш браузер використовує для перегляду веб-сторінок).

На самому базовому рівні, коли браузеру потрібен файл, розміщений на веб-сервері, він запитує його через HTTP-протокол. Коли запит досягає потрібного веб-сервера ("залізо"), сервер HTTP (ПЗ) приймає запит, знаходить запитуваний документ (якщо немає, то повідомляє про помилку 404) і відправляє назад, також через HTTP.



Щоб опублікувати веб-сайт, необхідний або статичний, або динамічний веб-сервер.

Статичний веб-сервер, або стек, складається з комп'ютера ("залізо") з сервером HTTP (ПЗ). Його називають статичним, бо *сервер посилає розміщені файли в браузер «як є»*.

Динамічний веб-сервер складається з статичного веб-сервера і додаткового програмного забезпечення, найчастіше сервера додатків і бази даних. Його називають «динамічним», тому що сервер додатків *змінює вихідні файли перед відправкою в ваш браузер* за HTTP.

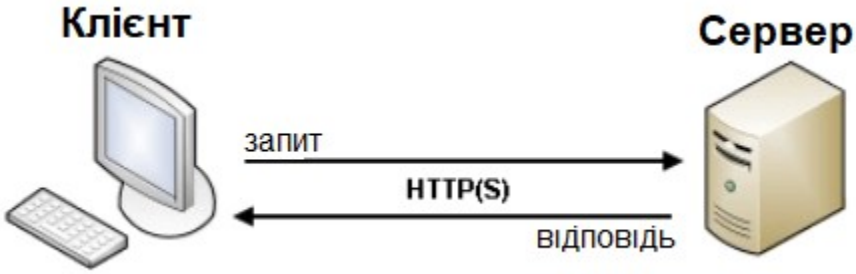
Наприклад, для отримання фінальної сторінки, яку ви переглядаєте в браузері, сервер додатків може заповнити HTML-шаблон даними з бази даних. Деякі сайти, складаються з тисяч веб-сторінок, які є не реальними HTML документами, а лише декількома HTML-шаблонами і гігантських баз даних. Така структура спрощує і прискорює супровід веб-додатків і доставку контента.

Куди відправляються дані?

Спробуємо з’ясувати куди ж відправляються дані.

Про клієнтську і серверну архітектуру

WEB базується на дуже простий клієнт-серверній архітектурі, яку можна узагальнити так: клієнт (зазвичай веб-браузер) відправляє запит на сервер (зазвичай веб-сервер, такий як Apache, Nginx, IIS, Tomcat, і т. д.), використовуючи протокол HTTP. Сервер відповідає на запит, використовуючи той самий протокол.



З боку клієнта HTML-форма - це не більше ніж зручний спосіб налаштування HTTP-запиту для відправки даних на сервер. Це дозволяє користувачеві надавати інформацію для доставки в HTTP-запиті.

На стороні клієнта: визначення способу відправки даних

Елемент <form> визначає спосіб відправки даних. Всі його атрибути призначені для того, щоб ви могли налаштувати запит на відправку, коли користувач натискає кнопку відправки. Двома його найбільш важливими атрибутами є action і method.

Атрибут action - визначає, куди відправляються дані. Його значення має бути дійсною URL. Якщо цей атрибут не вказаний, дані будуть відправлені на URL-адресу сторінки, що містить форму.

В цьому прикладі дані відправляються на абсолютну URL - http://foo.com:

```
<form action="http://foo.com">
```

Тут використовується відносна URL - дані відправляються на іншу URL на сервері:

```
<form action="/somewhere_else">
```

Якщо атрибути не вказані, як показано нижче, дані з форми <form> відправляються на ту ж сторінку, на якій розміщується дана форма:

```
<form>
```

Багато старих сторінок використовують наступний синтаксис, щоб вказати, що дані повинні бути відправлені на ту ж сторінку, яка містить форму; це було необхідно, бо до появи HTML5 атрибут action був обов'язковим.

```
<form action="#">
```

Наразі це більше не потрібно.

До уваги: Можна вказати URL, який використовує протокол HTTPS (безпечний HTTP). В такому випадку, дані шифруються разом з іншою частиною запиту, навіть якщо сама форма розміщується на небезпечній сторінці, доступ до якої здійснюється через HTTP. З іншого боку, якщо форма розміщується на захищеній сторінці, але ви вказуєте небезпечну URL-адресу HTTP з атрибутом action, всі браузери видають користувачеві попередження про небезпеку щоразу при відправленні даних, оскільки дані не шифруються.

Атрибут method - визначає спосіб відправки даних.

Протокол HTTP надає кілька способів виконати запит; Дані HTML-форми можуть передаватися кількома різними способами, найбільш поширеними з яких є метод GET і метод POST.

