DEVELOPMENT A TAG TEMPLATE ENGINE EXTENTION WITH PHP

Egnar Ozdikililer

Istanbul Technical University, Faculty of Aeronautics and Astronautics, Istanbul, Turkey
ozdikililer@itu.edu.tr

Abstract

Within the scope of this study, a PHP-based application has been developed that makes it easy to dynamically create comprehensive static pages with PHP and reduces the coding time and error rate. At the last level of the application the app creates an html page as a result by adding the information taken from the database. It is aimed to develop further in the short term and to add a new one to the open source server rendered software.

Keywords: php, template engine, php to html, engine development

INTRODUCTION

The concept of Template Engine arises when we want to use static pages dynamically and do it without much effort.

An effort to show a changing content in a fixed number has emerged, which can perform this process and also shorten the coding time.

RELATED WORKS

The most popular Template Engine concept JSX (JavaScript eXtension) structure today is an extension of Javascript. JSX is not a Framework.

It is a View, UI Library that configures the front face/page of the application. It enables the use of html syntax structure in Javascript. By writing small html codes piece by piece in Javascript, using them as components, and then putting them together with a certain hierarchy that we have created.

JSX provides a structure to modify and develop its own DOMs in html codes, it is actually a React extension and JSX is updated to work fully with any browser. Updating the DOM using JSX brings website performance improvement and development efficiency [1].

DOM (Document Object Model) is the structure that provides the visibility of web pages and the coexistence of HTML elements. The DOM is updated when the user enters any command on the web page. While this update runs smoothly on static pages, it often negatively affects the application on dynamic pages, weakening/weakening performance in terms of performance.

The VirtualDOM structure selected as a precaution comes into play.

VirtualDOM is a feature that comes with React. Here, if/when there is a change in the application, "only" the modified field is updated and passed to the DOM. VirtualDOM is actually the DOM in memory in the value/key pattern and is a copy of the DOM.

The general operation and rules can be summarized as follows;

• Each component can use a single parent element in Return.

• Custom keywords are found, eg className is used to define a class attribute in a component.

• The representation of a constant value or Javascript expressions in JSX format is done using { curly braces }. To give an example; When displayed without curly braces, the JSX format treats the value as text.

React app has folders similar to each other. These are the src(source) and public(source) folders.

When we open a page in a React web application through the browser, we see what is inside the index.html page. When we examine Index.html, we only see the title of the application and the information in the body, but we cannot see the element that fills the application's content.

It should be noted that the HTML code that the bill finally outputs to us. Actually, the purpose of React is to generate HTML code. React's declarative code outline eliminates many frontend problems.

The motto of the React Js team is known to be "learn once, write anywhere".

TAG ENGINE EXTENTION WITH PHP

Within the scope of this study, it is aimed to create its own server tag library by applying the above mentioned JSX browse rendering mentality to PHP. It is aimed that the user can create the templates they want with the tags they want.

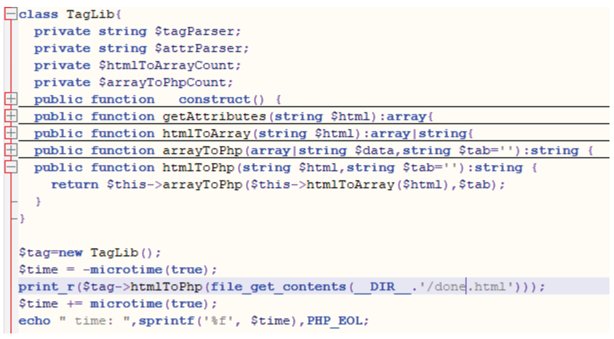
The extension written in this study, which was started with this purpose, has the quality of an application, and it was aimed and written to create the same structure with PHP [2,4].

It automatically generates the relevant structure code described below with php and prints a dynamic result as html.

The code that allows us to create an interface for users is written. It is intended to be developed as an open source library in the future.

The working code done is a php extension. It provides the use and formatation of html syntax structure in PHP. It offers a structure formation that it can replace and develop with its own embedded code - convertible structure even in simple html structure and complex codes. The real working flow chart of the application is given in Figure 1. As can be seen here, it reads the Index.html file it encounters in the server-site-based application and parses it into the relevant sections (htmlToPhp).

Portion of parse Tag code is given below.



```
class TagLib{
  private string $tagParser;
  private string $attrParser;
  private $htmlToArrayCount;
  private $arrayToPhpCount;
  public function   construct() {
  public function getAttributes(string $html):array{
  public function htmlToArray(string $html):array|string{
  public function arrayToPhp(array|string $data,string $tab=''):string {
  public function htmlToPhp(string $html,string $tab=''):string {
    return $this->arrayToPhp($this->htmlToArray($html),$tab);
  }
}

$tag=new TagLib();
$time = -microtime(true);
print_r($tag->htmlToPhp(file_get_contents(__DIR__.'/done.html')));
$time += microtime(true);
echo " time: ",sprintf('%f', $time),PHP_EOL;
```

The parsed html document requirements are determined and converted into the structure in which the new read will be created. It separates the attributes of the tags and puts them in the array. Convert it, and after this it parses all the tags in the HTML content and puts it into a hierarchical array.

