Programación web de CS50 con Python y JavaScript

OpenCourseWare

Donar (https://cs50.harvard.edu/donate)

Brian Yu (https://brianyu.me)
brian@cs.harvard.edu

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

wiki

> CS50W no presenta una correspondencia uno a uno entre conferencias y proyectos. Si estás intentando este proyecto sin haber visto al menos la Lección 3, ¡lo estás intentando demasiado pronto!

Diseñar una enciclopedia en línea similar a Wikipedia.

Fondo

Wikipedia (https://www.wikipedia.org/) es una enciclopedia en línea gratuita que consta de varias entradas de enciclopedia sobre diversos temas.

Cada entrada de la enciclopedia se puede ver visitando la página de esa entrada. Visitar https://en.wikipedia.org/wiki/HTML (https://en.wikipedia.org/wiki/HTML) , por ejemplo, muestra la entrada de Wikipedia para HTML. Observe que el nombre de la página solicitada (HTML) se especifica en la ruta `/wiki/HTML`. Reconozca también que el contenido de la página debe ser simplemente HTML que representa su navegador.

En la práctica, empezaría a resultar tedioso si cada página de Wikipedia tuviera que estar escrita en HTML. En cambio, puede resultar útil almacenar entradas de enciclopedia utilizando un lenguaje de marcado más ligero y amigable para los humanos. Wikipedia usa un lenguaje de marcado llamado Wikitext (https://en.wikipedia.org/wiki/Help:Wikitext) , pero para este proyecto almacenaremos entradas de enciclopedia usando un lenguaje de marcado llamado Markdown.

Lea la guía Markdown de GitHub (https://help.github.com/en/github/writing-on-github/basic-writing-and-formatting-syntax) para comprender cómo funciona la sintaxis de Markdown. Preste atención en particular a cómo se ve la sintaxis de Markdown para títulos, texto en negrita, enlaces y listas.

Al tener un archivo Markdown que represente cada entrada de la enciclopedia, podemos hacer que nuestras entradas sean más fáciles de escribir y editar. Sin embargo, cuando un usuario ve la entrada de nuestra enciclopedia, necesitaremos convertir ese Markdown a HTML antes de mostrárselo al usuario.

Empezando

- Descargue el código de distribución de https://cdn.cs50.net/web/2020/spring/projects/1/wiki.zip (https://cdn.cs50.net/web/2020/spring/projects/1/wiki.zip) y descomprímalo.

Comprensión

En el código de distribución hay un proyecto Django llamado `wiki` que contiene una única aplicación llamada `encyclopedia`.

Primero, abra `encyclopedia/urls.py`, donde se define la configuración de URL para esta aplicación. Observe que comenzamos con una única ruta predeterminada asociada con la `views.index` función.

A continuación, mira `encyclopedia/util.py`. No necesitarás cambiar nada en este archivo, pero observa que hay tres funciones que pueden resultar útiles para interactuar con las entradas de la enciclopedia. `list_entries` devuelve una lista de los nombres de todas las entradas de la enciclopedia guardadas actualmente. `save_entry` guardará una nueva entrada de la enciclopedia, dado su título y parte del contenido de Markdown. `get_entry` recuperará una entrada de enciclopedia por su título y devolverá su contenido de Markdown si la entrada existe o `None` si no existe. Cualquiera de las vistas que escriba puede utilizar estas funciones para interactuar con las entradas de la enciclopedia.

Cada entrada de la enciclopedia se guardará como un archivo Markdown dentro del `entries/` directorio. Si marca allí ahora, verá que hemos creado previamente algunas entradas de muestra. ¡Puedes agregar más!

Ahora, echemos un vistazo `encyclopedia/views.py`. Sólo hay una vista aquí ahora, la `index` vista. Esta vista devuelve una plantilla `encyclopedia/index.html`, proporcionando a la plantilla una lista de todas las entradas de la enciclopedia (obtenida llamando a `util.list_entries`, que vimos definida en `util.py`).

