

Controlador y Servicio	
@Controller	Definición de controlador
@RequestMapping ("/rutaBase")	Ruta raíz de todos los mappings
@GetMapping ("/ruta") @PostMapping, @PutMapping, @DeleteMapping public String showPage (Model model){ model.addAttribute ("city", "Lugo"); return "vista"; }	Verbo HTTP al que responde el metodo Método que devuelve vista Pase de parámetros a la vista Vista html que muestra
@GetMapping ({"/ruta1", "/ruta2", "/ruta3"})	Responde a varias rutas
@GetMapping ("/ruta") public String showPage (@RequestParam String p, Model model){ @RequestParam (required=false, defaultValue="X") String p @RequestParam Optional <String> p	Recibe param query en URL: ?p=valor Evitar error si vacío Evitar error si vacío: p.orElse("X")
@GetMapping ("/ruta/{p}") public String showPage (@PathVariable String p, Model model){ return "redirect:/rutaCompleta";	Recibe param en path URL: ruta/p Redirige a otro controlador, no vista
@Service public class MiServicioClase implements MiServicioInterfaz { En el controlador: @Autowired MiServicioInterfaz miServicioInterfaz;	Definición de Clase de servicio Inyectamos la interfaz

Vistas + Thymeleaf	
<html xmlns:th="http://www.thymeleaf.org">	Etiqueta html para vistas con Thymeleaf
<link href="/webjars/bootstrap/css/bootstrap.min.css" rel="stylesheet"> <script src="/webjars/bootstrap/js/bootstrap.bundle.min.js"></script>	Incluir Bootstrap con webjars
<span th:text="{city}">*</span>	Variable con Thymeleaf
<span th:if="{res>0}">...</span> <span th:unless="{res>0}">...</span>	Condicional con Thymeleaf
<div th:each="nombre:{listaNombres}"> <p th:text="{nombre}">*</p> </div>	For...each con Thymeleaf (texto)
<div th:each="producto:{listaProductos}"> <p th:text="{producto.nombre}">*</p> </div>	For...each con Thymeleaf (objeto) Necesita getters
<head th:fragment="myHead">...</head> (en templates/fragment.html) <head th:replace="/fragment.html::myHead"></head>	Definición de fragmento con Thymeleaf Sustitución de etiqueta por fragmento
<a th:href="@{/ruta}">link</a> <a th:href="@{/ruta (param={variable} )}">link</a> <a th:href="@{/ruta/{param} (param={variable} )}">link</a>	Enlace a ruta (URL local) Enlace a ruta con variable en query Enlace a ruta con variable en path

Formularios	
form action="#" method="post" th:action="@{/myForm/submit}" th:object="{formInfo}">	Etiqueta form con destino Y objeto con datos
<input type="text" id="nombre" th:field="**{nombre}">	Atributo de tipo texto vinculado a objeto
@GetMapping ("/myForm") public String showForm (Model model) { model.addAttribute ("formInfo", new FormInfo()); return "formView"; }	Mapping presentación formulario
@PostMapping ("/myForm/submit") public String myformSubmit (FormInfo formInfo) { ... public String myformSubmit (@ModelAttribute FormInfo formInfo) { ...	Recepción formulario formInfo es el objeto con los datos recib. @ModelAttribute... los pasa a la vista directm.
public String myformSubm (@Valid FormInfo formInfo, BindingResult br) {if (br.hasErrors()) { ...	Validación de datos recibidos

Modelo	
@Getter, @Setter, @EqualsAndHashCode, @ToString = @Data	Anotaciones Lombok
@NoArgsConstructor, @AllArgsConstructor	Anotaciones Lombok
@Min(value=0), @NotEmpty, @Email,	Validaciones en clase formulario
@Configuration, @Getter, @Setter @PropertySource("classpath:/fich.properties") public class MiClase { @Value("\${iva}") private Double iva; }	Fichero de parámetros Ruta del archivo Clase y mapeo de cada campo
private List<MiClase> repositorio = new ArrayList<>();	Repositorio en Memoria en el servicio.

