Actividad 4 del complementario nodejs

NOMBRE DEL INTEGRANTE

Valentina Hernández calderón

SERVICIO NACIONAL DE APRENDIZAJE SENA

CENTRO AGROEMPRESARIAL Y DESARROLLO PECUARIO DEL HUILA

TECNOLOGO EN A.D.S.O

CIUDAD DE GARZÓN HUILA

AÑO DE 2023

¿Cómo se integra Bootstrap en una aplicación Express?

Configuración de la aplicación Express:

Primero, asegúrate de tener Node.js y npm instalados. Luego, crea una nueva carpeta para tu proyecto y abre una terminal en esa ubicación. Ejecuta el siguiente comando para inicializar un nuevo proyecto de Node.js y generar un archivo package.json:

Forma

Descripción generada automáticamente con confianza media

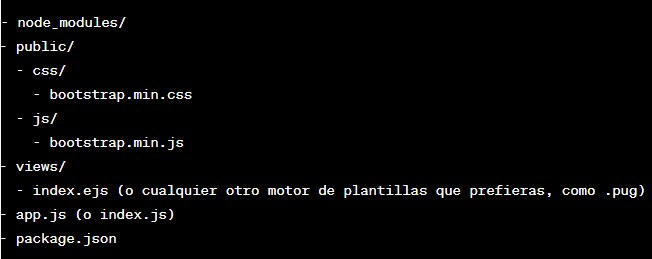
A continuación, instala Express y Bootstrap como dependencias de tu proyecto:

Forma

Descripción generada automáticamente con confianza media

Estructura de carpetas y archivos:

Asegúrate de tener una estructura de carpetas como la siguiente:



Integrar Bootstrap en tu aplicación Express:

Ahora, vamos a configurar nuestra aplicación Express para que pueda servir archivos estáticos, incluyendo los archivos CSS y JS de Bootstrap. En el archivo principal de la aplicación (por ejemplo, app.js o index.js), agrega el siguiente código:

Texto

Descripción generada automáticamente

Enlazar Bootstrap en la vista:

Dentro de la carpeta views, crea un archivo llamado index.ejs (o con el nombre que desees, pero asegúrate de que coincida con la ruta en el código de la aplicación). En este archivo, enlaza el archivo CSS de Bootstrap en la sección <head> y también puedes agregar los archivos JS de Bootstrap justo antes de cerrar la etiqueta </body>:

Texto

Descripción generada automáticamente

Ejecutar la aplicación:

Una vez que hayas realizado los pasos anteriores, ejecuta tu aplicación Express con el siguiente comando en la terminal:

Forma

Descripción generada automáticamente con confianza media  
Si todo está configurado correctamente, la aplicación debería iniciarse y podrás acceder a ella a través de tu navegador en http://localhost:3000.

Con estos pasos, habrás integrado Bootstrap en tu aplicación Express y podrás utilizar los estilos y componentes de Bootstrap en tus vistas.

¿Para qué sirve el archivo package.json?

El archivo package.json es un archivo fundamental en proyectos de Node.js y npm (Node Package Manager). Sirve para definir la configuración del proyecto, incluyendo su nombre, versión, dependencias, scripts, entre otras cosas. A continuación, te explico las principales funciones del archivo package.json:

1. Gestión de dependencias:

Una de las funciones principales del package.json es especificar las dependencias del proyecto. Estas son las bibliotecas y paquetes que tu proyecto necesita para funcionar correctamente. Pueden ser librerías propias de Node.js o de terceros. Cuando ejecutas el comando npm install (o npm i), npm lee el package.json y descarga todas las dependencias especificadas en la carpeta node\_modules.

1. Versión y autoría del proyecto:

El package.json contiene información sobre la versión actual del proyecto, el autor, la descripción y otra información relevante. Esto es útil para mantener un registro de la evolución del proyecto y su propósito.

1. Configuración de scripts:

En el package.json, puedes definir scripts personalizados que pueden ser ejecutados con el comando npm run nombre-del-script. Por ejemplo, puedes tener un script para iniciar tu aplicación, otro para ejecutar pruebas, uno para compilar el código, entre otros. Esto facilita la automatización de tareas comunes en el desarrollo.

1. Metadata del proyecto:

El package.json también puede contener metadatos adicionales, como la licencia bajo la cual se distribuye el código, las palabras clave que describen el proyecto, el repositorio de origen, las dependencias de desarrollo (paquetes necesarios solo en tiempo de desarrollo), etc.

