Лекція 2 : "Фреймворк у програмуванні "

 **Введення у поняття фреймворку**

Фреймворк ( або платформа розробки ) — це набір інструментів , бібліотек , правил та стандартів , призначених для спрощення та прискорення процесу створення програмних програм.

**Що таке фреймворк? Аналогія з готовими інгредієнтами для кулінарії**

Уявімо, що ви хочете приготувати вечерю для своєї сім'ї . Є два шляхи зробити це :

1. **З нуля:** Ви вирішуєте купити всі свіжі інгредієнти : м'ясо , овочі , спеції тощо .​ Вам потрібно буде все нарізати , змішати , приготувати. Це може зайняти багато часу , але у вас буде повний контроль над усім процесом та інгредієнтами .
2. **Використовуючи готовий набір:** Ви купуєте готовий набір для приготування страви. У ньому вже є попередньо нарізані овочі , приправлене м'ясо , а також інструкція з того, як це приготувати . Так, у вас менше контролю над деякими деталями, але ви готуєте вечерю швидше .

Тепер перенесемо цю аналогію на програмування :

**— Писати код із нуля** – це як готувати вечерю, купуючи всі інгредієнти окремо . Ви матимете​ повний контроль над кожною частиною вашого коду , але це може зайняти багато часу .

**— Використовуючи фреймворк** – це як використовувати готовий набір для приготування страви. Фреймворк пропонує готові « інгредієнти » ( компоненти , бібліотеки , інструменти ) та « інструкції » ( документація , рекомендації з кодингу ), що дозволяє вам швидше розробляти програми . Так, можливо , у вас буде менше контролю в деяких аспектах, але ви зможете зосередитися на основній логіці вашого додатки , не турбуючись про загальні рутинних задач.

Таким чином, фреймворк – це як готовий кулінарний набір для програмістів , що допомагає швидше та ефективніше «приготувати» програмне забезпечення .

Це , по суті , кістяк, який планує структуру програми та надає розробникам готові рішення для ряду типових завдань, таких як :

- Управління базою даних

- Робота з мережею

- Обробка користувальницьких запитів

- візуалізація інтерфейсу

 **Різниця між фреймворком та бібліотекою**

Насамперед чим поглибитися у роботу з фреймворками, важливо провести чітку грань між фреймворками та бібліотеками .

- Бібліотека надає функції або класи , які Ви можете використовувати у своєму коді. Контроль за програмним процесом залишається за розробником .

Приклад: бібліотека в JavaScript – це набір функцій для роботи з масивами , об'єктами та рядками, які розробник може використовувати за своїм розсуду .

- Фреймворк, на відміну бібліотеки , визначає архітектуру та контроль над основним програмним процесом . Розробник підлаштовується під фреймворк і дотримується його правил.

 **Основні принципи фреймворків**

Фреймворки побудовані на ряді ключових принципів , які допомагають розробникам писати код швидше та ефективніше :

1. Inversion of Control ( Інверсія управління ): Фреймворк сам керує потоком виконання програми , а розробник лише заповнює необхідні блоки. Приклад: ви створюєте контролери у фреймворку , а він сам викличе їх за необхідності .

2. Don't Repeat Yourself (DRY, Не повторюй себе : Фреймворки надають готові рішення для типових завдань, щоб ви не писали однаковий код багато разів. Приклад: робота з формами та їх валідацією у більшості фреймворків автоматизована .

3. Convention over Configuration ( Конвенції важливіше конфігурацій ): Фреймворки працюють на основі стандартних угод , і вам не потрібно витрачати час на налаштування програми , якщо ви слідуєте цим угодам . Приклад: фреймворк Rails у Ruby автоматично пов'язує таблиці бази даних із моделями за погодженими імен .

 Основні компоненти фреймворків

Фреймворки включають​​ безліч різних інструментів та компонентів . Залежно від типу​ фреймворка , його компоненти можуть відрізнятися , але основні елементи часто включають :

- Шаблони проектування ( design patterns ): певні архітектурні рішення для побудови коду ( наприклад , MVC - модель, уявлення , контролер ).

- Бібліотеки та API : готові функції для вирішення стандартних завдань ( робота з базами даних , мережевими запитами і т.д. ).

- Інструменти для тестування : вбудовані механізми для автоматизації тестування .

