УДК 004.056.55 **doi: 10.26906/SUNZ.2021.2.063**

А. І. Вінокуров, Г. І. Молчанов

Національний технічний університет «Харківський політехнічний інститут», Харків, Україна

ПЕРЕВАГИ ДИНАМІЧНИХ ВЕБ-СТОРІНОК НАД СТАТИЧНО-ГЕНЕРОВАНИМИ

Анотація. Предметом дослідження в статті є існуючі методи генерації відображення контенту веб-сторінок для користувачів веб-ресурсів. **Мета роботи** – аналіз переваг та недоліків основних підходів у створенні візуального веб-контенту для користувача, їхнє порівняння. **Висновки**: в сучасних системах веб-сайтів головним засобом спілкування з користувачем є інтерфейс, який має бути гнучким і зручним, а отже динамічно змінюватися відповідно до дій користувача. При цьому рівень обчислювальних потужностей сучасних пристроїв, що використовуються для доступу до веб-сайтів, є достатньо високим для швидкого та зручного відображення контенту, а динамічні сторінки адаптованими до різних цільових пристроїв. Таким чином забезпечується комфортний перегляд контенту та в той же час економне використання серверних ресурсів.

Ключові слова: HTML, CSS, JSP, Angular, Vue, Java, React, JavaScript, Redux, REST, C#, PHP, Python, Ruby.

Вступ

Вимоги інформаційної епохи зумовлюють необхідність переведення значної кількості інформації та людської діяльності у електронний вигляд. Не викликає сумнівів, що локальне сховище є небезпечним з точки зору можливого викрадення інформації, ненадійним з точки зору недовговічності роботи пристроїв збереження даних та й просто незручним через неможливість забезпечення вдалої класифікації, розподілення, точності пошуку. Стає зрозумілим чому в останній час стрімко збільшується кількість інформації, що зберігається на віддалених серверах, хмарних сховищах тощо.

Відповідно до потужностей, що використовуються для зберігання інформації, існують і програмні засоби, які відповідають за обробку даних. Є очевидним той факт, що можливість ефективної реалізації таких засобів має суттєве значення.

Інформація може бути загально-доступною, особистою, корпоративною чи навіть державною. Зазначаючи її важливість, доцільним стає забезпечення аутентифікації – виконання перевірки достовірності інформації та ідентифікаційних даних – на боці користувача. Подібний функціонал існує саме на динамічних веб-сторінках, проте статичні аналоги не мають достатнього рівня якості, зручності та швидкості.

Також вагомим питанням є зручність розробки користувацької частини тим чи іншим способом. Оскільки сучасні й прогресивні програми чи сайти розробляються в значній мірі цілими командами для кожного з напрямів, то необхідна можливість вдалої їхньої взаємодії, яка відбувається по різному для різних підходів, що обговорюються.

Таким чином, детальний аналіз сучасних підходів динамічного генерування сторінок є необхідним для розуміння переваг та недоліків, а також можливостей до вдалого використання цього інструменту. Під вдалим мається на увазі той варіант, коли забезпечується ефективне застосування та зручний процес розробки як у команді, так і у випадку індивідуального створення продукту.

Отже, метою роботи є аналіз переваг та недоліків основних підходів у створенні візуального веб-контенту для користувача, їхнє порівняння.

Виклад основного матеріалу

Оскільки мови програмування серверної частини, наприклад Java, C#, PHP, Python, Ruby, мають власні інструменти статичної генерації сторінок, то аналізувати всі випадки не доречно, а більш доцільним було б зупинитися на певній мові програмування та відповідному інструменті. В даному випадку одним із найбільш яскравих прикладів є мова програмування Java та її інструмент JSP.

Перш за все треба зазначити основні можливості JSP. Оскільки за допомогою Java можливо пряме створення коду сторінки із коду сервлету, що приймає запит, то виникає питання доцільності існування інструменту JSP.

