Modelo-Vista-Controlador (MVC) explicado - Con Legos

Para demostrar cómo funciona en la práctica una aplicación web estructurada con el patrón **Modelo-Vista-Controlador** (o **MVC**), hagamos un viaje por el camino de la memoria ...

¡Legos!

Tienes diez años, estás sentado en el piso de tu sala familiar y frente a ti hay un gran cubo de Legos. Hay Legos de diferentes formas y tamaños. Algunos azules, altos y largos. Como un remolque de tractor. Algunos rojos y casi en forma de cubo. Y algunos son amarillos, planos grandes y anchos, como láminas de vidrio. Con todos estos diferentes tipos de Legos, no se sabe lo que podrías construir.

Pero sorpresa, sorpresa, ya hay una **solicitud** . Tu hermano mayor corre y dice: “¡Oye! ¡Constrúyeme una nave espacial! "

"Está bien", piensas, "¡eso podría ser realmente genial!" Es una nave espacial.

Así que ponte manos a la obra. Empiezas a sacar los Legos que crees que vas a necesitar. Algunos grandes, otros pequeños. Diferentes colores para el exterior de la nave espacial, diferentes colores para los motores. Ah, y diferentes colores para las pistolas bláster. (¡Debes tener pistolas bláster!)

Ahora que tiene todos los **bloques de construcción** en su lugar, es hora de ensamblar la nave espacial. Y después de unas horas de arduo trabajo, ahora tienes frente a ti: ¡una nave espacial!

Corres a buscar a tu hermano para mostrarle el producto terminado. “¡Vaya, buen trabajo!”, Dice. “Eh”, piensa, “Solo lo pedí hace unas horas, no tenía que hacer nada, y ahí está. Ojalá *todo* fuera así de fácil ".

¿Qué pasaría si te dijera que construir una aplicación web es exactamente como construir con Legos?

Todo comienza con una *solicitud* ...

En el caso de los Legos, fue tu hermano quien te pidió que construyeras algo. En el caso de una aplicación web, es un usuario que ingresa una URL y solicita ver una determinada página.

Entonces tu hermano es el usuario.

La solicitud llega al *controlador* ...

Con los Legos, eres el controlador.

El controlador es responsable de tomar todos los **bloques de construcción** necesarios y organizarlos según sea necesario.

Esos bloques de construcción se conocen como *modelos* ...

Los diferentes tipos de Legos son los modelos. Tienes todos los tamaños y formas diferentes, y agarras los que necesitas para construir la nave espacial. En una aplicación web, los modelos ayudan al controlador a recuperar toda la información que necesita de la base de datos.

Entonces llega la solicitud ...

El controlador (usted) recibe la solicitud.

Va a los modelos (Legos) para recuperar los elementos necesarios.

Y ahora todo está en su lugar para producir el producto final.

El producto final se conoce como la *vista* ...

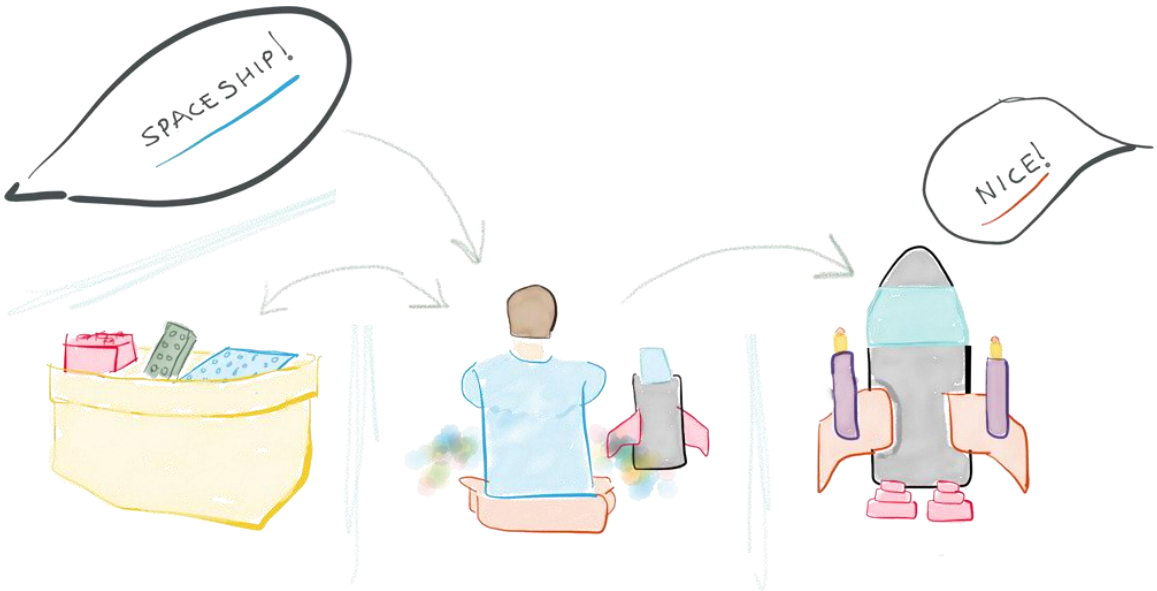
La nave espacial es la vista. Es el producto final que finalmente se muestra a la persona que hizo la solicitud (su hermano).

