Universidad de La Laguna

Tecnologías Web: Cliente

Jekyll.

Jekyll

- Generador de sitios estáticos
- Markdown para la composición de publicaciones y marcado Liquid para la construcción de plantillas.
- Es una plataforma pensada para blogs, pero se puede usar para cualquier sitio web estático.
- Incorpora Sass
- Es altamente configurable, extensible mediante plugins y tiene disponible una amplia gama de temas ya listos para su uso
- La configuración del servidor es sencilla
- Perdemos la capacidad de generar contenido dinámicamente.

Instalación de Jekyll

- Jekyll se instala como una gema de ruby:
  - Las dependencias del proyecto se especifican en el fichero **`Gemfile`**
  - El gestor de paquetes: **`bundle`**
- Comandos:
  - **`jekyll new [nombre_directorio]:`** Genera la estructura del proyecto
  - **`jekyll build:`** Genera el contenido del sitio
  - **`jekyll serve:`** lanza un servidor local para el desarrollo, `https://localhost.4000`

```
$ sudo apt-get install ruby

$ sudo gem install jekyll bundler

$ jekyll new myJekyllWeb

$ ls
404.html  about.markdown  _config.yml  Gemfile  index.markdown  _posts
```

Estructura del proyecto

```
├── _config.yml #Fichero de configuración
├── _data #Datos en el sitio: json, yaml, csv
│     ├── members.yml
│
├── _drafts #Borradores, no se publican. Se pueden visualizar en
│           //el servidor local con la opción --drafts
│
├── _includes #Parciales con etiquetado liquids que se pueden
│             #reutilizar.
├── _layouts  #Son las plantillas de los posts/páginas.
│
├── _posts    #Contenido dinámico.
│
├── _sasss   #Parciales sass
│    ├── _base.scss
│    └── _
├── _site #Ubicación del sitio generado.
│
├── .jekyll-metadata #Seguimiento de jekyll de los ficheros que no
│                    #necesitan ser actualizados.
├── index.html
│
└── Otros    #Cualquier otro fichero
```

Configuración del proyecto (_config.yml):

- Se especifican variables que controlan el proceso de generación del sitio web estático.
- Se pueden utilizar variables jekyll o definir nuevas.
- Se pueden usar en las plantillas para modificar el comportamiento
  - **`site, destination, layouts, encoding, title, name, description, email, ...`**
- Las variables se usan con el prefijo site: **`site.title`**

```
$ # Inicio
name:
description: Ejemplos Jekyll
themeJekyll: Amanixer
version: 0.1.0
timezone: Europe/Madrid

# Configuracion
url: http://misitio.com/blog
baseurl: /blog
paginate: 5
paginate_path: "page:num"
highlighter: pygments
markdown: kramdown
```

Código YAML:

- Lenguaje para la serialización válido para numerosos lenguajes de programación.
- variable: valor
- variable: [valor1, valor2, ..., valorn]

```
variable:
  - valor1
  - valor2
...
  - valorn
```

- estructura:

```
clave1: valor1
clave2: valor
```

Permalinks:

- Rutas para las páginas que se generan.
- Permiten establecer rutas diferentes a las del código en la salida
- Se pueden especificar:
  - Fichero de configuración
  - En las plantillas
- La sintaxis consiste en encadenar marcadores de posición en la ruta
- Ejemplo:
  - `/:categories/:year/:month/:day/:title:output_ext #post`
  - `/:title:outp_ext #página`
- Place holders que se puede usar: `year, month, week, day, categories`, ...

```
permalink: mypages  # => http://misitio.com/mypages/
```

Liquid:

- Lenguaje de especificación de plantillas en Jekyll.
- El código se estructura mediante:
  - Objetos: especifican donde incluir contenido
    - `{{...}} {{-, -}}`
  - Etiquetas: Establecen el control del flujo en las plantillas. No originan ningún contenido.
    - `{%...%} , {%-, and -%}`
  - Filtros: Modifican la salida de un objeto. Se aplican a una salida.
    - Separador de filtros: |

```
{{ page.title }}


{% if user %}
  Hello {{ user.name }}!
{% endif %}
```

```
{{ "/my/fancy/url" | append: ".html" }}
```

