EBook Gratis

APRENDIZAJE twig

Free unaffiliated eBook created from
Stack Overflow contributors.

#twig

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: twig

It is an unofficial and free twig ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official twig.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con la ramita

Observaciones

Esta sección proporciona una descripción general de qué es la ramita y por qué un desarrollador puede querer usarla.

También debe mencionar cualquier tema grande dentro de la rama, y vincular a los temas relacionados. Dado que la Documentación para la ramita es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Examples

Uso básico de API

También se puede instalar descargando el código fuente y colocándolo en un directorio de su proyecto. Sin embargo, hay muchos beneficios al usar compositor.

```
require '/path/to/lib/Twig/Autoloader.php';
Twig_Autoloader::register();

$loader = new Twig_Loader_Filesystem('/path/to/templates');

$options = array(
    'strict_variables' => false,
    'debug' => false,
    'cache'=> false
);

$twig = new Twig_Environment($loader, $options);
```

Al crear una nueva instancia de `Twig_Environment` , puede pasar una matriz de opciones como segundo argumento del constructor. Aquí hay una lista de las opciones disponibles:

- depuración (*booleano* , predeterminado `false`)

  Cuando se establece en verdadero, las plantillas generadas tienen un método `__toString()` que puede usar para mostrar los nodos generados.

- conjunto de caracteres (*cadena* , por defecto `utf-8`)

  El conjunto de caracteres utilizado por las plantillas.

- base_template_class (*cadena* , `Twig_Template` defecto)

  La clase de plantilla base a usar para las plantillas generadas.

- caché (*cadena o* `false` , `false` defecto)

Una ruta absoluta donde almacenar las plantillas compiladas, o falsa para deshabilitar el almacenamiento en caché (que es el valor predeterminado).

- auto_reload (*booleano* , predeterminado heredado de la *depuración*)

  Cuando se desarrolla con Twig, es útil volver a compilar la plantilla siempre que cambie el código fuente. Si no proporciona un valor para la opción auto_reload, se determinará automáticamente en función del valor de depuración.

- strict_variables (*booleano* , predeterminado `false`)

  Si se establece en falso, Twig ignorará silenciosamente las variables no válidas (variables y / o atributos / métodos que no existen) y las reemplazará con un valor nulo. Cuando se establece en verdadero, Twig lanza una excepción en su lugar.

- autoescape (*string o booleano* , por defecto `true`)

  Si se establece en verdadero, el escape automático de HTML se habilitará de forma predeterminada para todas las plantillas.

  A partir de Twig 1.8, puede configurar la estrategia de escape para usar (html, js, false para deshabilitar).

  A partir de Twig 1.9, puede configurar la estrategia de escape para usar (css, url, html_attr o una devolución de llamada de PHP que toma la plantilla "nombre de archivo" y debe devolver la estrategia de escape para usarla - la devolución de llamada no puede ser un nombre de función para evitar colisión con estrategias de escape incorporadas).

  A partir de Twig 1.17, la estrategia de escape de nombre de archivo determina la estrategia de escape que se debe utilizar para una plantilla basada en la extensión del nombre de archivo de la plantilla (esta estrategia no implica ningún gasto adicional en tiempo de ejecución, ya que el escape automático se realiza en el momento de la compilación).

- optimizaciones (*entero* , por defecto `-1`)

  Una bandera que indica qué optimizaciones aplicar:

  ```
  set to -1 to enabled all optimalizations
  set o 0 to disable all optimalitazations
  ```

Guía oficial de instalación de Twig

Una extensión Twig PHP (escrita en C) también se puede compilar e instalar, y el paquete PHP automáticamente aprovechará eso para optimizar algunas rutinas comunes.

¿Qué es la ramita?

Twig es un lenguaje de plantillas que compila a código PHP optimizado. Se utiliza principalmente para generar HTML, pero también puede utilizarse para generar cualquier otro formato basado en

texto. Es un componente independiente que se puede integrar fácilmente en cualquier proyecto de PHP.