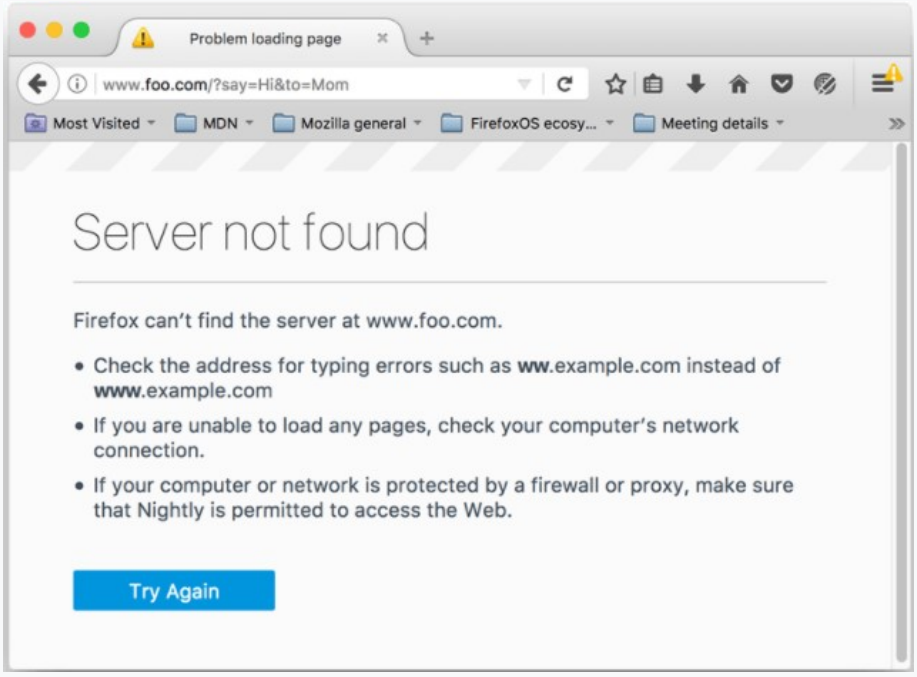
Щоб зрозуміти різницю між цими двома методами, давайте повернемося назад і згадаємо, як працює HTTP. Кожен раз, коли ви хочете отримати доступ до ресурсу в Інтернеті, браузер відправляє запит на URL-адресу. Цей HTTP-запит складається з двох частин: *заголовка*, який містить набір глобальних метаданих про можливості браузера, і *тіла*, яке може містити інформацію, необхідну серверу для обробки конкретного запиту.

Метод GET - це метод, який використовується браузером, щоб повідомити серверу, що *потрібно відправити назад даний ресурс*: «Ей, серверу, я хочу отримати цей ресурс». В цьому випадку браузер відправляє порожнє тіло. Оскільки тіло порожнє, і форма відправляється з використанням даного методу, то дані, що відправляються на сервер, додаються до URL-адреси.

Розглянемо наступну форму:

```
<form action="http://foo.com" method="get">
  <div>
    <label for="say">What greeting do you want to say?</label>
    <input name="say" id="say" value="Hi">
  </div>
  <div>
    <label for="to">Who do you want to say it to?</label>
    <input name="to" id="to" value="Mom">
  </div>
  <div>
    <button>Send my greetings</button>
  </div>
</form>
```

Оскільки використовується метод GET, ви побачите URL www.foo.com/?say=Hi&to=Mom, який з'явиться в адресному рядку браузера при відправці форми.



Дані додаються в URL як послідовність пар «ім'я / значення». Після того, як URL веб-адреси закінчилася, ми додаємо знак питання (?), за ним йдуть пари «ім'я / значення», кожна з яких розділена амперсандом (&). В цьому випадку ми передаємо дві частини даних на сервер:

say, зі значенням «Hi»

to, зі значенням «Mom»

HTTP-запит має наступний вигляд:

```
GET /?say=Hi&to=Mom HTTP/2.0
Host: foo.com
```

Метод POST діє трохи інакше. Браузер використовує цей метод для зв'язку з сервером при запиті відповіді з урахуванням даних, *представлених в тілі* HTTP-запиту: «Ей, серверу, поглянь на ці дані і відправ мені відповідний результат». Якщо форма відправляється з використанням цього методу, дані додаються в тіло HTTP-запиту.

Давайте розглянемо приклад - це та ж сама форма, яку ми розглядали вище, в розділі GET, але вже з атрибутом method, встановленим в post.

```
<form action="http://foo.com" method="post">
  <div>
    <label for="say">What greeting do you want to say?</label>
    <input name="say" id="say" value="Hi">
  </div>
  <div>
    <label for="to">Who do you want to say it to?</label>
    <input name="to" id="to" value="Mom">
  </div>
  <div>
    <button>Send my greetings</button>
  </div>
</form>
```

Коли форма відправляється з використанням методу POST, дані додаються ні до URL-адресою, а включаються в тіло запиту. HTTP-запит має наступний вигляд:

```
POST / HTTP/2.0
Host: foo.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

say=Hi&to=Mom
```

Заголовок Content-Length вказує розмір тіла, а заголовок Content-Type вказує тип даних, які відправляються на сервер.