En una aplicación web, la vista es la página final que el usuario ve en su navegador.

Para resumir...

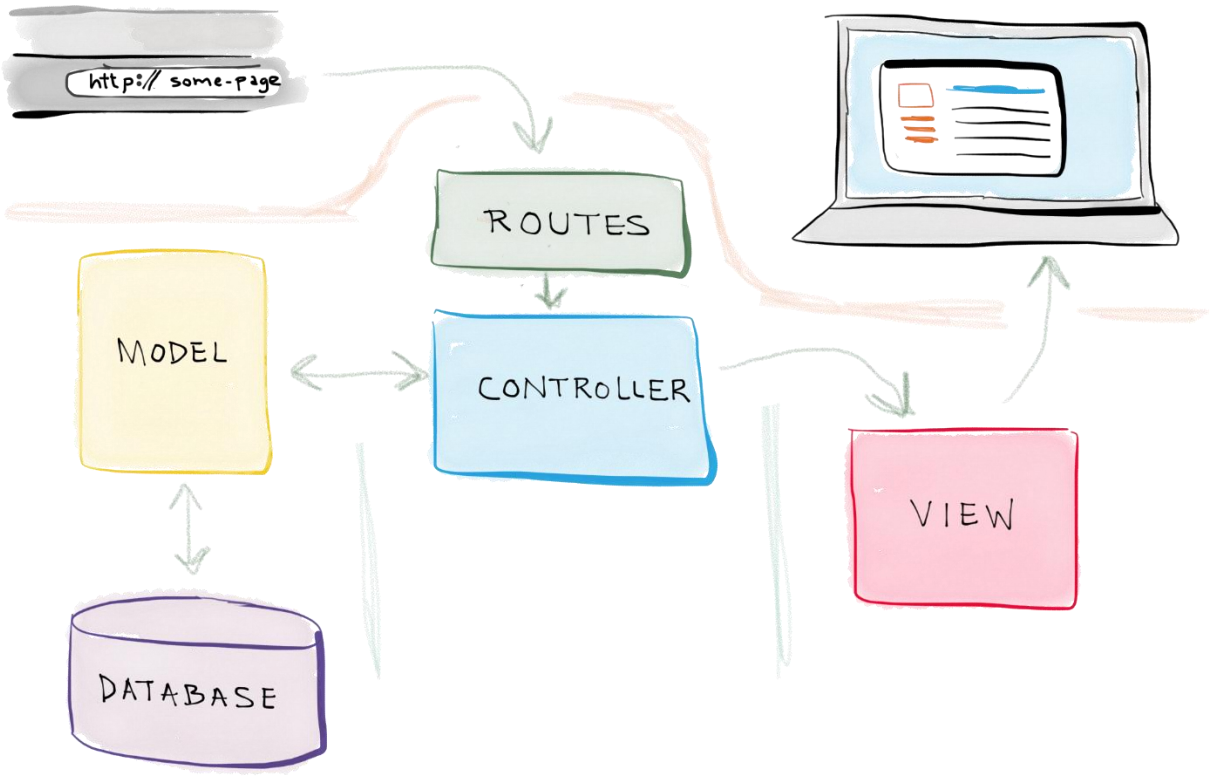
Al construir con Legos:

1. Tu hermano te pide que construyas una nave espacial.
2. Recibe la solicitud.
3. Recupera y organiza todos los Legos que necesitas para construir la nave espacial.
4. Usas los Legos para construir la nave espacial y presentar la nave espacial terminada a tu hermano.