Proporciona muchas características excelentes:

- Autoescaping para HTML (ayuda a prevenir XSS)
- Sintaxis diseñada teniendo en cuenta las plantillas (basadas en plantillas de Django)
- Herencia de plantillas
- Macros

Documentación oficial de plantillas Twig

Ejemplo de sintaxis de Twig:

```
{% extends "base.html" %}

{% block sidebar %}
  {{ parent() }}
  <span>Sidebar content specific to this page</span>
{% endblock sidebar %}

{% block body %}
  <p>Select an item:</p>
  <ul>
    {% for item in list %}
      <li><a href="/items/{{ item.id }}">{{ item.name }}</a>
    {% else %}
      <li>No items yet.
    {% endfor %}
  </ul>
{% endblock body %}
```

Introducción

Si tiene alguna exposición a otros idiomas de plantillas basadas en texto, como Smarty , Django o Jinja , debe sentirse como en casa con Twig . Es ***amigable*** tanto para los ***diseñadores como para los desarrolladores*** al seguir los principios de PHP y agregar funcionalidades útiles para los entornos de plantillas.

Las características clave son ...

- Rápido: Twig compila las plantillas hasta el código PHP optimizado. La sobrecarga en comparación con el código PHP normal se redujo al mínimo.
- Seguro: Twig tiene un **modo de caja de arena** para evaluar el código de la plantilla no confiable. Esto permite que Twig se utilice como lenguaje de plantilla para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla.
- Flexible: Twig es impulsado por un **lexer** y un **analizador** flexibles. Esto permite al desarrollador definir sus propias **etiquetas** y **filtros personalizados** , y crear su propio DSL.

Twig es utilizado por muchos proyectos de código abierto como Symfony , Drupal , eZPublish y

muchos frameworks tienen soporte también para ellos, como Slim , Yii , Laravel , Codeigniter , silex y Kohana , solo por nombrar algunos.

Instalación

La forma recomendada de instalar Twig es a través de Composer :

Para usuarios de **php 5.x**

```
composer require "twig/twig:~1.0"
```

Para usuarios de **php 7.x**

```
composer require "twig/twig:~2.0"
```

Lea Empezando con la ramita en línea: https://riptutorial.com/es/twig/topic/1100/empezando-con-la-ramita

Capítulo 2: Herencia de plantillas

Examples

Usando una plantilla base

base.twig.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>{{ title | default('Hello World') }}</title>
        <link rel="stylesheet" type="text/css" href="theme.css">
        {% block css %}
        {% endblock %}
    </head>
    <body>
    {% block body %}
        <nav>
        {% block navigation %}
            <a href="#">Link</a>
            <a href="#">Link</a>
            <a href="#">Link</a>
        {% endblock navigation %}
        </nav>
        <section id="container">
            <section id="content">
            {% block content %}
            <p>
            Lorem ipsum dolor sit amet.
            </p>
        </section>
        {% endblock content %}
    </section>
    {% endblock body %}
    </body>
</html>
```

page.twig.html

```
{% extends base.twig.html %}
{% block navigation %}
<a href="page2.html">Page 2</a>
<a href="page3.html">Page 3</a>
<a href="page4.html">Page 4</a>
{% endblock %}
{% block content %}
This is my first page
{% endblock content %}
```

Cambiar el contenido de una plantilla incluida.

article.twig.html

```
<article>
    <h1>{{ article.title }}</h1>
    {% block content %}
    <p>{{ article.content }}</p>
    {% endblock %}
</article>
```

articles.twig.html

```
{# use default template for article #}
{% for article in articles %}
    {% include "article.twig.html" %}
{% endfor %}
```

```
{# use custom template for article #}
{% for article in articles %}
    {% embed "article.twig.html" %}
        {% block content %}
            <img src="{{ article.image }}" alt="{{ article.title }}" />
            {{ parent() }}
        {% endblock %}
    {% endembed %}
{% endfor %}
```

Herencia variable

parent.html.twig

```
<!DOCTYPE html>
<html>
    <head>
        <title>{{ title_variable | default('Normal Page') }}</title>
    </head>
    <body>
         <h1>This is the {{ title_variable }} Page</h1>
    </body>
</html>
```

---

child.html.twig

