CURSO DE PROGRAMACIÓN FULL STACK

PROGRAMACIÓN WEB CON JAVA

Guía de Programación Web

Fundamentos Web

El éxito de la web se basa en dos factores fundamentales: el protocolo HTTP y el lenguaje HTML. El primero permite una implementación sencilla de un sistema de comunicaciones que permite enviar cualquier archivo de forma fácil, simplificando el funcionamiento del servidor y posibilitando que servidores poco potentes atiendan cientos o miles de peticiones y reduzcan de este modo los costes de despliegue. El segundo, el lenguaje HTML, proporciona un mecanismo sencillo y muy eficiente de creación de páginas enlazadas.

El protocolo HTTP

El protocolo HTTP (Hypertext Tranfer Protocol) es el protocolo principal de la World Wide Web. Es un protocolo simple, orientado a conexión y sin estado. Está orientado a conexión proque emplea para su funcionamiento un protocolo de comunicaciones (TCP, o Transport Control Protocol) de modo conectado, que establece un canal de comunicaciones entre el cliente y el servidor, por el cual pasan los bytes que constituyen los datos de la transferencia, en contraposición a los protocolos denominados de datagrama (o no orientados a conexión) que dividen la serie de datos en pequeños paquetes (o datagramas) antes de enviarlos, pudiendo llegar por diversas vías del servidor al cliente.

Existe una variante de HTTP denominada HTTPS (S significa "secure", o "seguro") que utiliza el protocolo de seguridad SSL (o "Secure Socket Layer") para cifrar y autenticar el tráfico de datos, muy utilizada por los servidores web orientados al comercio electrónico o por aquellos que albergan información de tipo personal o confidencial. De forma esquemática, el funcionamiento de HTTP es como sigue: el cliente establece una conexión TCP con el servidor, hacia el puerto por defecto para el protocolo HTTP (o el indicado expresamente en la conexión), envía una orden HTTP de solicitud de un recurso (añadiendo algunas cabeceras con información) y, utilizando la misma conexión, el servidor responde enviando los datos solicitados y, además, añadiendo algunas cabeceras con información.

Métodos de Petición

GET – Utilizado para obtener un recurso del servidor, identificado por una url. Puede utilizarse para enviar parámetros y su longitud es limitada.

POST – Utilizado para enviar datos de entrada al servidor, ya que no tiene limitaciones de longitud. Puede utilizarse para enviar parámetros y su longitud es ilimitada.

UPDATE – Utilizado para actualizar un recurso.

DELETE – Utilizado para eliminar un recurso.

Códigos de Respuesta

200 – OK, Petición procesada correctamente.

301 – Indica al browser que visite otra dirección.

403 – Acceso prohibido, por falta de permisos.

404 – No encontrado, cuando el documento no existe.

500 – Error interno en el servidor.

HTML

El lenguaje HTML es el utilizado para construir páginas web y tiene la característica principal de ser un lenguaje declarativo, es decir, que sirve para declarar qué elementos deben componer la página y qué atributos deben tener, pero no especifica la manera de obtener el resultado. Esto implica que la implementación que finalmente muestra una página web en una PC, depende del browser que se utilice. Por ello, la misma página que aparece correctamente en un browser, puede tener diferencias o errores si se la carga en otro browser.

El HTML es un lenguaje de marcado (markup) que utiliza tags para identificar los elementos que componen a una página. La sintaxis sigue las líneas generales del XML, un lenguaje multipropósito de marcado. Algunos de los tags más comunes en HTML son: <html>, <body>, <table>, <tr>, <td>, <form>, <input> y .

HTML5

HTML5 (HyperText Markup Language, versión 5) es la quinta revisión del lenguaje HTML. Define los nuevos estándares de desarrollo web, rediseñando el código para resolver problemas y actualizándolo así a nuevas necesidades. No se limita solo a crear nuevas etiquetas o atributos, sino que incorpora muchas características nuevas y proporciona una plataforma de desarrollo (Ver Apéndice E).

Formulario HTML

Los formularios HTML son una parte fundamental de las aplicaciones web, dado que constituyen la forma de solicitar datos al usuario de la aplicación. Mediante los formularios se puede enviar al servidor tanto cadenas de texto como archivos de cualquier tipo.

Un formulario web está compuesto por campos de entrada (inputs) de distintos tipos, tales como texto, contraseñas, listas de selección, botones, etc. Dos tipos especiales sirven al funcionamiento del formulario: submit y reset.

Un input de tipo submit aparece generalmente como un botón y cumple la función de enviar el formulario al servidor.

Un input de tipo reset aparece también como un botón y cumple la función de borrar los valores de todos los campos del formulario.

Un formulario se envía al servidor mediante una petición con método POST, dado que debe poder enviarse una cantidad ilimitada de datos y archivos adjuntos.

CSS

Es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos definidos con HTML y XHTML. Junto con HTML y JavaScript, CSS (del inglés Cascading Style Sheets) es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas, interfaces de usuario para aplicaciones web y GUIs para muchas aplicaciones móviles.