Puedes encontrar la plantilla mirando `encyclopedia/templates/encyclopedia/index.html`. Esta plantilla hereda de un `layout.html` archivo base y especifica cuál debería ser el título de la página y qué debería haber en el cuerpo de la página: en este caso, una lista desordenada de todas las entradas de la enciclopedia. `layout.html`, mientras tanto, define la estructura más amplia de la página: cada página tiene una barra lateral con un campo de búsqueda (que por ahora no hace nada), un enlace para ir a casa y enlaces (que aún no funcionan) para crear una nueva página o visita una página aleatoria.

Especificación

Completa la implementación de tu enciclopedia Wiki. Debes cumplir con los siguientes requisitos:

- **Página de entrada** : Al visitar `/wiki/TITLE`, donde `TITLE` está el título de una entrada de enciclopedia, se debe mostrar una página que muestra el contenido de esa entrada de enciclopedia.
  - La vista debe obtener el contenido de la entrada de la enciclopedia llamando a la `util` función apropiada.
  - Si se solicita una entrada que no existe, se le debe presentar al usuario una página de error que indica que no se encontró la página solicitada.
  - Si la entrada existe, se le debe presentar al usuario una página que muestra el contenido de la entrada. El título de la página debe incluir el nombre de la entrada.
- **Página de índice** : actualización `index.html` de modo que, en lugar de simplemente enumerar los nombres de todas las páginas de la enciclopedia, el usuario pueda hacer clic en el nombre de cualquier entrada para ir directamente a esa página de entrada.
- **Buscar** : permite al usuario escribir una consulta en el cuadro de búsqueda de la barra lateral para buscar una entrada de enciclopedia.
  - Si la consulta coincide con el nombre de una entrada de la enciclopedia, el usuario debe ser redirigido a la página de esa entrada.
  - Si la consulta no coincide con el nombre de una entrada de la enciclopedia, el usuario debería ser llevado a una página de resultados de búsqueda que muestra una lista de todas las entradas de la enciclopedia que tienen la consulta como subcadena. Por ejemplo, si la consulta de búsqueda fue `ytho`, `Python` debería aparecer en los resultados de la búsqueda.
  - Al hacer clic en cualquiera de los nombres de las entradas en la página de resultados de búsqueda, el usuario debería acceder a la página de esa entrada.

- **Nueva página** : al hacer clic en "Crear nueva página" en la barra lateral, el usuario debería acceder a una página donde puede crear una nueva entrada de enciclopedia.
  - Los usuarios deberían poder ingresar un título para la página y, en un archivo `textarea` (https://www.w3schools.com/tags/tag_textarea.asp), deberían poder ingresar el contenido de Markdown para la página.
  - Los usuarios deberían poder hacer clic en un botón para guardar su nueva página.
  - Cuando se guarda la página, si ya existe una entrada de enciclopedia con el título proporcionado, se le debe presentar al usuario un mensaje de error.
  - De lo contrario, la entrada de la enciclopedia debe guardarse en el disco y el usuario debe ser llevado a la página de la nueva entrada.
- **Editar página** : en cada página de entrada, el usuario debe poder hacer clic en un enlace para ir a una página donde puede editar el contenido de Markdown de esa entrada en un archivo `textarea` .
  - Debe `textarea` rellenarse previamente con el contenido de Markdown existente de la página. (es decir, el contenido existente debe ser la inicial `value` de `textarea`).
  - El usuario debería poder hacer clic en un botón para guardar los cambios realizados en la entrada.
  - Una vez guardada la entrada, el usuario debe ser redirigido nuevamente a la página de esa entrada.
- **Página aleatoria** : al hacer clic en "Página aleatoria" en la barra lateral, el usuario debería acceder a una entrada de enciclopedia aleatoria.
- **Conversión de Markdown a HTML** : en la página de cada entrada, cualquier contenido de Markdown en el archivo de entrada debe convertirse a HTML antes de mostrarse al usuario. Puede utilizar el `python-markdown2` (https://github.com/trentm/python-markdown2)paquete para realizar esta conversión, instalable mediante `pip3 install markdown2` .
  - Desafío para aquellos más cómodos: si se siente más cómodo, intente implementar la conversión de Markdown a HTML sin utilizar bibliotecas externas, sin admitir encabezados, texto en negrita, listas desordenadas, enlaces y párrafos. Puede que le resulte útil utilizar expresiones regulares en Python (https://docs.python.org/3/howto/regex.html) .