```
{% extends "parent.html.twig" %}
{% set title_variable = "Child" %}
```

Comparación de inclusión, extensión, uso, macro, incrustación

Hay varios tipos de herencia y reutilización de código en Twig:

Incluir

El objetivo principal es la **reutilización de código** . Considere utilizar `header.html.twig` & `footer.html.twig` dentro de `base.html.twig` como ejemplo.

header.html.twig

```
<nav>
   <div>Homepage</div>
   <div>About</div>
</nav>
```

---

base.html.twig

```
{% include 'header.html.twig' %}
<main>{% block main %}{% endblock %}</main>
```

Se extiende

El objetivo principal es **la herencia vertical** . Considere ampliar `base.html.twig` dentro de `homepage.html.twig` y `about.html.twig` como ejemplo.

base.html.twig

```
{% include 'header.html.twig' %}
<main>{% block main %}{% endblock %}</main>
```

---

homepage.html.twig

```
{% extends 'base.html.twig' %}

{% block main %}
<p>You are at the homepage</p>
{% endblock %}
```

---

about.html.twig

```
{% extends 'base.html.twig' %}

{% block main %}
<p>You are at the about page</p>
{% endblock %}
```

Utilizar

El objetivo principal es la **reutilización horizontal** . Considere usar `sidebar.product.html.twig` dentro de las `sidebar.product.html.twig` `single.product.html.twig` (extiende `product.layout.html.twig`) y `single.service.html.twig` (extiende 'service.layout.html.page'). (Es como macros, pero para bloques)

barra lateral.html.twig

```
<aside>{% block sidebar %}{% endblock %}</aside>
```

---

single.product.html.twig

```
{% extends 'product.layout.html.twig' %}

{% use 'sidebar.html.twig' %}
{% block main %}
<p>You are at the product page for product number 123</p>
{% endblock %}
```

---

single.service.html.twig

```
{% extends 'service.layout.html.twig' %}

{% use 'sidebar.html.twig' %}
{% block main %}
<p>You are at the service page for service number 456</p>
{% endblock %}
```

Macro

El objetivo principal es **tener marcas reutilizables en muchas plantillas con variables** . Considere una función que obtiene algunas variables y genera un marcado.

form.html.twig

{% macro input (nombre, valor, tipo)%} <input type = "{{type | default ('text')}}" name = "{{name}}" value = "{{value | e}} "}}" /> {% endmacro%}

---

profile.service.html.twig

```
{% import "forms.html.twig" as forms %}

<div>{{ forms.input('username') }}</div>
```

Empotrar

El objetivo principal es la **anulación de bloque** . Tiene funcionalidad de `Use` e `Include` juntos. Considere incluir `pagination.html.twig` en `product.table.html.twig` & `service.table.html.twig` .

pagination.html.twig

```
<div>
    <div>{% block first %}{% endblock %}</div>
    {% for i in (min + 1)..(max - 1) %}
        <div>{{ i }}</div>
    {% endfor %}
    <div>{% block last %}{% endblock %}</div>
</div>
```

---

product.table.html.twig

```
{% set min, max = 1, products.itemPerPage %}

{% embed 'pagination.html.twig' %}
    {% block first %}First Product Page{% endblock %}
    {% block last %}Last Product Page{% endblock %}
{% endembed %}
```

---

service.table.html.twig

```
{% set min, max = 1, services.itemPerPage %}

{% embed 'pagination.html.twig' %}
    {% block first %}First Service Page{% endblock %}
    {% block last %}Last Service Page{% endblock %}
{% endembed %}
```

Tenga en cuenta que el archivo incorporado (`pagination.html.twig` aquí) tiene acceso al contexto actual (variables `min` , `max` aquí). También puedes pasar variables extra al archivo incrustado:

pagination.html.twig

```
<p>{{ count }} items</p>
<div>
    <div>{% block first %}{% endblock %}</div>
    {% for i in (min + 1)..(max - 1) %}
        <div>{{ i }}</div>
    {% endfor %}
    <div>{% block last %}{% endblock %}</div>
</div>
```

---

product.table.html.twig

