11. Бекенд. Програмування серверної частини

Веб-додатки і більшість веб-сайтів використовують програмування серверної частини, щоб за потребою динамічно відображати різні дані, в основному взяті з бази даних. Серверна частина додатку розміщується на сервері і надсилає клієнту деякий код (наприклад, HTML, CSS і JavaScript) для відображення/

Найбільша користь програмування серверної частини – це можливість формувати контент веб-сайту під конкретного користувача. Динамічні сайти можуть формувати актуальний контент, відповідно до переваг і звичок користувача. Це спрощує використання сайтів за рахунок збереження особистих переваг та інформації, наприклад, повторного використання збережених даних кредитної картки для оптимізації наступних платежів.

Програмування серверної частини надає можливість взаємодіяти з користувачем сайту, надсилаючи повідомлення та оновлення електронною поштою або іншими каналами. Всі ці можливості дозволяють глибше взаємодіяти з користувачами.

Сенс програмування серверної частини додатку

Браузери взаємодіють з веб-серверами за допомогою протоколу HTTP. Запит від браузера містить URI, що однозначно визначає ресурс, метод, що визначає необхідну дію (наприклад, отримати, видалити або опублікувати ресурс) і може містити додаткову інформацію, закодовану в параметрах запиту або у файлі cookie .

Веб-сервери очікують повідомлень з клієнтськими запитами, обробляють і надсилають відповідь до браузера за допомогою відповідного HTTP повідомлення (HTTP-відповідь). Відповідь містить рядок стану, який показує, чи був запит успішним чи ні (наприклад, «HTTP/1.1 200 OK» у разі успіху).

Тіло успішної відповіді на запит може містити запитані дані (наприклад, HTML-сторінка, зображення тощо), які відображаються через веб-браузер.

Статичні сайти

Статичний сайт - це сайт, що повертає статичний вміст із сервера щоразу, коли запитується конкретний ресурс На рис.1. показано базову архітектуру веб-сервера для статичного сайту.

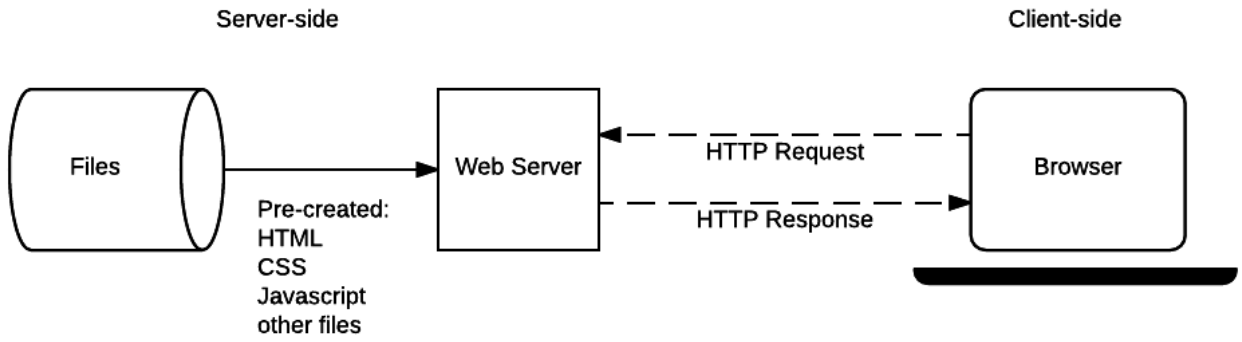

Рис.1. Спрощена схема статичного веб-сервера

Коли користувач хоче перейти на сторінку, браузер надсилає HTTP-запит "GET" із зазначенням URL сторінки.

Сервер витягує запитаний документ зі своєї файлової системи та повертає HTTP-відповідь, що містить документ та успішний статус (зазвичай 200 OK). Якщо файл не може бути надісланий з будь-яких причин, повертається статус помилки.