1. Control de versiones y resolución de conflictos:

Cuando colaboras en un proyecto con otros desarrolladores, el package.json asegura que todos tengan las mismas versiones de las dependencias. Esto evita conflictos y problemas de incompatibilidad, ya que npm instalará las versiones específicas de las dependencias listadas en el archivo.

1. Distribución de paquetes:

En el caso de que desees publicar tu propio paquete de Node.js en el registro de npm para que otros lo puedan instalar, el package.json es esencial. Incluye información relevante para publicar y compartir el paquete con la comunidad de desarrollo de Node.js.

En resumen, el archivo package.json es una parte crucial en el desarrollo de proyectos con Node.js y npm, ya que facilita la gestión de dependencias, la ejecución de scripts, la descripción del proyecto y otras tareas importantes para el desarrollo y distribución del software.

¿Qué es y para qué sirve EJS (Embedded JavaScript)?

EJS (Embedded JavaScript) es un motor de plantillas para JavaScript que te permite generar vistas dinámicas en aplicaciones web. Se utiliza comúnmente en el desarrollo de aplicaciones basadas en Node.js y Express, aunque también puede utilizarse en otros entornos de servidor y cliente.

EJS te permite mezclar código JavaScript con HTML para crear plantillas reutilizables y dinámicas. A través de la inclusión de marcadores especiales, EJS puede interpretar y renderizar el código JavaScript dentro de las plantillas y generar HTML con contenido dinámico.

Algunas características clave de EJS son:

1. Plantillas reutilizables: Puedes crear plantillas que contengan el diseño general de tus páginas web y luego reutilizarlas para generar diferentes páginas con contenido dinámico.
2. Inclusión de datos dinámicos: EJS te permite insertar datos dinámicos en las plantillas mediante el uso de etiquetas especiales <%= %>. Estas etiquetas se utilizan para mostrar variables o resultados de expresiones JavaScript en la página generada.
3. Estructuras de control: Puedes utilizar estructuras de control como bucles y condicionales en tus plantillas para crear contenido que cambie según las condiciones o el número de elementos en una lista.
4. Partes condicionales del código: EJS también admite etiquetas de código JavaScript para incluir partes condicionales o más complejas del código.

Para usar EJS en una aplicación Express, primero debes configurar la aplicación para que utilice EJS como su motor de plantillas. Esto se hace mediante el siguiente código en tu archivo principal de la aplicación (por ejemplo, app.js):

Texto

Descripción generada automáticamente

Luego, puedes renderizar vistas utilizando EJS en las rutas de tu aplicación. Por ejemplo:

Texto

Descripción generada automáticamente

En el ejemplo anterior, se renderiza la vista index.ejs, y se pasa un objeto data con variables que pueden ser utilizadas dentro de la plantilla para mostrar contenido dinámico.

Finalmente, en tu archivo index.ejs, puedes utilizar las etiquetas EJS para mostrar datos dinámicos y construir la página HTML generada:

Texto

Descripción generada automáticamente

Con esto, la página generada mostrará el título "EJS Demo" y el mensaje "¡Hola desde EJS!".

En resumen, EJS es un motor de plantillas que te permite generar vistas dinámicas en aplicaciones web mediante el uso de JavaScript embebido en HTML. Es una herramienta útil para crear contenido dinámico y reutilizable en tus aplicaciones web basadas en Node.js y Express.

¿Que son los Partials (parciales) EJS y para qué sirven?

Los Partials (parciales) en EJS son bloques de código reutilizables que te permiten definir fragmentos de una vista que pueden ser incluidos en múltiples lugares de la misma o en diferentes vistas. Son una forma de dividir una vista compleja en componentes más pequeños y manejables, lo que facilita la organización y mantenimiento del código.

¿Cómo funcionan los Partials en EJS?

En EJS, los Partials se crean utilizando la etiqueta especial <%- include() %>. La sintaxis para incluir un Partial en una vista es la siguiente:



Donde 'nombre-del-partial' es el nombre del archivo de la vista que deseas incluir. El archivo del Partial debe estar ubicado en la misma carpeta de vistas o en una subcarpeta de las vistas.

¿Para qué sirven los Partials en EJS?