Liquid. Tags

- Control:
  - `if, elseif/else, unless, case/when`
- Comentarios:
  - `{% comment %} Texto del comentario {% endcomment %}`
- Bucles:
  - `for, range, if, break, continue, tablerow, ...`
- Raw: Deja de compilar las etiquetas, para evitar conflictos de sintaxis
- Variable: Crea nuevas variables
  - `assign, capture, increment, decrement`

Front Matter

- Texto preliminar para indicar que Jekyll debe procesar el archivo y cómo lo debe procesar.
- Formato YAML válido
- Se incluye entre 2 líneas de 3 guiones.
- Generalmente especifica un layout y metadatos de la página.
- Se pueden utilizar variables predefinidas, que estarán accesibles en el resto del fichero y layouts e includes de la página.
- Variables típicas en matter:
  - `layout, permalink, published, date, category, categories, tags`
- **Si se usan tags y variables de Liquid, aunque no se especifique nada, se debe incluir la zona front matter vacía.**
- Se pueden usar variables personalizadas

```
---
layout: mipagina
title: Aprendiendo Jekyll
---
```

Liquid. Filters:

```
$ {{ -17 | abs }} # => 17

$ #Convierte en un array elementos separados por el separador indicado
{% assign beatles = "John, Paul, George, Ringo" | split: ", " %}
```

```
#Envía a la salida los elementos en el array
{% for member in beatles %}
  {{ member }}
{% endfor %}

     =>   John
          Paul
          George
          Ringo
```

```
This page was last updated at {{ "now" | date: "%Y-%m-%d %H:%M"
}}.
```

Páginas

- Contenido que no está vinculado a una fecha.
- Incluirlas en el directorio raíz o en un subdirectorio como fichero html
- También se admite markdown, que se transpilará a html
- Se pueden determinar en qué url se despliegan con permalinks

Posts:

- Se especifican como un fichero de texto y jekyll los convierte en posts del blog.
- Para crearlo se añade un fichro con nombre:
  - YEAR-MONTH-DAY-title.MARKUP
  - MARKUP puede ser md o html
- Debe incluir el **`front matter`**
- En el front matter se pueden añadir categorías y etiquetas.

Includes

- Permiten incluir código en otro fichero.
- Se utiliza la etiqueta include
  - `{% include nombrefichero.html %}`
- El fichero a incluir se busca en el directorio _includes
- Se pueden usar rutas relativas con la etiqueta include_relative
  - `{% include_relative nombredir/nombrefichero.html %}`
- Se pueden usar variables.
- Se pueden usar parámetros.

```
   <a href="{{ include.url }}">
     <img src="{{ include.file }}" alt="{{ include.alt }}"/>
   </a>

Parámetros: url - file - alt

{% include image.html url="http://mipagina.com" file="logo.png" alt="Mi logo"
%}
```

Layouts

- Permiten definir plantillas jekyll que eviten la repetición de código.
- Se incluyen en el directorio _layouts
- Por convenio se crea el layout default.html del que heredan otros posibles layouts.
- En las páginas en las que se va a usar un layout se especifica en el front matter:
- La herencia en los layouts se logra definiendo un nuevo layout que tenga en el front matter el layout del que hereda y que incorpore las diferencias con el layout original

```
---
  title: Ejemplo layout
  layout: default
---
```

```
---
  title: Ejemplo herencia layout
  layout: default
---
  <p>{{ page.date }} - Written by {{ page.author }}</p>


  {{ content }}
```

Assets

- Se pueden usar ficheros Sass y CoffeeScript.
- Se crean con el front matter vacío.
- Los ficheros Sass se graban con su extensión .sass o .scss y Jekyll los transpila al formato css
- Se pueden especificar directivas para la transpilación
    - Elegir el directorio donde se buscan los parciales

      sass:

        sass_dir: _sass