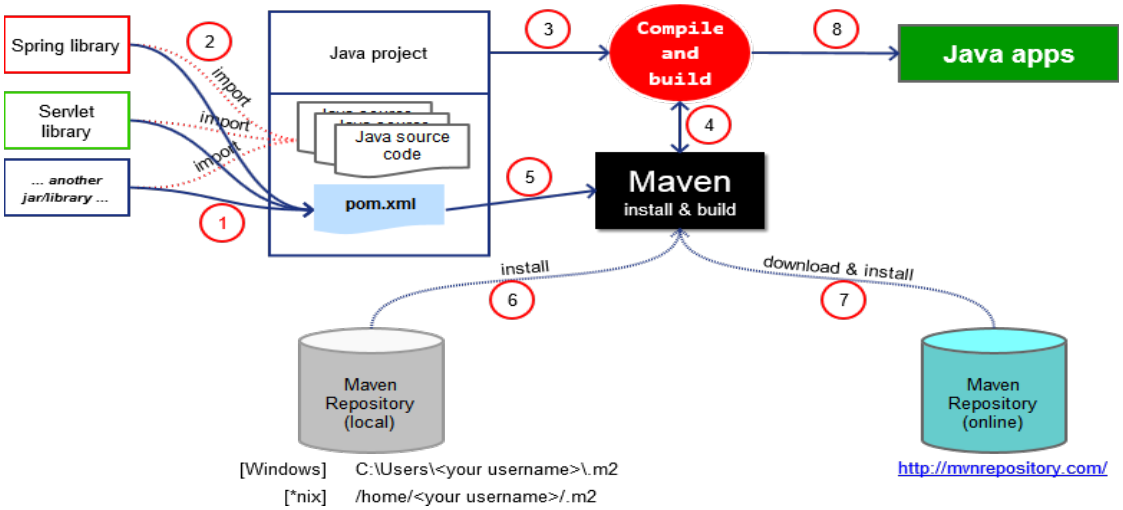
CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este, características tales como las capas o layouts, los colores y las fuentes. Esta separación busca mejorar la accesibilidad del documento, proveer más flexibilidad y control en la especificación de características presentacionales, permitir que varios documentos HTML compartan un mismo estilo usando una sola hoja de estilos separada en un archivo .css, y reducir la complejidad y la repetición de código en la estructura del documento.

Cookies

Las cookies son variables que son pasadas desde el servidor al cliente en una respuesta. El cliente debe almacenarlas por un periodo de tiempo, durante el cual debe enviar la cookie en cada petición que realiza al servidor. De esta manera, el cliente tiene una forma de identificarse ante el servidor como un visitante que regresa.

Maven

Maven es una herramienta de software para la gestión y construcción de proyectos Java. Utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. El modelo de configuración es simple y está basado en un formato XML (pom.xml). Además, Maven tiene objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado. La siguiente figura ilustra los pasos que lleva a cabo esta herramienta desde la importación de librerías hasta la generación de la aplicación Java.



Spring Framework

Spring es un framework alternativo al stack de tecnologías estándar en aplicaciones JavaEE. Spring popularizó ideas como la inyección de dependencias o el uso de objetos convencionales (POJOs) como objetos de negocio.

Spring es el framework más popular para el desarrollo de aplicaciones empresariales en Java, para crear código de alto rendimiento, liviano y reutilizable. Su finalidad es estandarizar, agilizar, manejar y resolver los problemas que puedan ir surgiendo en el trayecto de la programación.

Spring, ofrece como elemento clave el soporte de infraestructura a nivel de aplicación, brindando un completo modelo tanto para la configuración como para la programación de aplicaciones empresariales desarrolladas bajo Java, sin discriminación en cuanto al despliegue de la plataforma.

Spring MVC

Spring Web MVC es un sub-proyecto Spring que está dirigido a facilitar y optimizar el proceso creación de aplicaciones web utilizando el patrón Modelo Vista Controlador.

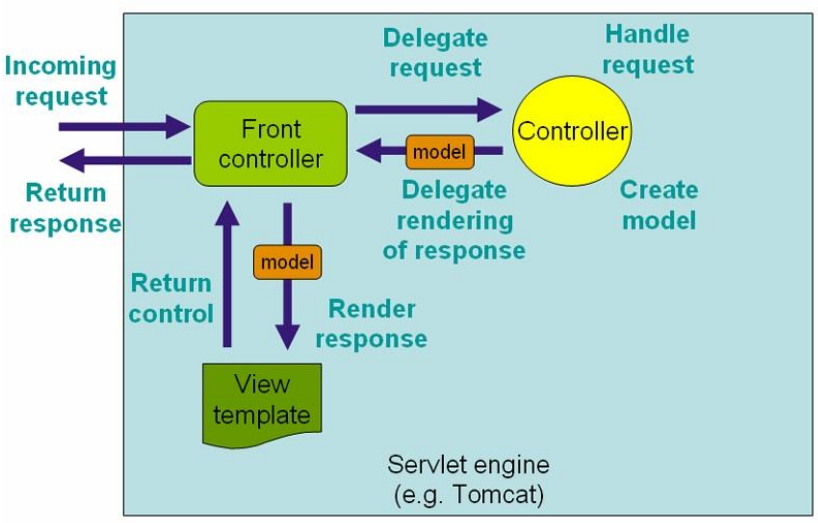
Modelo representa los datos o información que maneja la aplicación web.

Vista son todos los elementos de la Interfaz de Usuario.

Controlador será el encargado manipular los datos en base a la interacción del usuario.

Procesamiento de una Petición en Spring MVC

A continuación, se describe el flujo de procesamiento típico para una petición HTTP en Spring MVC. Spring es una implementación del patrón de diseño "front controller".



- Todas las peticiones HTTP se canalizan a través del *front controller*. En casi todos los frameworks MVC que siguen este patrón, el *front controller* no es más que un servlet cuya implementación es propia del framework. En el caso de Spring, la clase `DispatcherServlet`.

- El *front controller* averigua, normalmente a partir de la URL, a qué `Controller` hay que llamar para servir la petición. Para esto se usa un `HandlerMapping`.

- Se llama al `Controller`, que ejecuta la lógica de negocio, obtiene los resultados y los devuelve al servlet, encapsulados en un objeto del tipo `Model`. Además se devolverá el nombre lógico de la vista a mostrar (normalmente devolviendo un String, como en JSF).