```
{% set min, max = 1, products|length %}

{% embed 'pagination.html.twig' with {'count': products|length } %}
    {% block first %}First Product Page{% endblock %}
    {% block last %}Last Product Page{% endblock %}
{% endembed %}
```

Lea Herencia de plantillas en línea: https://riptutorial.com/es/twig/topic/2922/herencia-de-plantillas

Capítulo 3: Ramita extensible

Observaciones

Twig ya tiene algunos filtros y funciones incorporados , pero ¿qué sucede si faltan las funciones integradas o si tiene que acceder a algunas funciones predeterminadas de `PHP` en una plantilla?

Examples

Agregar filtros / funciones personalizados

Aquí hay algunos ejemplos de cómo agregar nuevos filtros / funciones a la `twig` ,
el synax para agregar `Twig_Function` s es el mismo que `Twig_Filter` , solo cambia las palabras clave en consecuencia

```
<?php
   $twig = new Twig_Environment($loader);

   /* You can chain a global function */
   $twig->addFilter(new Twig_SimpleFilter('floor', 'floor'));

   /* You can specify a custom function */
   $twig->addFilter(new Twig_SimpleFilter('money', function($value, $currency, $prefix =
false, $decimals = 2, $dec_point = "." , $thousands_sep = ",") {
       $value = number_format($value, $decimals, $dec_point, $thousands_sep);
       if ($prefix) return $currency.' '.$value;
       return $value.' '.$prefix;
   });

   /* You can chain an object's method */
   $twig->addFilter(new Twig_SimpleFilter('foo_bar', array($foo, 'bar')));
```

Creando una extensión de Twig

Puede agrupar todas sus funciones / filtros / pruebas personalizadas ... dentro de una clase `Twig_Extension` personalizada:

ProyectoTwigExtension

```
class ProjectTwigExtension extends Twig_Extension {

    public function getFunctions() {
        return array(
            new Twig_SimpleFunction('twig_function_name', array($this,
'getTwigFunctionName')),
            new Twig_SimpleFunction('twig_function_foo', array($this, 'getTwigFunctionFoo')),

        );
```

```
    }
    public function getFilters() {
        return array(
            new Twig_SimpleFilter('twig_filter_name' , array($this, 'getTwigFilterName')),
            new Twig_SimpleFilter('twig_filter_foo' , array($this, 'getTwigFilterFoo')),
        );
    }

    public function getName() {
        return 'ProjectTwigExtension';
    }
}
```

Registrar extensión en ramita

```
$twig = new Twig_Environment($loader);
$twig->addExtension(new ProjectTwigExtension());
```

Más opciones se pueden encontrar en los documentos oficiales.

Fecha de nacimiento simple para filtrar por edad

Cómo ...

1 - usa la clase de extensión de ramita que se extiende

```
use \Twig_Extension

class dobToAge extends \Twig_Extension {
```

2 - Agregue el filtro apropiado anulando el método getFilters ()

```
 public function getFilters() {
        return array(
            'age' => new \Twig_Filter_Method($this, 'getAge'),
        );
 }
```

3 - Agregue un poco de lógica para obtener la edad de una Fecha de nacimiento determinada

```
 public function getAge($date)
     {
        if (!$date instanceof \DateTime) {
        // turn $date into a valid \DateTime object or let return
        return null;
         }

     $referenceDate = date('01-01-Y');
     $referenceDateTimeObject = new \DateTime($referenceDate);
     $diff = $referenceDateTimeObject->diff($date);
     return $diff->y;
    }
}
```

Entonces, llame a su filtro de la siguiente manera,

```
{{ yourDateOfBirthInstance | age }}
```

Lea Ramita extensible en línea: https://riptutorial.com/es/twig/topic/4923/ramita-extensible

Capítulo 4: Sintaxis básica de la plantilla

Introducción

Explicación de construcciones de sintaxis de plantillas básicas

Examples

Variables de acceso

En las plantillas de Twig se puede acceder a las variables utilizando la notación de llaves dobles `{{ variableName }}`.

Ejemplo básico de saludo al usuario.