- Інструменти розгортання та складання : для швидкої складання та розгортання коду до робочої середу .

 Приклади популярних фреймворків

- Frontend -фреймворки\*:

- React ( JavaScript бібліотека для створення користувальницьких інтерфейсів , часто використовується як фреймворк.

- Angular ( JavaScript фреймворк від Google для створення динамічних веб-додатків.

- Vue.js ( JavaScript ): прогресивний фреймворк для розробки користувальницьких інтерфейсів .

**- Backend -фреймворки :**

- Django ( Python ): фреймворк, відомий своєю простотою і "батарейками" ( великим кількістю вбудованих інструментів ).

- Ruby on Rails ( Ruby ): потужний фреймворк для створення веб-додатків з філософією " конвенції " важливіше конфігурації ”.

- Spring ( Java ) : популярний фреймворк для створення програм на Java .

 Навіщо використовувати фреймворки?

Використання фреймворків дає розробникам ряд важливих переваг :

1. Прискорення розробки : завдяки наявності готових рішень для типових завдань, фреймворки дозволяють зосередитися на бізнес-логіці , а не на низькорівневій роботі .

2. Організація коду : фреймворки надають чітку структуру для додатків, що допомагає уникнути хаосу в коді і покращує його підтримку .

3. Стандартизація : використання популярних фреймворків допомагає підтримувати стандарти розробки , що полегшує взаємодія з іншими розробниками .

4. Підвищення безпеки : багато фреймворків мають вбудовані механізми захисту від поширених​ загроз ( наприклад , захист від XSS-атак, SQL- ін'єкцій тощо ) .

5. Тестування та підтримка : більшість сучасних фреймворків надають вбудовані інструменти для тестування та підтримки додатків, що спрощує процес розробки .

 Недоліки використання фреймворків

Незважаючи на переваги , фреймворки мають і свої недоліки :

1. Крива навчання : вивчення фреймворку може зайняти деяке час , особливо якщо фреймворк складний .

2. Обмежена гнучкість : іноді фреймворки нав'язують архітектурні рішення , які можуть підходити всім задач .​

3. Надмірність : фреймворки можуть включати в себе багато непотрібних для вашого проекту компонентів , що збільшує розмір додатки .

 Висновок

Фреймворки – це незамінний інструмент для сучасних розробників . Вони дозволяють зосередитися на рішенні бізнес- завдань, надаючи готові інструменти для роботи із типовими проблемами. Однак розробнику важливо вибрати правильний фреймворк залежно від завдань, що він вирішує . Важливо пам'ятати , що не завжди стоїть вибирати самий популярний фреймворк - іноді менше відоме рішення може краще підходити для конкретного проекту .

Фреймворки – це основа швидкого та якісного створення додатків, та їх використання допомагає розробникам досягати високих результатів у короткі терміни .

Давайте спробуємо познайомитися з фреймоворк React

React характеризують його творці , розробники компанії Facebook ось саме за аналогією з кулінарним рецептом . З'явившись у 2013 році, React швидко став набирати обороти і отримав найширше поширення . На момент створення курсу на GitHub у проекту понад 200 тисяч зірок .

Секрет успіху в тому, що React дозволив під іншим кутом подивитися на процес створення інтерфейсів . Він різко знизив поріг входу та складність одержуваних рішень . Причому не лише в порівнянні з ручною роботою з DOM, але й порівняно зі багатьма фреймворками.

І хоча React як бібліотеку для малювання можна вбудовувати в існуючий технологічний стек там, де це має сенс , він також здатний взяти на себе повне управління фронтендом . Щоправда, у цьому випадку для ефективної роботи доведеться підключити ще деякі ключові доповнення , такі як redux та react-router .

Фундаментальна ідея , що лежить в основі роботи React , виявилася настільки потужною , що її розширили далеко за межі браузера. З React можна працювати як на сервері ( server-side rendering ), так і на мобільних платформах ( React Native ). Ви не дочули : зараз мовою​ JavaScript можна створювати програми під мобільні платформи , які працюють майже так само ефективно , як і нативні додатки . Таку ситуацію , коли один підхід використовується для реалізації різних завдань (сайт, мобільні додатки ) називається **« Learn once , Write anywhere »** .

Налаштування библиотеки





















