**http://www.enmilocalfunciona.io/**

**Curso Spring**

Victorherrerocazurro – [vherrero@pronoide.es](mailto:vherrero@pronoide.es)

FindByJar 🡪 sitio para saber en que jar esta una clase

[Mvnrepository](https://mvnrepository.com/)

Curso Atsistemas Spring

[Spring inicializr](https://start.spring.io/)

[Tutoriales y guías rápida de cómo empezar a currar con spring](https://spring.io/guides)

[OWASP](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project) 🡪 te dicen el mínimo de las seguridad que debe de comtemplar tu app.

[Thymeleaf](https://www.thymeleaf.org/).org.--> para hacer plantillas para pintar pantallas html

JSR-303

* Estudiarr las espresiones regulares para las validaciones de atributos

**Contexto spring**

* Ideal. personas con el mismo nombre
* A fines , que empiecen por la misma letra
* JavaConfir
* @autowire, component, service, repositoriy , qualifier

16. ejercicio. Sprint data JPA

Y publicar como url los requisitos

* Alta y búsqueda a fines

MVC Spring

* Servlets
* ASP.
* Filtros. Fillter
* Listener

Ejecuercico 24

* Utilizar el 14.
* CRUD de las dos entidades
* Obtener las facturas de un cliente
* Obetner cliente de una factura.

**Ejercicio 1.**

Configuración de Spring por xml.

Para poder usar Spring, hay que bajar la librería:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.3.6.RELEASE</version>

</dependency>

Primero se define los vean en un fichero xml

<bean id=*"primerSaludador"*

class=*"com.atsistemas.curso.SaludadorConsolaImpl"*/>

Para iniciar el contexto de spring

ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("ApplicationContext.xml");

Para acceer y usar un bean concreto

Saludador saludador = context.getBean(Saludador.class);

Cada vez que hacemos un getbean, es siempre la misma instancia, al no ser que en la definición se diga lo contrario. Bien en el xml scope="prototype"/> o por anotación en el bean @Scope("prototype")

**Ejercicio 2.**

Igual que el primer ejercicio, pero con configuración por java config.

Para iniciar el contexto de spring:

AnnotationConfigApplicationContext context = **new** AnnotationConfigApplicationContext(ConfiguracionSpring.**class**);

En lugar del xml, creamos una clase que debe tener la anotación

@Configuration junto al nombre de la clase y cada uno de los bean se anota con @Bean

**Ejercicio 3 Patrón Singleton.**

Ejemplo de cómo se implementa a ‘mano’ el patrón singleton.

Solo se pueden instanciar hombres desde la Clase DIOS.

**Ejercicio 4. Inyección por dependencias**

Se crea una interfaz con las operaciones CRUD

Se crean dos implementaciones diferentes, una en memoria y otra por JDBC

SE crea una clase Servicio que explone la operación de Alta. Como en esta clase se utilizan dos implementaciones, hay que utilizar la inyección por contructor o bien por setters, por lo que hay que crear dos variables privadas al intefaz (NO A LA IMPLEMENTAIÓN) y crear un contructor donde se reciban ambas o bien hacerlo por setters. Hay que utilizar el autowire, para que sepa buscar las implementaciones de esa interfaz. Como vamos a inyecdtar dos dao iguales en cuanto a l interfaz y como hay dos implementaciones, utilizamos el @Qualifier para que sepa cuál de las dos usar.

**Ejercicio 5. Relacionado con el 4.**

Ejercicio para exponer operaciones de búsqueda de personas a fines con otras personas..

altaUsuario

busquedaAfines

busquedaIdeal

* Crear una entidad Persona con id y nombre
* Interfaz para las operaciones CRUD de Persona.
* Clase que implemente la interfaz anterior usando persistencia en memoria. Inyección por constructor y autowire en la operación del constructor.
* Interfaz y clase del servicio que contiene la lógica de negocio de las operaciones:

EsAfin . Si se los nombre comienzan con el mismo carácter

EsIdeal. Si se llaman iguales.

Interfaz y clase del servicio que accederá al negocio.

**Ejercicio 6. Properties**

* Se utiliza creando el bean **PropertyPlaceholderConfigurer.** Este bean es el que usa el fichero properties. Hay que decirle donde está el fichero properties y retornar el new bean
* Para hacer uso de las propiedades creadas en el fichero , se hace mediante @Value("${db.user}")

**Ejercicio 6. Multiidioma.**

Hay que usar un bean de tipo MessageSource messageSource() y que retorne un new ResourceBundleMessageSource, donde se le asignará el nombre del fichero properties sin extensión y sin el código de idioma, con el método messageSource.setBasename("Mensajes");, donde el nombre del fichero sería mensaje\_ + idioma , ejemplo “Mensajes\_en.properties”.

Para acceder a los mensajes, sería :

contextoSpring.getMessage(

"page.index.welcome",

new Object[]{"Victor"},

Locale.getDefault()));

El segundo parámetro es object, porque está sustituyendo del fichero el {0}, que es como una variable, ejemplo del fichero:

page.index.welcome=Hola **{0}**

**Ejercicio 8.**

**Ejercicio 9. Aspectos**

Aspecto a mano para entender el fin de spring

**Ejercicio 10. Spring.**

Para aspectos, básicamente es crear una clase que será la que haga el código que queremos filtrar, anotándolo como ASPECT y en los métodos poner qué tipo de paquete+clase+método queremos capturar donde se admiten comodines. Para cada método podemos usar anotaciones para decirles cuando se ejecuta. @after, @before, @afterReturn, etc.

La clase nueva de aspecto, se creará en el fichero de configuración como un bean más.

Ejercicio 23

**Ejercicio 24**