```
<!DOCTYPE html>
<html>
  <body>
    <span>Hello {{ name }}</span>
  </body>
</html>
```

Accediendo a los elementos del array.

La ramita como parámetro puede recibir matriz. Para acceder a un elemento específico de la matriz, puede usar la notación de corchete de acceso de la matriz php `{{ array[key] }}`.

Ejemplo anterior modificado para usar la matriz como parámetro

```
<!DOCTYPE html>
<html>
  <body>
    <span>Hello {{ user['name'] }}</span>
  </body>
</html>
```

Acceso a las propiedades del objeto.

Los objetos también se pueden pasar como un parámetro a la plantilla. La notación 'Dot' (.) Se usa para acceder a propiedades específicas del objeto `{{ object.propertyName }}`.

Mismo ejemplo con objeto como parámetro.

```
<!DOCTYPE html>
<html>
  <body>
    <span>Hello {{ user.name }}</span>
```

```
  </body>
</html>
```

Filtros

Las variables pueden ser modificadas utilizando filtros. Para aplicar el filtro a la variable siga el nombre de la variable con la tubería `|` y nombre del filtro:

```
{{ variable|filterName }}
```

Por ejemplo, para mostrar el valor de la variable en mayúsculas utilice la construcción.

```
{{ variable|upper }}
```

Los filtros pueden ser parametrizados. Los parámetros del filtro se pasan entre paréntesis como una lista separada por comas (,):

```
{{ variable|filterName(param1, param2, ...) }}
```

Para redondear el número a una precisión dada, podemos usar filtro `round` , acepta hasta 2 parámetros. El primero especifica la precisión (predeterminado: 0), el segundo método de redondeo (predeterminado: común).

Para redondear el número a 1 decimal usando el método común, puede usar `{{ number|round(1, 'common') }}` o `{{ number|round(1) }}` ya que el método predeterminado es común.

Los filtros también se pueden usar en objetos incrustados y variables de matriz:

```
{{ array['key'] | upper }} {{ object.text | upper }}
```

Los filtros también se pueden concatenar:

```
{% set array = "3,1,2"|split(',') %}

{{ array | sort | first }}
```

La lista de filtros básicos está disponible aquí.

Bloques condicionales

Las partes de la plantilla se pueden mostrar condicionalmente. `If` declaración se utiliza para este propósito. Es similar a la declaración `if` en lenguajes de programación. El contenido del bloque se ejecuta / muestra si una expresión se evalúa como `true` .

```
{% if enabled == false %}
    Disabled
{% endif %}
```

Desactivado se mostrará solo cuando `enabled` será igual a `false` .

Se pueden crear múltiples ramas usando `elseif` y `else` .

```
{% if temperature < 10 %}
    It's cold
{% elseif temperature < 18 %}
    It's chilly
{% elseif temperature < 24 %}
    It's warm
{% elseif temperature < 32 %}
    It's hot
{% else %}
    It's very hot
{% endif %}
```

En bucle

Para los bucles puede ser realmente útil en TWIG, permitiendo la creación de páginas web dinámicas de datos.

Digamos que creamos una simple serie de números:

```
{% set array = "3,1,2" %}
```

Luego podemos iterar sobre la matriz e imprimir lo que queramos. Cualquier dato dentro del bloque de matriz se generará en relación con la cantidad de datos dentro de la matriz. Este ejemplo imprimirá tres elementos h1 con los datos de la matriz concatenados.