Los Partials en EJS tienen varios usos y beneficios:

1. Reutilización de código: Los Partials permiten reutilizar bloques de código en diferentes vistas. Por ejemplo, si tienes un encabezado o un menú que aparece en múltiples páginas, puedes crear un Partial para ello y luego incluirlo en cada una de las vistas donde se necesita.
2. Mantenimiento y organización: Al dividir una vista compleja en Partials más pequeños, el código se vuelve más organizado y fácil de mantener. Si necesitas hacer cambios en un componente específico, solo tienes que modificar el Partial correspondiente, y el cambio se reflejará en todas las vistas que lo utilizan.
3. Legibilidad y claridad: Al utilizar Partials, el código de tus vistas puede ser más legible y claro, ya que cada componente tiene su propio archivo y se puede entender de manera aislada.

Ejemplo de uso de Partials en EJS:

Supongamos que tenemos una vista llamada index.ejs que necesita incluir un encabezado y un pie de página en todas sus páginas. Podemos crear dos Partials: header.ejs y footer.ejs.

header.ejs:

Texto

Descripción generada automáticamente

footer.ejs:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

index.ejs:

Texto

Descripción generada automáticamente

Con esta estructura, cada vez que se renderiza la vista index.ejs, el encabezado y el pie de página serán incluidos automáticamente gracias al uso de los Partials.

En resumen, los Partials en EJS son bloques de código reutilizables que te permiten dividir una vista en componentes más pequeños y manejables. Facilitan la reutilización de código, la organización y el mantenimiento de las vistas en aplicaciones web basadas en Node.js y Express.

¿Qué es un Content Delivery Network (CDN) y para que nos sirve?

Un Content Delivery Network (CDN), en español Red de Entrega de Contenidos, es una red de servidores distribuidos geográficamente que trabajan en conjunto para entregar contenido web de manera más rápida y eficiente a los usuarios finales. Su principal objetivo es reducir la latencia y mejorar la velocidad de carga de los sitios web y aplicaciones, brindando una mejor experiencia al usuario.

¿Cómo funciona un CDN?

Cuando una página web o aplicación utiliza un CDN, los activos estáticos como imágenes, hojas de estilo (CSS), archivos JavaScript y otros elementos multimedia son replicados y almacenados en varios servidores ubicados en diferentes puntos geográficos (llamados "nodos" o "puntos de presencia").

Cuando un usuario intenta acceder a un sitio web o aplicación, el CDN selecciona automáticamente el servidor más cercano a la ubicación del usuario (basado en la geolocalización) para servir el contenido. Esto reduce la distancia física que deben recorrer los datos desde el servidor hasta el usuario, lo que disminuye la latencia y mejora el tiempo de carga de la página.

Ventajas y beneficios del CDN:

1. Mejora en la velocidad de carga: Al reducir la distancia física entre el usuario y el servidor, los tiempos de respuesta y carga de la página se acortan considerablemente. Esto resulta en una experiencia más rápida y fluida para los usuarios.
2. Menor latencia: La latencia, o el tiempo que toma que un paquete de datos viaje de un punto a otro, se reduce significativamente cuando se utiliza un CDN. Esto es especialmente beneficioso para usuarios en ubicaciones geográficas lejanas al servidor principal del sitio.
3. Ahorro de ancho de banda: Al descargar y almacenar en caché los activos estáticos en los nodos del CDN, se reduce la carga en el servidor principal, lo que resulta en un ahorro de ancho de banda y una mejora en el rendimiento general del sitio.
4. Mayor disponibilidad y resistencia: Los CDN suelen ofrecer redundancia en sus servidores, lo que significa que, si uno de los servidores falla o experimenta problemas, otros servidores pueden tomar su lugar, manteniendo el contenido disponible y accesible para los usuarios.
5. Protección contra ataques DDoS: Los CDN a menudo tienen medidas de seguridad integradas para mitigar los ataques de denegación de servicio distribuido (DDoS), ayudando a proteger el sitio web o aplicación contra este tipo de amenazas.
6. Globalización del contenido: Si tienes un sitio web con audiencia internacional, un CDN te permite servir contenido de manera eficiente en diferentes regiones, asegurando una experiencia uniforme para los usuarios de todo el mundo.

En resumen, un Content Delivery Network (CDN) es una red de servidores que distribuye y entrega contenido web de manera más rápida y eficiente a los usuarios finales. Al utilizar un CDN, se mejora la velocidad de carga, se reduce la latencia y se optimiza el rendimiento general del sitio web o aplicación, proporcionando una experiencia más rápida y confiable para los usuarios.