Y en una aplicación web:

1. Un usuario solicita ver una página ingresando una URL.
2. El **controlador** recibe esa solicitud.
3. Utiliza los **modelos** para recuperar todos los datos necesarios, los organiza y los envía a ...
4. **View** , que luego usa esos datos para representar la página web final presentada al usuario en su navegador.



Desde un punto de vista más técnico

Con la funcionalidad MVC resumida, profundicemos un poco más y veamos cómo funciona todo en un nivel más técnico.

Cuando escribe una URL en su navegador para acceder a una aplicación web, está solicitando ver una determinada página dentro de la aplicación. Pero ¿cómo sabe la aplicación qué página mostrar / representar?

Al crear una aplicación web, define lo que se conoce como **rutas** . Las rutas son, esencialmente, patrones de URL asociados con diferentes páginas. Entonces, cuando alguien ingresa una URL, detrás de escena, la aplicación intenta hacer coincidir esa URL con una de estas rutas predefinidas.

Entonces, de hecho, hay *cuatro* componentes principales en juego: **rutas**, **modelos**, **vistas** y **controladores** .

Rutas

Cada ruta está asociada con un controlador, más específicamente, una determinada función *dentro de* un controlador, conocida como **acción del controlador** . Entonces, cuando ingresa una URL, la aplicación intenta encontrar una ruta coincidente y, si tiene éxito, llama a la acción del controlador asociada a esa ruta.

Veamos una ruta básica de Flask como ejemplo:

```
@app.route('/')
def main_page():
    pass
```

Aquí establecemos la `/`ruta asociada a la `main_page()`función de visualización.

Modelos y controladores

Dentro de la acción del controlador, normalmente ocurren dos cosas principales: los modelos se utilizan para recuperar todos los datos necesarios de una base de datos; y esos datos se pasan a una vista, que muestra la página solicitada. Los datos recuperados a través de los modelos generalmente se agregan a una estructura de datos (como una lista o diccionario), y esa estructura es lo que se envía a la vista.

Volviendo a nuestro ejemplo de matraz:

```
@app.route('/')
def main_page():
    """Searches the database for entries, then displays them."""
    db = get_db()
    cur = db.execute('select * from entries order by id desc')
    entries = cur.fetchall()
    return render_template('index.html', entries=entries)
```

Ahora, dentro de la función de vista, tomamos datos de la base de datos y realizamos una lógica básica. Esto devuelve una lista, que asignamos a la variable `entries`, que es accesible dentro de la plantilla *index.html* .

Puntos de vista

Finalmente, en la vista, se accede a esa estructura de datos y la información contenida en ella se utiliza para representar el contenido HTML de la página que el usuario ve en última instancia en su navegador.

Nuevamente, volviendo a nuestra aplicación Flask, podemos recorrer el `entries`, mostrando cada uno usando la sintaxis Jinja:

```
{% for entry in entries %}
  <li>
    <h2>{{ entry.title }}</h2>
    <div>{{ entry.text|safe }}</div>
  </li>
{% else %}
  <li><em>No entries yet. Add some!</em></li>
{% endfor %}
```

Resumen

Entonces, un resumen técnico más detallado del proceso de solicitud de MVC es el siguiente:

1. Un usuario solicita ver una página ingresando una URL.
2. La aplicación hace coincidir la URL con una **ruta** predefinida .
3. Se llama a la **acción del controlador** asociada con la ruta.
4. La acción del controlador usa los **modelos** para recuperar todos los datos necesarios de una base de datos, coloca los datos en una matriz y carga una **vista** , pasando a lo largo de la estructura de datos.
5. La **vista** accede a la estructura de datos y la utiliza para representar la página solicitada, que luego se presenta al usuario en su navegador.