```
{% for current in array %}
    <h1>This is number {{ current }} in the array </h1>
{% endear %}
```

Tenga en cuenta que `{{ current }}` se utilizó para acceder a los datos y no a una versión de la `array`

Ver este trabajo: https://twigfiddle.com/mxwkea/2

Otro ejemplo de esto podría ser usar objetos. Por ejemplo, supongamos que tenemos un objeto Entidad con el campo 'nombre' junto con sus *captadores y definidores* relevantes. Si varias de estas entidades también se almacenan dentro del Array, se puede acceder a ellas como a cualquier otro Objeto dentro de TWIG:

```
{% for currentObject in ArrayOfObjects %}
    {{ currentObject.name }}
{% endfor %}
```

Vea esto trabajando con datos JSON: https://twigfiddle.com/mxwkea

Operador ternario (taquigrafía si-entonces-más) y operador de fusión nula

El operador ternario (`?:`

El soporte para el operador ternario extendido se agregó en **Twig 1.12.0** .

```
{{ foo ? 'yes' : 'no' }}
```

Evalúa:

> si `foo` echo `yes` mas echo `no`

```
{{ foo ?: 'no' }}
```

o

```
{{ foo ? foo : 'no' }}
```

Evalúa:

> si `foo` eco, si `no` echo `no`

```
{{ foo ? 'yes' }}
```

o

```
{{ foo ? 'yes' : '' }}
```

Evalúa:

> si `foo` echo `yes` si no echo nada

El operador de unión nula (`??:`

```
{{ foo ?? 'no' }}
```

Evalúa:

> Devuelve el valor de `foo` si **está definido** y **no es nulo** , de lo contrario `no`

Manejo de espacios en blanco

Para eliminar espacios en blanco (espacios, pestañas, nuevas líneas ...) entre etiquetas HTML, use una etiqueta `spaceless` espacio:

```
{% spaceless %}

    <div>
        <span>foo bar </span>
    </div>
{% endspaceless %}
{# produces output <div><strong>foo bar </strong></div> #}
```

Si necesita eliminar los espacios en blanco en un nivel por etiqueta, use el *modificador de control de espacios en blanco,* es decir, guión (-). Usándolo, puede recortar espacios en blanco iniciales o finales:

```
{% set value = 'foo bar' %}

<span>  {{- value }}  </span>
{# produces '<span>foo bar  </span>' #}

<span>  {{ value -}}  </span>
{# produces '<span>  foo bar</span>' #}

<span>  {{- value -}}  </span>
{# produces '<span>foo bar</span>' #}

<span {%- if true  %} class="foo"{% endif %}>
{# produces '<span class="foo">' #}

<span {%- if false %} class="foo"{% endif %}>
{# produces '<span>' #}
```

Lea Sintaxis básica de la plantilla en línea: https://riptutorial.com/es/twig/topic/8697/sintaxis-basica-de-la-plantilla

Capítulo 5: Uso avanzado

Observaciones

Tenga en cuenta que la extensión Twig no es compatible con PHP7 y, aunque hay una solicitud de extracción para resolver esta situación, aún no se incorporó a la versión 1.x. Con la versión más reciente de Twig 2.x, la extensión C se eliminó pero podría ser parte de 2.1 . Algunos puntos de referencia revelan mejoras mínimas con la extensión C en PHP7.

Examples

Construyendo la extensión C

La extensión C es una característica opcional de Twig que ofrece algunas mejoras de rendimiento en la representación de plantillas. El código fuente de la extensión se encuentra en el directorio de código fuente de Twig en `ext/twig` . Se compila como cualquier otra extensión de PHP:

```
cd ext/twig
phpize
./configure
make
make install
```

Habilitar la extensión en el archivo `php.ini` . Una vez que se instale la extensión, Twig lo detectará automáticamente y lo utilizará en el tiempo de ejecución.

Lea Uso avanzado en línea: https://riptutorial.com/es/twig/topic/8696/uso-avanzado

Creditos

S. No	Capítulos	Contributors
1	Empezando con la ramita	4444, Community, DarkBee, Icode4food, insertusernamehere, mleko, Object, Paulpro, user1950896
2	Herencia de plantillas	DarkBee, Ḟ̄ą́ḿíṅḡ ỏḿƀíé, nicolallias, PastyAndPeas, Trix
3	Ramita extensible	DarkBee, Object
4	Sintaxis básica de la plantilla	jkucharovic, mleko, PastyAndPeas, Trix
5	Uso avanzado	Icode4food, mleko, Ricardo Velhote