- Un `ViewResolver` se encarga de averiguar el nombre físico de la vista que se corresponde con el nombre lógico del paso anterior.

- Finalmente, el *front controller* (el `DispatcherServlet`) redirige la petición hacia la vista, que muestra los resultados de la operación realizada.

Spring Boot

Es una herramienta que nace con la finalidad de simplificar aún más el desarrollo de aplicaciones basadas en el ya popular framework Spring. Spring Boot busca que el desarrollador solo si centre en el desarrollo de la solución, olvidándose por completo de la compleja configuración que actualmente tiene Spring Core para poder funcionar.

Spring Boot centra su éxito en las siguientes características que lo hacen extremadamente fácil de utilizar:

- *Configuración:* cuenta con un complejo módulo que autoconfigura todos los aspectos de nuestra aplicación para poder simplemente ejecutar la aplicación, sin tener que definir absolutamente nada.
- *Resolución de dependencias:* Con Spring Boot solo hay que determinar qué tipo de proyecto estaremos utilizando y él se encarga de resolver todas las librerías/dependencias para que la aplicación funcione.
- *Despliegue:* Spring Boot se puede ejecutar como una aplicación Stand-alone, pero también es posible ejecutar aplicaciones web, ya que es posible desplegar las aplicaciones mediante un servidor web integrado, como es el caso de Tomcat.
- *Métricas:* Por defecto, Spring Boot cuenta con servicios que permite consultar el estado de salud de la aplicación, permitiendo saber si la aplicación está prendida o apagada, memoria utilizada y disponible, etc.

Programación en Capas

La programación por capas es un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño.

La programación por capas es una técnica de ingeniería de software propia de la programación por objetos, éstos se organizan principalmente en las siguientes capas:

Capa de Interfaz
Esta capa resuelve la presentación de datos al usuario. Esta capa se encarga de "dibujar" las pantallas de la aplicación al usuario, y tomar los eventos que el cliente genere (por ejemplo, el hacer click en un botón).

Capa de Comunicación
En esta capa están los controladores y es capa responsable de mediar entre la interfaz de usuario y las capas inferiores. En esta capa contiene el dispatcher encargado de enrutar las peticiones así como los controladores de acceso a los servicios web.

Capa de Servicios
Esta capa resuelve la lógica de la aplicación. Contiene los algoritmos, validaciones y coordinación necesaria para resolver la problemática. Los elementos fundamentales de esta capa son los objetos de dominio. Estos objetos representan los objetos principales del negocio. La lógica para manipular los objetos que representan los datos se encuentra en los llamados objetos de negocio (Service Object).

Capa de Acceso a Datos
Esta capa resuelve el acceso a datos, abstrayendo a su capa superior de la complejidad del acceso e interacción con los diferentes orígenes de datos. Esta capa se encarga de proveer un API simple de usar, orientado al negocio, sin exponer complejidades propias de un repositorio de datos.

En esta capa se resuelven:

- cualquier acceso a la base de datos
- cualquier acceso a filesystem
- cualquier acceso a otros sistemas
- cualquier acceso a un repositorio de datos en cualquier forma.

Thymeleaf

Es una biblioteca Java que implementa un motor de plantillas de XML/XHTML/HTML5 (también extensible a otros formatos) que puede ser utilizado tanto en modo web como en otros entornos no web. Se acopla muy bien para trabajar en la capa vista del MVC de aplicaciones web, pero puede procesar cualquier archivo XML, incluso en entornos desconectados.

Thymeleaf proporciona un módulo opcional para la integración con Spring MVC. El objetivo principal de es permitir la creación de plantillas de una manera elegante y un código bien formateado. Sus dialectos Standard y SpringStandard permiten crear potentes plantillas naturales que se pueden visualizar correctamente en los navegadores de Internet, por lo que también funcionan como prototipos estáticos. Thymeleaf también puede extenderse desarrollando tus propios dialectos.

Preguntas de Aprendizaje

1) Maven:
   a) Es una herramienta para formatear código
   b) Es una herramienta para automatizar tareas
   c) Es un IDE para construir aplicaciones web
   d) Ninguna de las anteriores

2) Spring Framework es:
   a) Un framework para el desarrollo de aplicaciones PHP
   b) Un framework para el desarrollo de aplicaciones .net
   c) Un framework para el desarrollo de aplicaciones Java
   d) Todas las anteriores

3) Un esquema:
   a) Determina en qué orden deben aparecer los elementos
   b) Determina qué atributos pueden tener
   c) Identifica los objetos que pueden aparecer en un documento XML
   d) Todas las anteriores

4) ¿Cómo se puede inyectar una Collection Java en Spring?
   a) Usando los tags list, set, map ó props.
   b) Usando los tags lit, set, map ó collection.
   c) Usando los tags list, set, props ó collection.
   d) Usando los tags list, collection, map ó props.

