3.1. Toma el proyecto del ejercicio 2.3 del tema anterior y desarrolla una clase de tipo @Controller que contenga diferentes @GetMapping con las rutas quieras que devuelvan las vistas solicitadas (index, palmares, galería-fotos, enlaces-externos).

Contesta en el PDF las siguientes cuestiones:

a) ¿Tienes que cambiar de ubicación las vistas? ¿Por qué?

Si, los archivos de vistas pasarán de static a template, ya que es la ruta que deben seguir para poder acceder desde controlador.

b) ¿Tienes que cambiar el código HTML del menú de navegación de las páginas?

En este caso deberemos eliminar los puntos de href y cambiarlos por th:href, además deberemos introducir el nombre de la variable que tenemos en el controlador, no la vista directamente.

c) ¿Tienen que llamarse igual las rutas del GetMapping y las vistas?

No es necesario las rutas nos indican que ruta deberemos introducir en el path de navegación y las vistas nos indicara que template vamos a recibir como respuesta a ese path.

La página index será servida para las URL: /index, /home, o simplemente /. Ya que las rutas y las vistas no tienen por qué llamarse igual, renombra las vistas con el sufijo “view”: indexView.html, palmaresView.html, photogalleryView.html, linksView.html, etc. Así podemos distinguir bien por el propio nombre lo que es una vista y lo que es una ruta o URL gestionada por el controlador.

Recuerda añadir en el application.properties la propiedad: spring.thymeleaf.cache=false y recuerda también de que la etiqueta <html> lleva un atributo xmlns.

Utiliza th:href y th:src en lugar de los atributos HTML href y src respectivamente.

Captura de pantalla de computadora

Descripción generada automáticamenteElimina el mapping para los enlaces-externos y haz que muestre la vista linksView.html mediante un archivo de configuración. Debes crear una clase que implemente WebMvcConfigurer, puedes llamarle como quieras: WebMvcConfigurerImpl, WebMvcConfig, etc.

3.2. Añade al proyecto anterior contenido dinámico pasándole información a las plantillas mediante un model y representándolo con etiquetas Thymeleaf. La página de inicio puede tener el año actual, por ejemplo ©2024 tomado de la fecha del sistema del servidor. Para ello puedes usar el método estático LocalDate.now().

La página de palmarés puede recibir la lista con los nombres de los títulos obtenidos por el equipo

(por ahora será un ArrayList de String en el controlador, pero más adelante debería tomar los datos

Interfaz de usuario gráfica

Descripción generada automáticamentedesde una base de datos).

3.3. Haz una copia del proyecto anterior y realiza los siguientes cambios:

• Si en la página de inicio, en la URL, se le pasa el parámetro: ?usuario=XXX mostrará el mensaje de

bienvenida con un texto personalizado para ese usuario, pero si no le pasa nada, será un mensaje

genérico (Bienvenido XXX a nuestra web vs. Bienvenido a nuestra web). Hazlo primero sin Optional,

luego ponla entre comentarios y haz una segunda versión con Optional.

• Añade BootStrap en su versión agnóstica (esto es, se define la versión empleada en el pom.xml

mediante webjars-locator).

• Utiliza fragmentos para no tener duplicado el código html tanto del <head> como del menú. El

menú podría utilizar la etiqueta <nav> y ser algo así:

<nav th:fragment="menu" class="navbar navbar-expand-sm">

<ul class="navbar-nav">

<li><a class="nav-link active" th:href="@{/}">Home</a></li>

<li><a class="nav-link active" th:href="@{/palmares}">Palmarés</a></li>

<li><a class="nav-link active" th:href="@{/galeria-fotos}">Galería de fotos</a></li>

<li><a class="nav-link active" th:href="@{/enlaces}">Enlaces externos</a></li>

</ul>

</nav>

Es buena práctica que las clases estén agrupadas en carpetas (paquetes) por lo que podrías hacer una carpeta ‘controllers’ para los controladores y otra ‘config’ para la clase de configuración creada

Interfaz de usuario gráfica

Descripción generada automáticamentepreviamente. A medida que avance el curso tendremos nuevas carpetas: services, repositories, utils, security, etc.

3.4. Implementa el ejemplo de los apuntes que genera números aleatorios en un nuevo proyecto.

Simplemente debes crear el controlador y la plantilla con el código mostrado.

Una vez que funcione correctamente, añade estilos CSS dinámicos. Los cambios a realizar serán los siguientes:

- Si la lista de números está vacía no se mostrará la tabla.

- Los números menores de 50 se mostrarán con color de letra verde oscuro sobre fondo verde claro.

- Los números mayores o iguales a 50 con color de letra rojo y fondo rosa.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteDeberás crear las clases CSS que contengan los atributos de los dos puntos anteriores.

3.5. Realiza una aplicación con la apariencia que se muestra en la figura siguiente, de forma que

presente tres imágenes a las que los visitantes pueden votar. Al clicar sobre cada una de las imágenes aumentará la cantidad de votos de esa imagen que se muestra debajo de ella. El contador de votos podrías mantenerlo en tres variables distintas, una para cada imagen, pero piensa que en un futuro añadamos más imágenes.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

3.6. Realiza una aplicación que implemente una calculadora como la mostrada en la figura siguiente.

El usuario iniciará introduciendo los dígitos del primer operando clicando en la botonera. Al pulsar en el botón ‘+’ pasará al segundo operando (si se pulsa ese botón en otra situación no hará nada). Luego introducirá los dígitos del segundo operando y finalmente pulsará el botón ‘=’ para mostrar el resultado (si pulsa el botón’=’ en otra situación, tampoco hará nada). El botón ‘clear’ vuelve a la situación inicial.

Te hará falta una variable de tipo enumeración para saber en qué estado estás: si introduciendo el primero operando, el segundo o acabas de mostrar el resultado.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteSi la vista se llama indexView.html y se llega a ella desde el mapping raíz: @GetMapping(“/”), el resto de mappings (añadir dígito, pulsar en el ‘+’, etc.) lo más normal es que los métodos de esos mappings terminen con: return “redirect:/” para volver a la vista inicial con sus datos en el Model.