JSP-код пишеться всередині коду веб-сторінки і дозволяє розташовувати Java-код, змішуючи його з кодом розмітки HTML. При цьому ніяких додаткових методів описувати не треба, код буде виконуватися напряму. На рис. 1 наведений приклад коду для виведення повідомлення за допомогою JSP.

```
<html>
  <head>
     <title>Greetings!</title>
  </head>
  <body>
      <h1>Hello world!</h1>
      <i><%= out.println("JSP works") %></i>
  </body>
</html>
```

Рис. 1. Приклад JSP-коду

З огляду на присутність Java-коду в коді вебсторінки, з'являється необхідність наявності відповідних знань і умінь користуватися цією мовою програмування. А це саме по собі є додатковою і чималою вимогою до веб-розробника.

Проте існує бібліотека тегів JSTL, що дозволяє програмувати за допомогою заздалегідь визначених інструкцій без використання Java-коду безпосередньо на веб-сторінці та потребує мінімум додаткових знань.

Приклад реалізації простого коду вказаним вище методом наведений на рис. 2.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
 prefix="c" %>
<html>
  <head>
      <title>Greetings!</title>
  </head>
  <body>
        <c:out value="Hello world! - From JSTL">
        </c:out>
  </body>
</html>
```

Рис. 2. Приклад JSTL-коду

Зазначена бібліотека включає до себе основні можливості JSP, при цьому має велику кількість переваг, серед них: підвищена читабельність, відсутність Java-коду, об'ємна стандартна бібліотека тегів, в ній істотно підвищено безпека коду, завдяки заздалегідь сформованим обмеженням та перевіркам. Також існує можливість легкого створення користувацьких тегів на відміну від функцій у JSP. Створення власних JSTL-інструкцій дійсно є простим, оскільки дозволяє використати як Java-код, так і інші JSTL-теги.

Проте жодний із вказаних засобів чи можливостей JSP не вирішує основні проблеми серверного підходу до генерування веб-сторінок. Серед основних проблем наявні:

- витрати серверних ресурсів, часу та пам'яті;
- незручність при відлагодженні;
- неможливість повноцінного тестування;
- порушення принципу розділення відповідальності;
- вимога додаткових знань у розробника.

Раніше подібні недоліки вважалися такими, яких не уникнути і всі існуючі аналоги працювали за подібною схемою, мали схожі проблеми, як і метод на основі JSP. Таким чином, як тільки з'явився альтернативний шлях, або можливість вирішити ці проблеми, почався активний перехід на нові технології. На рис. 3 можна побачити графіки популярності запитів у пошуковій системі Google. Синім кольором зображено те, що відповідає JSP, червоним, жовтим та зеленим позначено React, Angular, Vue відповідно.

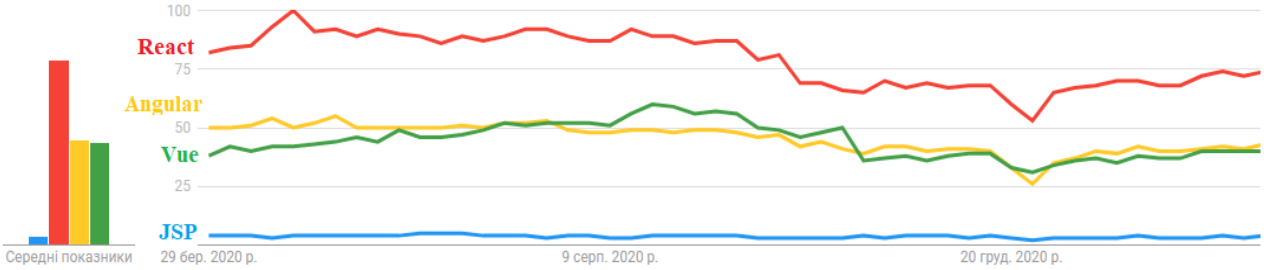

Рис. 3. Статистика пошуку веб-технологій

Можна помітити значний програш у популярності JSP протягом усього 2020-го року порівняно з усіма іншими зазначеними технологіями. Навіть якщо провести аналіз прогресії, то очевидним стає те, що дана технологія не набирає популярності, а навпаки спостерігається певний спад, хоча й дуже повільний, однак точно не зріст.

Оскільки бізнес диктує свої умови, які вимагають руху лише до більш доцільного, сучасного, перспективного, то очевидною стає тенденція у популяризації динамічних сторінок на базі JavaScript-фреймворків.

На рис. 4 зображена популярність запитів на фреймворки у порівнянні одне до одного, де вчергове можна побачити, що JSP не є передовою технологією в жодній з країн світу.

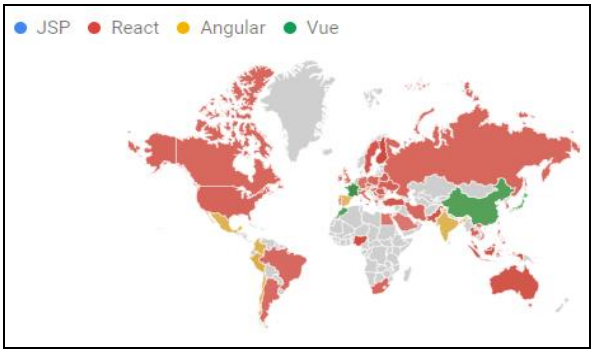

Рис. 4. Статистика пошуку веб-технологій

Внаслідок застарілості технології серверної генерації сторінок, очевидним є зростання популярності альтернативних рішень. Однак не розкритим залишається питання щодо можливостей зазначених JavaScript фреймворків та їхніх значних переваг, а також постійного й швидкого розвитку в різних напрямках.

Розглянемо на прикладі React як статистично-найпопулярнішого фреймворку. У React.js програма складається з компонентів, що працюють за принципами ООП, а саме підтримують наступне: наслідування, композицію, агрегацію, інкапсуляцію. Оскільки логіка компонентів написана на JavaScript, замість шаблонів можна з легкістю передавати складні дані у програму і зберігати стан окремо від DOM. Компоненти реалізують метод render(), який приймає вхідні дані і повертає те, що буде показано користувачу. У цьому прикладі використовується XML-подібний синтаксис під назвою JSX. Доступ до вхідних даних, які передаються в компонент, можна отримати за допомогою render() та this.props.

JSX (рис. 5) є основним способом написання React-коду, хоча й не виключним. Підтримується й «чистий» JavaScript-код, що не потребує використання охоплюючих HTML-тегів. При бажанні звичайний JSX-код можна трансформувати до подібного за допомогою інструменту Babel. Таким чином, будь-який із підходів є прийнятним та однаково ефективним для роботи з даним фреймворком.

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Привіт, {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Петро" />,
  document.getElementById('hello-example'),
);
```