In the prepared code, structures in the form of class TagX {…} are created.

By creating element-based and arranging the complex structure, values from the relevant tagX are created and added to the relevant sections as needed. In this way, code readability also becomes easier.

In tag block formation, a new php file is created (Table.php) with the data in the database. As a result, it creates the result.htm file and itself is ready for use on the page.

It allows us to create static websites that can be updated in real time.

While the system is static, it turns into a dynamic structure that is stripped of its stability.

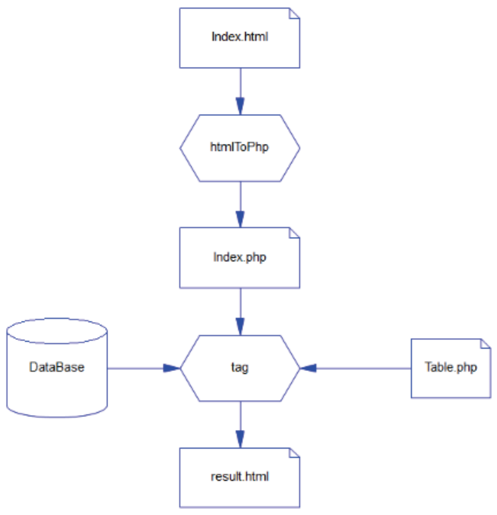


Fig. 1. *Tag engine extention create process*

It offers a new perspective in the PHP world that further reduces coding, allowing us to easily add and update pages with a few tabs [3,5,6].

The PHP format, in which we write the html codes in PHP,

```
Index.html
<Table employees={$employees}/>
```

Fig. 2. Code block Index.html.

creates the tagX structure and the smallest unit elements, which we will examine in detail, in detail.

,

```
index.php
tag('Table',[employees='{$employees}']);
```

Fig. 3. Code block index.php.

The PHP structure starts when the Table structure is encountered in the Index.html parse operation.

Here, the field created by index.php, which corresponds to the Table field and will be managed from TagX, is parsed and Table.php is formed.

```
Table.php
<table>
  <tbody>
  <tr>
    <th>Name</th>
    <th>Address</th>
  </tr>
  <?php foreach($employees as $employee){ %}
    <tr>
      <td><?=$employee.name?></td>
      <td><?=$employee.address?></td>
    </tr>
  <?php } ?>
  </tbody>
</table>
```

Fig. 3. Code block Table.php.

Here, the final result.html is formed by adding the employee.json

```
database
employees.json
[
  {
    'name':'Egnar Ozdikililer',
    'address':'Istanbul Levent'
  },
  {
    'name':'Elis Ozdikililer',
    'address':'Istanbul Levent'
  }
]
```

Fig. 4. Code block employees.json.

data received from the database with the service (Fig. 5).

```
result.html
<table>
  <tbody>
  <tr>
    <th>Name</th>
    <th>Address</th>
  </tr>
  <tr>
    <td>Egnar Ozdikililer</td>
    <td>Istanbul Levent</td>
  </tr>
  <tr>
    <td>Elis Ozdikililer</td>
    <td>Istanbul Levent</td>
  </tr>
  </tbody>
</table>
```

Fig. 5. Code block result.php.

As a result, the pages written are automatically coded and used as html without any errors and time-wasting.

Since the application server – site is running, it does not expect the data reaching the database to be local. Here, it is designed to read the data received in the web service or microservice structure and use it in an integrated manner in the relevant structure. In the given example, it is reading a JSON (JavaScript Object Notation) file. JSON is an open standard file and data interchange format. Tests have shown suitability for other data structures.

CONCLUSION

In the scope of this study, a new one has been added to PHP-based, server rendered, open source software. An application has been written, which instantly transforms static pages into dynamic pages, facilitating the operation, reducing the coding time and error rate, adding the data taken from the database and creating an html page as a result.

REFERENCE

[1] https://reactjs.org/docs/introducing-jsx.html last access 01.11.2022

[2] https://php.org/ last access 21.10.2022
[3] Nixon R., Learning PHP, MySQL & JavaScript, 5th Edition, O'Reily 2004
[4] Ullman L., PHP for the Web: Visual Quick Start Guide, 5th Edition, Prime 2004
[5] Ullman L., PHP and MySQL for Dynamic Web Sites, 4th Edition, Peachpit Press 2014
[6] Schlossnagle G., Advanced PHP Programming 3th Edition, Sams Publishing, 2004.