5) En HTML, para referenciar una hoja de estilo externa se emplea la etiqueta <link> con el atributo
   a) href
   b) src
   c) url
   d) Ninguno de los anteriores

6) En HTML, para indicar el orden de tabulación entre los controles de un formulario se emplea el atributo
   a) alt
   b) index
   c) tab
   d) tabindex

7) ¿Cuál de los siguientes Doctype es el que utilizarías para un documento HTML5?
   a) <!doctype html5>
   b) <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 5.0 Transitional//EN">
   c) <!doctype html>
   d) Ninguno de los anteriores

8) Selecciona el elemento en el cuál incluirías los links de la navegación primaria de un sitio:
   a) <section>
   b) <menu>
   c) <header>
   d) <nav>

9) Selecciona el elemento inválido:
   a) <meter>
   b) <hgroup>
   c) <progress>
   d) <post>

10) Selecciona el tipo de input válido
   a) type="boolean"
   b) type="textarea"
   c) type="alphanumeric"
   d) type="range"

11) Si tuvieras que crear un campo de búsqueda, ¿qué sería lo más apropiado?
   a) <input type="find" />
   b) <input type="text" search />
   c) <input type="search" />
   d) Ninguno de los anteriores

12) Si estuvieras maquetando un blog, en cuál de los siguientes elementos pondrías los links a archivo, categorías, íconos de redes sociales, links a artículos más populares, etc
   a) <aside>
   b) <section>
   c) <summary>
   d) Ninguno de los anteriores

13) Para que un input tenga el foco apenas se termine de cargar el documento, ¿cuál de las siguientes soluciones aplicarías?
   a) Utilizar el atributo booleano autofocus
   b) Utilizar el atributo autofocus="true"
   c) Utilizar el atributo placeholder
   d) Ninguno de los anteriores

14) ¿Para qué se utiliza el contenido dentro de la etiqueta <canvas>?
   a) Se muestra como link, que al hacer click en él nos muestra el canvas.
   b) Es el contenido alternativo que muestran solo los navegadores que no soportan canvas.
   c) Es el texto que le podemos añadir al canvas a modo de título.
   d) Ninguno de los anteriores

15) ¿Cuál es el lugar correcto en un documento HTML para hacer referencia a una hoja de estilo externa?
   a) Al principio del documento
   b) En la sección <head>
   c) En la sección <body>
   d) Al final del documento

16) ¿Cómo se hace en CSS para que el texto esté centrado?
   a) center: true
   b) text-center: true
   c) align: center
   d) text-align: center

17) ¿Qué atributo de HTML se emplea para definir los estilos en línea?
   a) class
   b) css
   c) font
   d) style

18) ¿Qué significa CSS?
   a) Cascading Style Sheets
   b) Creative Style Sheets
   c) Computer Style Sheets
   d) Colorful Style Sheets

19) ¿Qué etiqueta de HTML se emplea para definir una hoja de estilo interna?
   a) <css>
   b) <link>
   c) <script>
   d) <style>

20) ¿Cuál es la propiedad de CSS que permite cambiar el tamaño del texto?
   a) text-size
   b) text-style
   c) font-size
   d) font-style

21) ¿Qué propiedad no existe en CSS?
   a) border-color
   b) border-line
   c) border-style
   d) border-width

22) ¿Cuál es la sintaxis correcta de CSS para que todos los elementos <p> aparezcan en negrita?
   a) p {text-decoration:bold}
   b) p {text-size:bold}
   c) p {text-style:bold}
   d) p {font-weight:bold}

23) ¿Cómo se puede cambiar el margen izquierdo de un elemento con CSS?
   a) text-indent
   b) indent
   c) marginleft
   d) margin-left

24) En CSS, para ocultar un elemento y que no se visualice se emplea
   a) display: hidden;
   b) display: invisible;
   c) display: never;
   d) Las anteriores respuestas no son correctas

25) ¿Cómo se cambia el color del texto de un elemento en CSS?
   a) textcolor
   b) text-color
   c) color
   d) fgcolor

Ejercicios de Aprendizaje

Para la realización de este trabajo práctico se recomienda consultar el "Apéndice E" para obtener más información acerca de las anotaciones del framework Spring Boot, Thymeleaf, HTML5 y CSS. Se recomienda ver todos los videos e ir haciendo lo que vayan necesitando.

VIDEOS:

A) Fundamentos Web
B) Configuración Spring

1.

Sistema de Reservas: Librería Web

El objetivo de este ejercicio consiste en el desarrollo de un sistema web de reserva de libros en JAVA utilizando una base de datos MySQL, JPA como framework de persistencia y Spring Boot como framework de desarrollo web.

Creación de la Base de Datos MySQL

Crear el esquema sobre el cual operará el sistema de reservas de libros. Para esto, en el IDE de base de datos que esté utilizando (por ejemplo, Workbench) ejecute la sentencia:

CREATE DATABASE libreria;

De esta manera se creará una base de datos vacía llamada librería.

Paquetes del Proyecto

Los paquetes que se deben utilizar para el proyecto se deben estructurar de la siguiente manera:

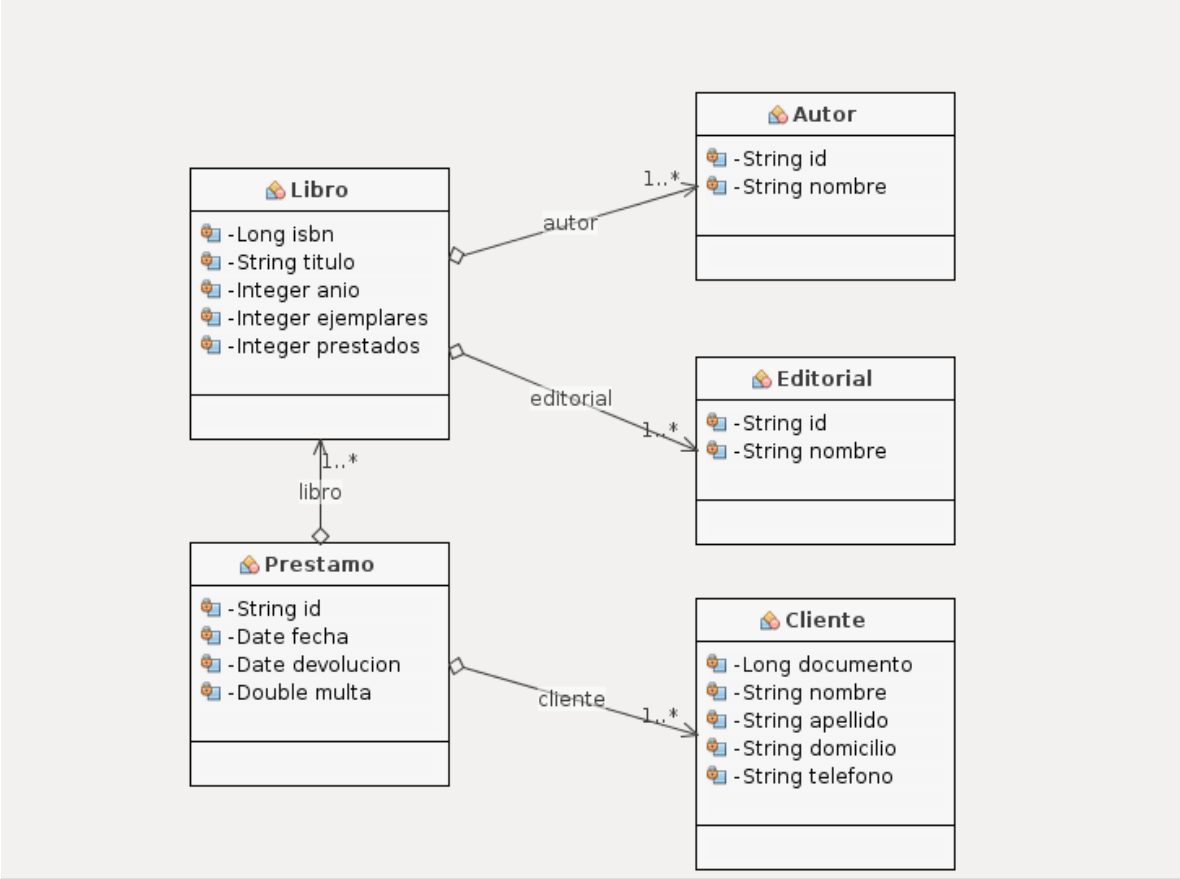
- interfaz: en este paquete se almacenarán aquellas clases que se utilizarán como interfaz con el usuario.
- controladores: en este paquete se almacenarán aquellas clases que se utilizarán para mediar entre la interfaz con el usuario y las capas inferiores.
- servicios: en este paquete se almacenarán aquellas clases que llevarán adelante lógica del negocio.
- repositorios: en este paquete se crearán los repositorios que servirán como interfaces entre el modelo de objetos y la base de datos relacional.
- entidades: en este paquete se almacenarán aquellas clases que es necesario persistir en la base de datos.

VER VIDEO: Capa de acceso a datos

Capa de Datos

Entidades y Repositorios

Crear el siguiente modelo de entidades y los repositorios correspondientes para este modelo:



Spring utiliza una anotación para identificar aquellas clases que serán entidades y repositorios. Todas las entidades deben estar marcadas con la anotación @Entity y los repositorios con la anotación @Repository.

Entidad Libro

La entidad libro modela los libros que están disponibles en la biblioteca para ser prestados. En esta entidad, el atributo “ejemplares” contiene la cantidad total de ejemplares de ese libro, mientras que el atributo “prestados” contiene cuántos de esos ejemplares se encuentran prestados en este momento. El repositorio que persiste a esta entidad (LibroRepositorio) debe contener los métodos necesarios para guardar/actualizar libros en la base de datos, realizar consultas y eliminar o dar de baja según corresponda.

Entidad Cliente

La entidad cliente modela los clientes (a quienes se les presta libros) de la biblioteca. Se almacenan los datos personales y de contacto de ese cliente. El repositorio que persiste a esta entidad (ClienteRepositorio) debe contener los métodos necesarios para guardar/actualizar un cliente en la base de datos, realizar consultas y eliminar o dar de baja según corresponda.

Entidad Préstamo

La entidad préstamo modela los datos de un préstamo de libros. Esta entidad registra la fecha en la que se efectuó el préstamo y la fecha en la que se devolvieron los libros, y un campo multa que debe pagar el cliente cuando devuelve un libro luego de su fecha de devolución. Esta entidad también registra los libros que se llevaron en dicho préstamo y quien fue el cliente al cual se le prestaron. El repositorio que persiste a esta entidad (PrestamoRepositorio) debe contener los métodos necesarios para registrar un préstamo en la base de datos, realizar consultas y realizar devoluciones, etc.

Entidad Autor

La entidad autor modela los autores de libros. El repositorio que persiste a esta entidad debe contener todos los métodos necesarios para guardar en la base de datos, realizar consultas y eliminar o dar de baja según corresponda. El repositorio que persiste a esta entidad (AutorRepositorio) debe contener los métodos necesarios para guardar/actualizar un cliente en la base de datos, realizar consultas y eliminar o dar de baja según corresponda.

Entidad Editorial

La entidad editorial modela las editoriales que publican libros. El repositorio que persiste a esta entidad (EditorialRepositorio) debe contener los métodos necesarios para guardar/actualizar una editorial en la base de datos, realizar consultas y eliminar o dar de baja según corresponda.

VIDEOS:

A) Capa de Servicios
B) Modelo Vista Controlador

Capa de Servicios

Spring utiliza una anotación para identificar aquellas clases que serán servicios. Todos los servicios deben estar marcados con la anotación @Service.

LibroServicio

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para administrar libros (consulta, creación, modificación y eliminación).

ClienteServicio

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para administrar clientes (consulta, creación, modificación y eliminación).