Рис. 5. Приклад JSX-коду

На рис. 5 надано приклад абсолютно ідентичної програми до зазначеної раніше, єдина різниця, що тут код написаний на чистому JavaScript, без використання JSX. Це не впливає на швидкість виконання чи споживання пам'яті, лише зручно розробнику.

```
class HelloMessage extends React.Component {
  render() {
    return React.createElement(
      "div",
      null,
      "Привіт, ",
      this.props.name
    );
  }
}

ReactDOM.render(React.createElement(HelloMessage,
{ name: "Петро" }),
 document.getElementById('hello-example'));
```

Рис. 6. Приклад React-коду без JSX

React притримує парадигми декларативного програмування. Таким чином, від розробника лише вимагають описання різних частин інтерфейсу в різних станах системи, а сам фреймворк буде самостійно рендерити необхідні елементи, оновлюючи сторінку у відповідності до динамічних змін, ініційованих користувачем чи іншими чинниками.

Використовуючи так звані «пропси» і «стан», ми можемо створити невелику програму для складання списку справ. Цей приклад використовує стан для відстеження поточного списку елементів, а також тексту, введеного користувачем. Хоча обробники подій здаються вбудованими, вони будуть зібрані та реалізовані за допомогою методу делегування подій. При цьому жодних обмежень про інші технології, які будуть використовуватись додатково, не існує. Тому є можливим розробляти нові функції в React, не переписуючи існуючий код. React також може рендеритись на сервері, використовуючи Node, і приводити в дію мобільні програми, які використовують React Native.