Динамічні сайти

Динамічний веб-сайт – це сайт чи додаток, де частина вмісту відповіді генерується динамічно лише за необхідності. На динамічному веб-сайті HTML-сторінки зазвичай створюються шляхом заповнення HTML-шаблонів даними із бази даних. Такий підхід є значно ефективним способом зберігання великої кількості контенту, ніж використання статичних сайтів.

Динамічний сайт може повертати різні дані URL-адреси на основі інформації, наданої користувачем або збереженими налаштуваннями, і може виконувати інші операції, як частина повернення відповіді (наприклад, відправлення повідомлень).

Більшість кодів для підтримки динамічного веб-сайту повинні виконуватися на сервері. Створення цього коду відомо, як "програмування серверної частини" або бекенд.

На рис.2 показано просту архітектуру динамічного сайту . Як і на попередній схемі, браузери надсилають HTTP-запити до серверу, сервер обробляє запити та повертає до браузеру HTTP-відповіді.

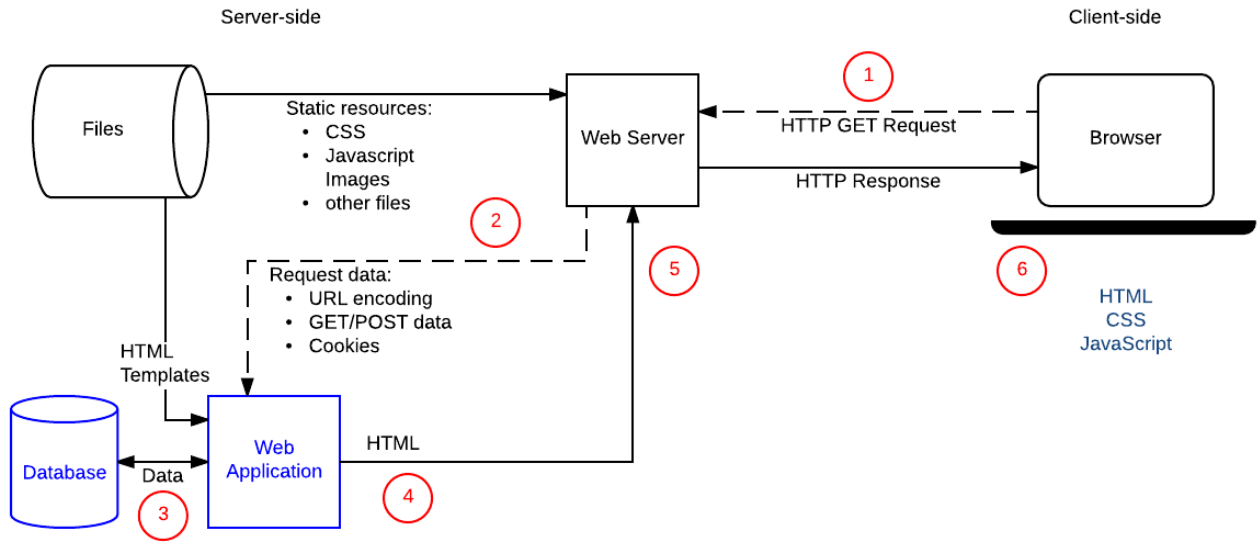

Рис.2. Спрощена схема динамічного веб-сервера

Запити на завантаження статичних ресурсів обробляються як і у випадку статичних сайтів Статичні ресурси — це файли, які є незмінними, зазвичай це: CSS, JavaScript, зображення, попередньо створені PDF-файли тощо).

Запити динамічних даних надсилаються (2) у код серверної частини (показано на діаграмі як Веб-додаток). Для «динамічних запитів» сервер інтерпретує запит, читає необхідну інформацію з бази даних (3), комбінує витягнуті дані з шаблонами HTML та повертає відповідь, що містить згенерований HTML (5, 6).

Різниця у програмуванні серверної та клієнтської частини

Коди, задіяні у серверній частині та клієнтській частині значно відрізняються:

- Вони мають різні цілі та призначення.