Перегляд HTTP-запитів

HTTP-запити ніколи не відображаються користувачу (якщо ви хочете їх побачити, Вам потрібно використовувати такі інструменти, як наприклад Chrome Developer Tools або Firefox Network Monitor). Наприклад, дані форми можна побачити на вкладці Мережа (Network) в Chrome наступним чином (після відправки форми):

1. натисніть F12
2. Виберіть Network
3. Виберіть "All"

4. Виберіть "foo.com" у вкладці "Name"
5. Виберіть "Headers"
6. Потім ви можете отримати дані форми, як показано на малюнку нижче.

Єдине, що відображається для користувача - URL, яка викликається. Як обговорювалося раніше, запит з методом GET дозволить користувачеві побачити інформацію з запиту в URL, а запит з методом POST не дозволить. Можна назвати дві причини, чому це може бути важливим:

1. Якщо необхідно відправити пароль (або будь-яку іншу важливу інформацію), ніколи не використовуйте метод GET, інакше ризикуєте відобразити цю інформацію в URL-рядку, що небезпечно.
2. Якщо необхідно відправити великий обсяг інформації, POST-метод є кращим, тому що деякі браузери обмежують довжину URL. До того ж, багато серверів так само обмежують довжину оброблюваних URL.

На стороні сервера: отримання даних.

Який би HTTP ви не вибрали, сервер повертає рядок, який буде послідовно проаналізовано для отримання даних в форматі листа з парами «ключ / значення». Спосіб отримання доступу до цього листа залежить від платформи розробки або особливостей фреймворка, який використовується. Саме технології, які використовуються, визначають, як обробляються скопійовані ключі. Часто, пріоритетним є останнє отримане для даного ключа значення.

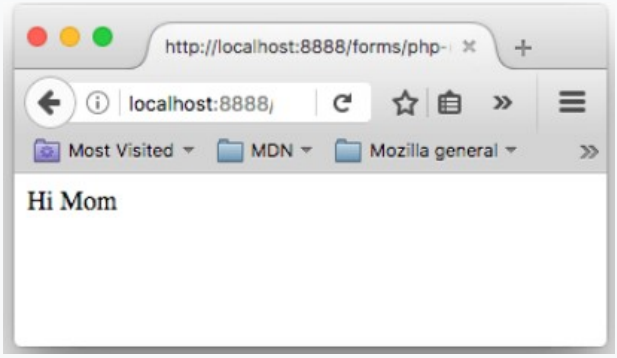
Приклад: Чистий PHP

PHP пропонує кілька глобальних об'єктів доступу до даних. Наприклад, при використанні POST-методу, в наведеному нижче прикладі, дані просто виходять і демонструються користувачеві. Зрозуміло, як використовувати такі дані - вирішувати тільки вам. Ви можете відобразити їх, помістити в базу даних, відправити поштою або передати куди-небудь ще.

```php
<?php
  // The global $_POST variable allows you to access the data sent with the POST method by name
  // To access the data sent with the GET method, you can use $_GET
  $say = htmlspecialchars($_POST['say']);
  $to  = htmlspecialchars($_POST['to']);

  echo  $say, ' ', $to;
?>
```

Приклад демонструє сторінку з даними, які були відправлені. Коли дані передані на відправку (submit), вони передані в форму php-example.php, яка містить PHP код з прикладу вище. Коли код буде виконаний, браузер виведе (output) оброблене повідомлення: Hi Mom.



Цей приклад не буде працювати, якщо завантажити його в браузер локально - **браузер не може інтерпретувати PHP код**, після відправки даних з форми, браузер просто запропонує завантажити PHP файл. Щоб приклад запрацював, необхідно відправити його на PHP сервер. Для тестування PHP на локальних серверах можете спробувати MAMP (Mac and Windows) і / або AMPPS (Mac, Windows, Linux).

Приклад: Python

Цей приклад показує, як можна використати Python для вирішення того ж завдання - відобразити відправлені дані на сторінці. У цьому прикладі використовується Flask framework для візуалізації шаблонів, що підтримують форму відправки даних

```python
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def form():
    return render_template('form.html')

@app.route('/hello', methods=['GET', 'POST'])
def hello():
    return render_template('greeting.html', say=request.form['say'], to=request.form['to'])

if __name__ == "__main__":
    app.run()
```

Два шаблону з коді вище взаємодіють наступним чином:

- form.html: Та ж форма, що і вище The POST method, тільки з використанням action до {{url_for ('hello')}}. (Це Jinja2 шаблон, який початково HTML, але може містити виклик Python-коду в фігурних дужках, який запуститься веб-сервером. Url_for ('hello') буквально говорить, що після відправки даних переадресує їх в / hello.)
- greeting.html: Цей шаблон просто містить рядок, яка відображає два біта даних, переданих йому при відображенні. Це зроблено за допомоги функції hello (), зазначеної вище, яка виконується, коли запит направляється в / hello URL.

Знову ж таки, цей код не буде працювати, якщо ви просто спробуєте завантажити його прямо в браузер. Python працює трохи інакше, ніж PHP - щоб запустити цей код, потрібно встановити Python / PIP, потім встановити Flask використовуючи команду: pip3 install flask. Після цього, ви зможете запустити файл з прикладу, використовуючи команду: python3 python-example.py, потім відкрити localhost: 5000 в своєму браузері.