Тобто React може бути абсолютно спокійно використаним для генерації веб-сторінки на боці серверу, при цьому не втрачаючи своїх можливостей чи властивостей. Для генерації на боці серверу доречно використовувати технологію Node.js, яка також є доволі популярною технологією для серверів, хоча це й є певним обмеженням при бажанні серверної генерації сторінок. React дозволяє взаємодіяти з іншими бібліотеками та фреймворками. У прикладі на рис. 7 використана зовнішня бібліотека — Markdown, для зміни значення, збільшення естетичної цінності текстових елементів.

```
const str = '# Heck Yes\n\nThis is great!'

<Markdown options={{ wrapper: 'article' }}>
  {str}
</Markdown>;

// or

compiler(str, { wrapper: 'article' });

// renders

<article>
  <h1>Heck Yes</h1>
  <p>This is great!</p>
</article>
```

Рис. 7. Приклад JSX-коду з Markdown

Формат Markdown загалом має багато функцій для зручного зображення тексту: списки відмічені символами, пронумеровані послідовності, відмічення заголовного тексту особливим шрифтом, створення горизонтальних ліній різної конфігурації (суцільні, штрихові, пунктирні, штрих-пунктирні, створені зі спеціальних символів, подвійної товщини, дві паралельні лінії), користувацький розмір інтервалів між рядками. Також існують варіанти дублювання можливостей HTML кодом Markdown. При цьому результат іноді буде значно більш компактним, зручним, зрозумілим і універсальним. Приклад наведено на рис. 8.

```
* Пункт в маркованому (ненумерованому) списку
    * Підпункт, відділений 4 пробілами
* Інший пункт в маркованому списку

1. Пункт в нумерованому списку
    1.1. Підпункт, відділений 4 пробілами
2. Інший пункт в нумерованому списку
```

Рис. 8. Приклад Markdown інтерфейсу

Однією з основних переваг динамічних сторінок є власне їхня здатність до динамічного змінювання без перевантаження чи навіть запитів на сервер. Окрім як реакції на дії користувача також допустимими є зміни з часом (код наведено на рис. 9). Також на рис. 10 можна побачити результат, який показує приклад зміни тексту відповідно до поточного часу і зміна, яка відбувається кожну секунду.

Інший варіант використання фреймворку React – це його варіація під назвою React Native. Він дозволяє використання подібних механізмів розробки для формування інтерфейсів мобільних програм, поєднуючи сучасні засоби формування веб-сторінок та мобільну розробку, при цьому закріплюючи окремі ролі різних технологій у випадку створення мобільних застосунків, що мають за задачу виконання функцій програмного забезпечення. Однак для різних ігрових додатків подібний фреймворк не є доречним. Останнім, проте чи не одним із найважливіших моментів є цілковита підтримка архітектури REST, завдяки технології модуля Redux, що є відкритою JavaScript бібліотекою для керування станами програм, які працюють на основі цієї мови.

```
function tick() {
  const element = (
    <div>
      <h1>Hello, world!</h1>
      <h2>Зараз {new Date().toLocaleTimeString()}.</h2>
    </div>
  );
  ReactDOM.render(element, document.getElementById('root'));
}

setInterval(tick, 1000);
```