JPA	
@Entity	Entidad gestionada por JPA
@Id	Clave de una entidad
@GeneratedValue	@Id es autogenerado por la BD.
public interface ProductoRepository extends JpaRepository<T, ID> { List<T> findAll() Optional<T> findById (ID id) void delete (T t), void deleteById (ID id) T save (T)	Repositorio JPA Métodos de repositorio incluidos generados por defecto. Hay más...
List<Persona> findByEmail (String email);	Metódos de repo. derivados por nombre
@Query("select e from Empleado e where e.salario>=?1") List<Empleado> getEmpleadoSalarioAlto (Double salar);	Metodo de repo. con query.
@ManyToOne @OnDelete (action = OnDeleteAction.CASCADE) private Departamento departamento;	Relación muchos a uno (n empleados – 1 departamento)
@OneToMany(fetch=FetchType.EAGER, cascade=CascadeType.REMOVE) private List<Empleado> empleados = new ArrayList<>();	Relación uno a muchos (1 departamento – n empleados)
mappedBy="departamento"	Se añade a @OneToMany si bidirecc.
@ToString.Exclude	Se añade a la entidad @OneToMany
Entidad NM: {...@ManyToOne N + @ManyToOne M + AtribExtra} Entidad N: @OneToMany mappedBy="n" List<NM> nm = new... Entidad M: @OneToMany mappedBy="m" List<NM> nm= new ..	Relación muchos a muchos N-M con atributos extra.
@OneToOne (mappedBy="empleado")	Relación uno a uno (en cada una de las dos)
public List<Empleado> getEmpleadosPaginados(Integer pageNum) { Pageable paging = PageRequest.of(pageNum, 10, Sort.by("nombre").ascending()); Page<Empleado> pagedResult = empleadoRepository.findAll(paging); if (pagedResult.hasContent()) return pagedResult.getContent(); else return null; } pagedResult.getTotalPages()	Resultados paginados, de 10 en 10, ordenador por nombre.  Devuelve la página pasada como parámetro
@Inheritance(strategy = InheritanceType.SINGLE_TABLE) @DiscriminatorColumn(name = "tipoPaciente") @DiscriminatorValue(value = "1") @Inheritance(strategy = InheritanceType.JOINED)	Herencia single_table. (en superclase)  (en subclases) Herencia tipo Joined (en superclase)

pom.xml	
<artifactId> <name>	Nombre del proyecto
<dependencies>web, thymeleaf, devtools, test,	Dependencias básicas
<dependencies> webjars-bootstrap, webjars-locator	Dependencia para BootStrap con Maven
<dependencies> lombok, data-jpa, validation, h2, modelmapper	Dependencia para acceso a datos
<dependencies> springfox-boot-starter, springfox-swagger-ui	Incorporar Swagger
<dependencies> spring-boot-starter-webflux	WebClient
<dependencies> spring-boot-starter-security,	Control de acceso
<dependencies> thymeleaf-extras-sprigsecurity5	

API Rest	
@RestController @GetMapping, @DeleteMapping, @PostMapping, @PutMapping	Controlador API Rest Métodos CRUD
public ResponseEntity<?> metodo ( @Valid @RequestBody T t @PathVariable Long id) {...	Firma método controlador: con envío de datos (PUT,POST), e id.
ResponseEntity.ok(recurso) ResponseEntity.notFound().build(); ResponseEntity.status(HttpStatus.CREATED).body(recurso) ResponseEntity.noContent().build()	Respuesta 200 Respuesta 404 (no encontrado) Respuesta 201 (creado) Respuesta 204 (borrado)
HttpStatus: OK, CREATED, NO_CONTENT, NOT_FOUND, FORBIDDEN, BAD_REQUEST	Valores posibles HttpStatus
public ResponseEntity<?> listElements (){ public ResponseEntity<?> newElement (@RequestBody Entidad e){ @JsonIgnore	Petición sin cuerpo (p.ej. GET, DELETE) Petición con cuerpo (p.ej: POST,PUT)
public class NombreException extends RuntimeException { public NombreException (Long id) { super("mensaje: " + id); }}	Rel.bidirec (en clase 1 de 1 a n) Creación de excepción
if ( . . . ) throw new NombreException (id)	Lanzar excepción
findById(id).orElseThrow()->new NombreException(id))	Lanzar excepción desde JPA findById
try { /*llamada a metodo de servicio*/. } catch(NombreException ex) throw new ResponseStatusException(HttpStatus.XX, ex.getMessage());	Gestión de errores ResponseStatusEx. Captura excepción y devuelve un estado http y un mensaje.
@RestControllerAdvice @ExceptionHandler (NombreException.class)	Gestión de errores centralizada. Clase con métodos que devuelve ResponseEntity<?>

Seguridad MVC	
@Bean public SecurityFilterChain filterChain(HttpSecurity http) throws Exception { http.authorizeRequests().antMatchers(rutas).permisos .anyRequest().authenticated().and() .formLogin().permitAll().and().logout().permitAll().and() .exceptionHandling().accessDeniedPage("/accessError"); return http.build(); }	Control de acceso: Permisos: permitAll(), hasRole (rol), hasAnyRole (rol1,rol2), denyAll()

application.properties	
server.port=9000	Puerto del servidor (defecto 8080)
spring.thymeleaf.cache=false	Refresh automático plantilla
spring.datasource.url=jdbc:h2:mem:nombreBD spring.datasource.driverClassName=org.h2.Driver spring.datasource.username=sa spring.datasource.password= spring.jpa.database-platform=org.hibernate.dialect.H2Dialect spring.jpa.show-sql=true spring.h2.console.enabled=true	<b>JPA:</b> nombre base de datos, Driver del SGBD usuario/pass de la BD.  Mostrar instruc. SQL en consola Habilitar Consola
server.error.include-message=always server.error.include-stacktrace=never	Configuración de seguridad
spring.mvc.pathmatch.matching-strategy=ant-path-matcher	Para Swagger

Otros	
@Bean CommandLineRunner metodo(argumentos) { return args -> { /*código*/ }; } @Bean CommandLineRunner initData(MiServicio miServicio) { return args -> { miServicio.add( new Persona("pepe",10)); }; }	CommandLineRunner (en clase con main) para código inicial. Ejemplo: añadir al repo desde servicio