- Як правило, вони не використовують однакові мови програмування. Виняток складає JavaScript, який можна використовувати на стороні сервера та клієнта.
- Вони виконуються у різних середовищах операційної системи.

Код, який виконується в браузері, відомий як код клієнтської частини, перш за все пов'язаний з покращенням зовнішнього вигляду і поведінки веб-сторінки. Це включає вибір і стилізацію компонентів інтерфейсу користувача, створення макетів, навігацію, перевірку форм тощо.

Навпаки, програмування на стороні сервера в основному включає вибір вмісту, який повертається браузеру у відповідь на запити. Код на стороні сервера обробляє такі завдання, як перевірка надісланих даних та запитів, використання баз даних для зберігання та витягування даних та надсилання правильних даних до клієнта.

Код клієнтської частини написаний з використанням HTML, CSS та JavaScript. Він запускається у браузері і практично не має доступу до базової операційної системи (включаючи обмежений доступ до файлової системи).

Веб-розробники не можуть контролювати, який браузер може використовувати користувач для перегляду веб-сайту. Браузери забезпечують сумісність з функціями коду, і одним із завдань клієнтського програмування є забезпечення кросбраузерності сайту.

Код серверної частини може бути написаний на будь-якій кількості мов програмування - приклади популярних мов серверної частини включають PHP, Python, Ruby, C# і NodeJS. Код серверної частини має повний доступ до операційної системи сервера, і розробник може вибрати доцільну мову програмування.

Розробники зазвичай пишуть свій код, використовуючи веб-фреймворки. Веб-фреймворки - це набори функцій, об'єктів, правил та інших конструкцій коду, призначених для вирішення загальних проблем, прискорення розробки та спрощення різних типів завдань.

Фреймворки для розробки клієнтської і серверної частин є різні. Фреймворки клієнтської частини спрощують верстку та подання даних, тоді як фреймворки серверної частини забезпечують багато «звичайної» функціональності веб-сервера, яку в іншому випадку розробник мав здійснити самостійно (наприклад, підтримка сесій, підтримка користувачів та автентифікація, простий доступ до бази даних, шаблонів бібліотек тощо).

Примітка: Фреймворки клієнтської частини часто використовуються для прискорення написання коду клієнтської частини, але можна писати весь код власноруч. Інколи для створення невеликого простого сайту, написання власноруч коду може бути більш швидким та ефективним,.

І, навпаки, у разі написання коду серверної частини веб-додатку без фреймворку: здійснення життєво важливої функції, такої як HTTP сервер, складно зробити з нуля. Використання бекенд-фреймворків забезпечують це разом з іншими корисними інструментами.

Дії у серверній частині

Програмування серверної частини дозволяє ефективно доставляти актуальну інформацію, складену для індивідуальних користувачів. Нижче перелічено типові застосування та переваги бекенда.

Ефективне зберігання та доставка інформації

Програмування серверної частини дозволяє зберігати інформацію в базі даних і динамічно створювати та повертати HTML та інші типи файлів (наприклад, PDF, зображення тощо). Також є можливість просто повернути дані (JSON, XML) для відображення, використовуючи відповідний фреймворк клієнтської частини. Це зменшує завантаження процесора на сервері і кількість даних, що передаються.

Сервер не обмежений у надсиланні інформації з баз даних і може повертати результат виконання певних сценаріїв або дані із сервісів комунікації. Контент може бути навіть цільовим щодо пристрою клієнта, який його отримує. Через те, що інформація знаходиться в базі даних, її також можна легко передати та оновити через інші бізнес-системи (наприклад, відстеження).

Налаштований досвід взаємодії

Сервери можуть зберігати та використовувати інформацію про клієнтів, щоб постачати зручний та зроблений індивідуально користувацький досвід взаємодії. Наприклад, багато сайтів зберігають дані кредитних карток, щоб не потрібно було вводити їх повторно. Сайти, такі як Google Maps, можуть використовувати збережене та поточне місцезнаходження для надання інформації про маршрут, а також історію пошуку або подорожей для виділення місцевих підприємств у результатах пошуку.