PrestamoServicio

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para administrar préstamos (consulta, préstamo, modificación y eliminación).

AutorServicio

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para administrar autores (consulta, creación, modificación y eliminación).

EditorialServicio

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para administrar editoriales (consulta, creación, modificación y eliminación).

Capa de Comunicación

Spring utiliza una anotación para identificar aquellas clases que serán controladores. Todos los controladores deben estar marcadas con la anotación @ Controller. Algunos de los controladores a desarrollar son los siguientes.

LibroController

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para operar con la interfaz del usuario diseñada para la gestión de libros (guardar/modificar libro, listar libros, eliminación).

ClienteController

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para operar con la interfaz que gestiona clientes (guardar/modificar, listar clientes, eliminación).

PrestamoController

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para operar con la interfaz o portal para gestionar préstamos (guardar/modificar, listar préstamos realizados, devolución, eliminación).

AutorController

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para operar con la interfaz que gestiona los autores (guardar/actualizar, listar autores, eliminación).

EditorialController

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para operar con la interfaz que gestiona editoriales (guardar/modificar, listar editoriales, eliminación).

Capa de Interfaz

Esta capa tiene la responsabilidad de llevar adelante las funcionalidades necesarias para interactuar con el usuario. Las interfaces para este proyecto tienen que estar desarrolladas en HTML5 y se debe utilizar la biblioteca Thymeleaf y CSS para implementar las plantillas. Además, se debe utilizar el framework de Bootstrap para los componentes.

Se deben diseñar y crear todas las interfaces web necesarias para llevar a cabo las siguientes funcionalidades:

- Administrar Autores: cargar datos de un autor, modificar datos, listar autores, eliminar.
- Administrar Editoriales: cargar los datos de una editorial, modificar los datos, listar editoriales.
- Administrar Libros: cargar datos de un nuevo libro, modificar datos, listar libros, eliminar.
- Administrar clientes: cargar datos de los clientes que desean pedir prestado un libro, modificar datos de los clientes, realizar listados y eliminar clientes.
- Realizar Préstamos: cargar los datos de un préstamo. Tener en cuenta que para realizar un préstamo se deben incluir los libros a prestar y el cliente asociado. Modificar un préstamo (por ejemplo, renovar la fecha de devolución), listar préstamos realizados, etc.

A continuación, se muestran algunos ejemplos para el módulo de Administración de Autores.

a) Listar Autores

Administración de Autores

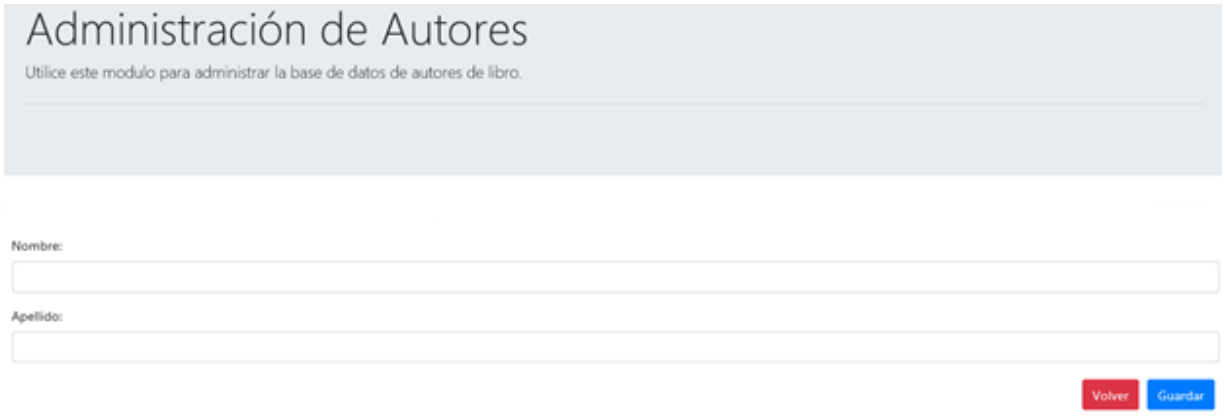
Libreria > Autores > Listado

Buscar

Codigo	Nombre	Apellido	Acciones
0001	Sebastian	Arbona	Modificar - Eliminar
0002	Sebastian	Perez	Modificar - Eliminar
0003	Sebastian	Gomez	Modificar - Eliminar
0004	Federico	Arbona	Modificar - Eliminar
0005	Juan	Arbona	Modificar - Eliminar
0006	Roberto	Arbona	Modificar - Eliminar

6 autores encontrados

b) Cargar Autor



Ejercicio Complementario

2. Sistema de Estancias en el Extranjero Web

El objetivo de este ejercicio consiste en el desarrollo de un sistema web para una pequeña empresa que se dedica a organizar estancias en el extranjero dentro de una familia. El sistema debe registrar la reserva de casas por parte de los clientes que desean realizar alguna estancia. El objetivo es el desarrollo de un sistema web de reservas de casas para realizar estancias en el exterior, utilizando el lenguaje JAVA, una base de datos MySQL, el framework de persistencia JPA y Spring Boot como framework de desarrollo web.

Creación de la Base de Datos MySQL

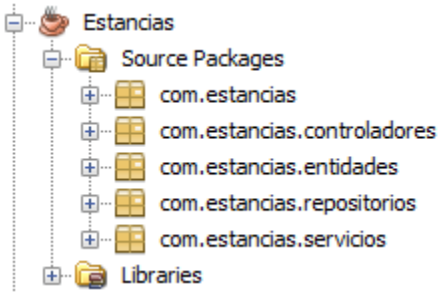
Crear el esquema sobre el cual operará el sistema de reservas de casas. Para esto, en el IDE de base de datos que esté utilizando (por ejemplo, Workbench) se debe ejecutar la sentencia:

```
CREATE  DATABASE estancias;
```

De esta manera se creará una base de datos vacía llamada estancias.