Рис. 9. Динамічний React код

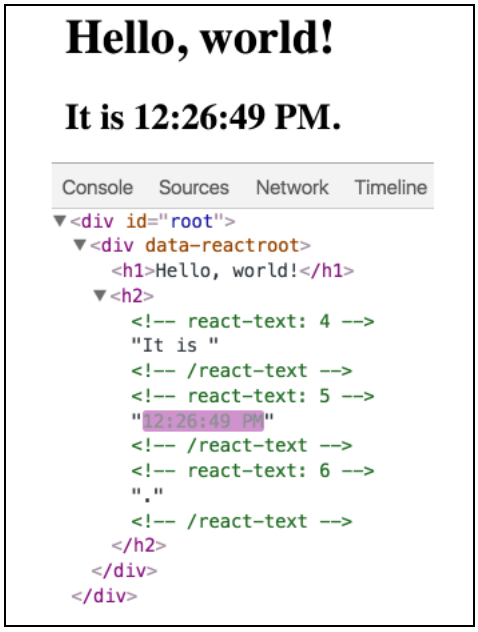

Рис. 10. Динамічна сторінка на основі React

Його часто використовують разом із фреймворками React чи Angular. Перш за все Redux – це контейнер для застосунків на основі JavaScript. Він допомагає оптимізувати код програм, сприяє їхньому налагодженню, має невеликий розмір та ідеально інтегрується із React.

Висновок

Таким чином, у відповідності до зазначених аргументів можна зробити деякі висновки. Одним із головних є те, що технологія статичної генерації сторінок, наприклад JSP, є застарілою на даний час, хоча вона і може стати в нагоді для простих сайтів із мінімальними вимогами до динамічності чи швидкості роботи.

Такі сайти можуть дозволити собі повне перевантаження сторінки щораз, як відбувається зміна в її стані або дія користувача.

Тим не менш багато сучасних сайтів, веб-застосунків, інтернет-ресурсів потребують більшого від частини із інтерфейсом, тому тенденції значно змінюються, особливо враховуючи популярність REST-архітектури та модулю Redux, що дозволяє досконале "спілкування" між частиною користувацького інтерфейсу та серверною частиною застосунку.

Було розглянуто приклад сучасних засобів керування веб-сторінками у вигляді JavaScript фреймворків, а саме Vue, Angular, React. React був детальніше проаналізований і згідно більшості пунктів порівняння стає очевидною перевага більш сучасної технології.

Також стало очевидним, що сучасні JavaScript модулі та елементи досконало доповнюють один одного, через це будь-які нові технології можуть бути легко впроваджені до вже наявних модулів, що забезпечує ефективне оновлення програм.

Отже, обчислювальні потужності персональних комп'ютерів та серверів зростають, нові технології стають можливими до користування та розвитку. Тому вони стають популярнішими, що неодноразово підтверджується статистикою з усього світу.

Саме через це стає очевидною тенденція у цій сфері розробки програмного забезпечення і динамічні сторінки займають у ній найперше місце.

СПИСОК ЛІТЕРАТУРИ

1. Difference between JavaScript and JSP https://www.geeksforgeeks.org/difference-between-javascript-and-jsp/
2. Difference Between JSP and JavaScript https://www.educba.com/jsp-vs-javascript/
3. Angular vs jsp vs react vs vue https://www.npmtrends.com/angular-vs-jsp-vs-react-vs-vue
4. Сравнение JSP и VueJS https://dernasherbrezon.com/posts/compare-jsp-vuejs/
5. Статистика пошуку за запитами про різні інструменти https://trends.google.com/trends/explore?q=%2Fm%2F0bsn3,React,Angular,Vue
6. Документація React.js https://reactjs.org/
7. Документація про Markdown https://daringfireball.net/projects/markdown/syntax
8. Документація Redux https://redux.js.org/

Received (Надійшла) 18.03.2021
Accepted for publication (Прийнята до друку) 21.04.2021

Advantages of dynamic web-pages over statically-generated ones

Artemii Vinokurov, Heorhii Molchanov

Abstract. The subject of research in the article is the methods of generating the presentation of content for users of web resources. **The purpose** of the work is to compare and analyze the pros and cons of the main approaches to creating visual web content for the users. **Conclusions:** modern web-site systems include an interface as the primary way to communicate with the user, so it should be flexible and convenient. Therefore, the interface is likely to change dynamically corresponding to user actions. At the same time, the computing power of modern devices used to access websites is high enough for fast and convenient display of content. Dynamic pages are also adapted to different end-user devices.

Keywords: HTML, CSS, JSP, Angular, Vue, Java, React, JavaScript, Redux, REST, C#, PHP, Python, Ruby.