Більш глибокий аналіз звичок користувача може бути використаний для прогнозування їхніх інтересів та подальших налаштувань відповідей та повідомлень, наприклад, надання списку раніше відвіданих популярних місць на карті.

Контрольований доступ до контенту

Програмування серверної частини дозволяє сайтам обмежувати доступ не авторизованим користувачам та надавати лише ту інформацію, яку користувачеві дозволено бачити.

Реальні приклади:

- Соціальні мережі, такі як Facebook, дозволяють користувачам повністю контролювати свої дані, але лише своїм друзям дозволяти переглядати чи коментувати. Користувач визначає, хто може переглядати його дані та чиї дані з'являються на його стіні. Авторизація - центральна частина досвіду взаємодії.
- Блог контролює доступ до контенту: статті видно всім, але тільки авторизовані користувачі можуть коментувати чи редагувати контент.
- На сайті онлайн-банкінгу доступ до контенту контролюється. Наприклад, без авторизації на ньому практично нічого не можна зробити, окрім перегляду загальної та довідкової інформації. Після авторизації стають доступними всі дозволені фінансові операції.

Зберігання інформації про сесію/стан

Програмування серверної частини дозволяє розробникам використовувати сесії – спочатку це механізм, що дозволяє серверу зберігати інформацію про поточного користувача сайту та надсилати різні відповіді на основі цієї інформації.

Це дозволяє, наприклад, сайту знати, що користувач був попередньо авторизований і виводити посилання на адресу електронної пошти або історію замовлень або, можливо,

зберегти прогрес простої гри, так щоб користувач міг повернутися на сайт продовжити з того місця, де він закінчив.

Повідомлення та засоби зв'язку

Сервери можуть надсилати спільні або власні повідомлення безпосередньо через сайт або електронною поштою, через смс, миттєві повідомлення, відеозв'язок або інші засоби зв'язку.

Ось кілька прикладів:

- Facebook або Twitter надсилає повідомлення електронною поштою та смс-повідомлення, щоб повідомити про нові розмови.
- Amazon регулярно надсилає листи на електронну пошту, що пропонують товари, схожі на ті, які вже були куплені або переглядалися вами, які можуть вас зацікавити.
- Веб-сервер може надсилати повідомлення адміністратору сайту, попереджаючи його про те, що на сервері закінчується пам'ять або про підозрілу активність користувача.
- Найпоширеніший вид повідомлень – це підтвердження реєстрації. Якщо створити новий обліковий запис на великому порталі використовуючи адресу електронної пошти, то на пошту надходить лист, який підтверджує факт реєстрації або містить інформацію про необхідність активувати обліковий запис.

Аналіз даних

Веб-додаток може збирати багато даних про своїх користувачів: що вони шукають, що вони купують, що вони рекомендують, як довго вони залишаються на кожній сторінці. Програмування серверної частини може бути використане, щоб вдосконалити відповіді, що базуються на аналізі цих даних.

Наприклад, Google рекламує товари на підставі попередніх пошуків. Facebook формує стрічку новин і рекламу, орієнтуючись на користувача чи його друзів. Rozetka буде пам’ятати список переглянутих, вподобаних і куплених товарів.

Висновки

Код серверної частини виконується на веб-сервері і його основна роль полягає в контролюванні інформації, що надсилається користувачеві. Тоді як код клієнтської частини в основному визначає структуру і спосіб подання інформації користувачу. Це корисно, оскільки дозволяє створювати веб-сайти, які ефективно доставляють інформацію, зібрану для конкретних користувачів і мати чітке уявлення про завдання, які вирішує розробник бекенда.

Код серверної частини може бути написаний різними мовами програмування, і для спрощення процесу написання коду варто використовувати веб-фреймворк.

Джерела інформації

https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Introduction