Paquetes del Proyecto

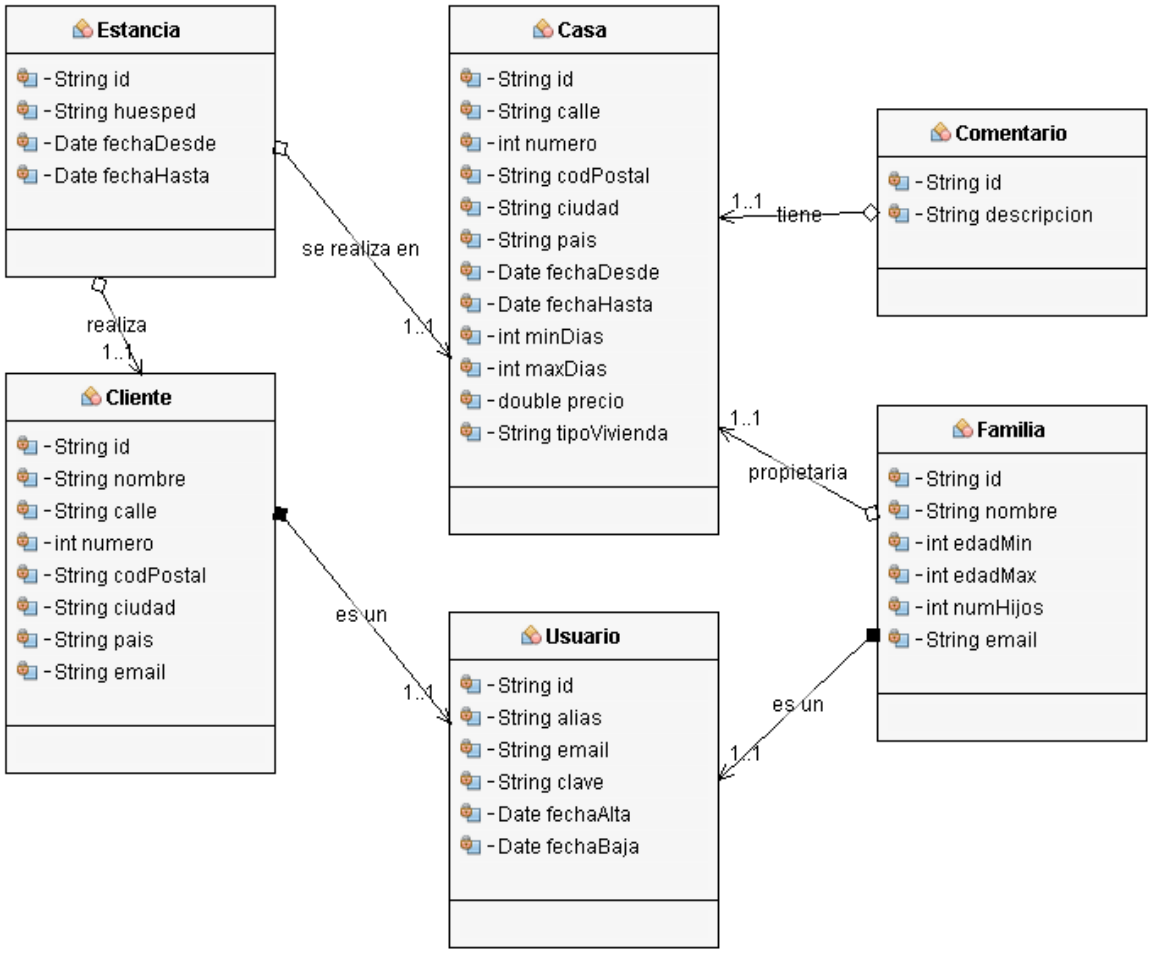
En este proyecto se debe utilizar la misma estructura de capas que en el ejercicio anterior.



Capa de Datos

Entidades y Repositorios

Crear el siguiente modelo de entidades y agregar los repositorios correspondientes. Todas las entidades deben estar marcadas con la anotación @Entity y los repositorios con la anotación @Repository.



Entidad Usuario

La entidad usuario modela los datos de un usuario que accede al sistema para registrarse como familia y ofrecer una habitación de su casa para estancias, o bien, como un cliente que necesita realizar una reserva. De cada usuario se debe registrar el nombre de usuario (alias), el correo electrónico, el password y la fecha de alta. El repositorio que persiste a esta entidad (UsuarioRepositorio) debe contener los métodos necesarios para registrar el usuario en la base de datos, realizar consultas y eliminar.

Entidad Familia

La entidad familia modela las familias que habitan en diferentes países y que ofrecen alguna de las habitaciones de su hogar para acoger a algún chico (por un módico precio). De cada una de estas familias se conoce el nombre, la edad mínima y máxima de sus hijos, número de hijos y correo electrónico. El repositorio que persiste a esta entidad (FamiliaRepositorio) debe contener los métodos necesarios para guardar/actualizar los datos de las familias en la base de datos, realizar consultas y eliminar o dar de baja según corresponda.

Entidad Casa

La entidad casa modela los datos de las casas donde las familias ofrecen alguna habitación. De cada una de las casas se almacena la dirección (calle, numero, código postal, ciudad y país), el periodo de disponibilidad de la casa (fecha_desde, fecha_hasta), la cantidad de días mínimo de estancia y la cantidad máxima de días, el precio de la habitación por día y el tipo de vivienda. El repositorio que persiste a esta entidad (CasaRepositorio) debe contener los métodos necesarios para guardar/actualizar los datos de una vivienda, realizar consultas y eliminar.