Consejos

- De forma predeterminada, al sustituir un valor en una plantilla de Django, Django HTML escapa del valor para evitar generar HTML no deseado. Si desea permitir que se genere una cadena HTML, puede hacerlo con el `safe` (https://docs.djangoproject.com/en/4.0/ref/templates/builtins/#safe)filtro (como agregando `|safe` después del nombre de la variable que está sustituyendo).

▶ **¿Usando el espacio de código CS50?**

Cómo enviar

1. Visite este enlace (https://submit.cs50.io/invites/89679428401548238ceb022f141b9947) , inicie sesión con su cuenta de GitHub y haga clic en **Autorizar cs50** . Luego, marque la casilla que indica que desea otorgar acceso al personal del curso a sus envíos y haga clic en **Unirse al curso** .
2. Instale Git (https://git-scm.com/downloads) y, opcionalmente, instale `submit50` (https://cs50.readthedocs.io/submit50/) .

Cuando envíe su proyecto, el contenido de su `web50/projects/2020/x/wiki` rama debe coincidir con la estructura de archivos del código de distribución descomprimido tal como se recibió originalmente. Es decir, sus archivos no deben estar anidados dentro de ningún otro directorio de su propia creación. Su rama tampoco debe contener ningún código de ningún otro proyecto, solo este. Si no cumple con esta estructura de archivos, es probable que su envío sea rechazado.

A modo de ejemplo, para este proyecto eso significa que si el personal de calificación visita `https://github.com/me50/USERNAME/tree/web50/projects/2020/x/wiki` (donde está su propio nombre de usuario de GitHub como se proporciona en el formulario a continuación) , `USERNAME` deberíamos ver los tres subdirectorios (`encyclopedia` ,,) y el archivo. Si no es así como está organizado su código cuando lo verifica, reorganice su repositorio según sea necesario para que coincida con este paradigma. `entries` `wiki` `manage.py`

3. Si lo ha instalado `submit50` , ejecute

   ```
   submit50 web50/projects/2020/x/wiki
   ```

   De lo contrario, usando Git, envíe su trabajo a `https://github.com/me50/USERNAME.git` , donde `USERNAME` está su nombre de usuario de GitHub, en una rama llamada `web50/projects/2020/x/wiki` .
4. Grabe un screencast (https://www.howtogeek.com/205742/how-to-record-your-windows-mac-linux-android-or-ios-screen/) que no exceda los 5 minutos de duración (y no lo cargue más de un mes antes de enviar este proyecto), en el que demuestre la funcionalidad de su proyecto. Asegúrese de que cada elemento de la especificación anterior se demuestre en su video. No es necesario mostrar su código en este video, solo su aplicación en acción; Revisaremos su código en GitHub. Sube ese video a YouTube (https://www.youtube.com/upload) (como no listado o público, pero no privado) o en otro lugar. En la descripción de su video, debe marcar la hora en la que su video demuestra cada uno de los siete (7) elementos principales de la especificación. **Esto no es opcional** , los videos sin marcas de tiempo en su descripción serán rechazados automáticamente.
5. Envíe este formulario (https://forms.cs50.io/f6c75c7d-8939-4b66-855e-6cc827c868cf) .

Luego puede ir a https://cs50.me/cs50w (https://cs50.me/cs50w) para ver su progreso actual.