Entidad Cliente

La entidad cliente modela información de los clientes que desean mandar a sus hijos a alguna de las casas de las familias. Esta entidad es modelada por el nombre del cliente, dirección (calle, numero, código postal, ciudad y país) y su correo electrónico. El repositorio que persiste a esta entidad (ClienteRepositorio) debe contener los métodos necesarios para guardar/actualizar los datos de un cliente, realizar consultas y eliminar.

Entidad Reserva

La entidad reserva modela los datos de las reservas y estancias realizadas por alguno de los clientes. Cada estancia o reserva la realiza un cliente, y además, el cliente puede reservar varias habitaciones al mismo tiempo (por ejemplo para varios de sus hijos), para un periodo determinado (fecha_llegada, fecha_salida). El repositorio que persiste a esta entidad (ReservaRepositorio) debe contener los métodos necesarios para realizar una reserva, actualizar los datos (por ejemplo, fecha de la reserva), realizar consultas de las reservas realizadas para una determinada vivienda y eliminar reserva.

Entidad Comentario

La entidad comentario permite almacenar información brindada por los clientes sobre las casas en las que ya han estado. El repositorio que persiste a esta entidad (ComentarioRepositorio) debe contener los métodos necesarios para guardar los comentarios que realizan los clientes sobre una determinada una vivienda.

Capa de Servicios

Utiliza la anotación @Service para identificar aquellas clases que serán servicios. Todos los servicios deben estar marcados con esta anotación.

UsuarioServicio

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para administrar usuarios (alta de usuario, consultas, y baja o eliminación).

FamiliaServicio

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para administrar familias (creación, consulta, modificación y eliminación).

CasaServicio

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para administrar las casas (creación, consulta, modificación y eliminación).

ClienteServicio

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para administrar clientes (creación, consulta, modificación y eliminación).

ReservaServicio

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para realizar las reservas de viviendas (reservar, consultar reservas realizadas, modificación y eliminación).

Capa de Comunicación

Spring utiliza una anotación para identificar aquellas clases que serán controladores. Todos los controladores deben estar marcadas con la anotación @Controller. Algunos de los controladores a desarrollar son los siguientes.

UsuarioController

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para operar con la interfaz del usuario diseñada para la gestión de usuarios (dar de alta un usuario, cambiar clave, listar usuarios registrados, dar de baja un usuario).

FamiliaController

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para operar con la interfaz del usuario diseñada para la gestión de familias (guardar/modificar datos de la familia, listar familias, eliminar).

CasaController

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para operar con la interfaz del usuario diseñada para la gestión de viviendas (guardar/modificar datos de la casa, listar viviendas, eliminar).

ClienteController

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para operar con la interfaz del usuario diseñada para la gestión de clientes (guardar/modificar, listar clientes, eliminar).

ReservaController

Esta clase tiene la responsabilidad de llevar adelante las funcionalidades necesarias para operar con la interfaz o portal para gestionar reservas de estancias (guardar/modificar, listar estancias reservadas/realizadas, eliminación).

Capa de Interfaz

Esta capa tiene la responsabilidad de llevar adelante las funcionalidades necesarias para interactuar con el usuario. Las interfaces para este proyecto tienen que estar desarrolladas en HTML5 y se debe utilizar la biblioteca Thymeleaf y CSS para implementar las plantillas.

Se deben diseñar y crear todas las interfaces web necesarias para llevar a cabo las siguientes funcionalidades:

- Administrar usuarios: registrar nuevos usuarios en el sistema, cambiar clave, listar usuarios, dar de baja o eliminar.

- Administrar familias: cargar datos de una familia, modificar datos, consultar familias, eliminar familias que no ofrecen más sus viviendas para estancias. Las familias deben darse de alta una vez que hayan sido registradas como usuario.

- Administrar casas: cargar los datos de una vivienda y asociar a la familia correspondiente, modificar los datos (por ejemplo, fechas en las cuales se encuentra disponible), listar casas (país, el periodo de disponibilidad, cantidad de días mínima y máxima de estancia, el precio de la habitación por día, el tipo de vivienda y el nombre del propietario). Se debe eliminar los datos de una casa cada vez que se da de baja la familia propietaria.

- Administrar clientes: cargar datos de los clientes que desean reservar una habitación para realizar una estancia, modificar datos de los clientes, realizar consultas y eliminar clientes. Al igual que las familias, un cliente puede darse de alta una vez que se haya creado el usuario correspondiente.

- Realizar Reservas: El portal principal debería permitir que una persona que ingresa a la web pueda consultar las viviendas que se encuentran disponibles para realizar estancias (validando fechas disponibles). Una vez que el usuario encuentra una vivienda que se adecua a sus preferencias y quiere realizar una reserva, recién en ese momento se solicita que se registre como usuario y luego se procede a realizar la reserva.
    - Cuando el usuario se registra se le debe dar la opción de elegir si se quiere registrar como familia que ofrece una vivienda o como cliente. Dependiendo de la opción elegida se pide que complete los datos correspondientes (familia o cliente) para continuar con su registro.
    - Si el usuario ya se encuentra restirado entonces debe realizar el login para poder continuar.
    - Se debe tener en cuenta que para realizar una reserva la vivienda no debe estar ya reservada por otro cliente.
    - Se debe permitir que un cliente modifique su reserva, por ejemplo: cambiar las fechas, o la elimine en caso de no poder realizarla.
    - Se deben poder listar las reservas realizadas por parte de los clientes.