NOTA:
SI HA INSTALADO MONGODB Y NO LE SALE EN SU PC, TENGA CUIDADO!!!
AL INSTALAR MONGODB, ASÉGURSE DE DESMARCAR LA OPCIÓN DE INSTALAR MONGO COMPASS.
ESTA OPCIÓN APARECE DESPUÉS DE QUE SELECCIONA LA OPCIÓN DE INSTALACIÓN COMPLETA.

PARTE I
Acceso a BBDD No-SQL con JPA

Para este taller vamos a usar MongoDB: que es una BD basada en Documentos.
1. Creamos un nuevo proyecto de Spring Boot (File → New → Spring Starter Project). Name: MongoBasico. Group com.cenfotec, Artifact MongoBasico, descripcion mongo basico. Paquete: com.cenfotec.mongobasico.

2. En Depedencias seleccione únicamente Spring Data MongoDB. Finish.

3. Haga un nuevo paquete com.cenfotec.mongobasico.domain

4. En el paquete recién creado agregue una clase llamada Journal.

5. Escriba en la clase Journal los siguientes campos

```
@Id
private String id;
private String title;
private Date created;
private String summary;
@Transient
private SimpleDateFormat format = new SimpleDateFormat("MM/dd/yyyy");
```

Los imports deben quedar asi:

```
import org.springframework.data.annotation.Id;
import org.springframework.data.annotation.Transient;
```

6. Haga los set y get para **Id, title, created y summary**. Agregue este constructor:

```
public Journal(String title, String summary, String date) throws
ParseException{
this.title = title;
this.summary = summary;
this.created = format.parse(date);
}
```

7. Agregue el constructor vacío (sin parámetros)

8. Agregue además este get y el siguiente método toString

```
public String getCreatedAsShort(){
```

```
return format.format(created);
}
public String toString(){
StringBuilder value = new StringBuilder("* JournalEntry(");
value.append("Id: ");
value.append(id);
value.append(",Title: ");
value.append(title);
value.append(",Summary: ");
value.append(summary);
value.append(",Created: ");
value.append(getCreatedAsShort());
value.append(")");
return value.toString();
}
```

9. Agregue el paquete com.cenfotec.mongobasico.repository. Coloque ahí la interfaz JournalRepository, que contendrá:

```
public interface JournalRepository extends MongoRepository<Journal, String> {
  public List<Journal> findByTitleLike(String word);
}
```

10. Haga un neuvo paquete y llámelo com.cenfotec.mongobasico.service

En él, haga una INTERFAZ llamada JournalService. Coloque en ella el siguiente código:

```
public interface JournalService {
      public List<Journal> getAllJournals();
      public void saveJournal(Journal newJournal);
      public List<Journal> findJournalsByTitle(String title);
      public void deleteAllJournals();
}
```

11. En el mismo paquete, agregue la CLASE JournalServiceImpl con el siguiente código

```
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.cenfotec.monbobasico.domain.Journal;
import com.cenfotec.monbobasico.repository.JournalRepository;

@Service
public class JournalServiceImpl implements JournalService {

      @Autowired
```

```
	JournalRepository journalRepo;

	@Override
	public List<Journal> getAllJournals() {
		return journalRepo.findAll();
	}

	@Override
	public void saveJournal(Journal newJournal) {
		journalRepo.save(newJournal);
	}

	@Override
	public List<Journal> findJournalsByTitle(String title) {
		return journalRepo.findByTitleLike(title);
	}

	@Override
	public void deleteAllJournals() {
		journalRepo.deleteAll();
	}

}
```

13. En MongoBasicoApplication, agregue los siguientes imports:

```
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
```

14. Agregue antes del main la siguiente línea:

```
private static final Logger log = LoggerFactory.getLogger(MongoBasicoApplication.class);
```

(el import es de slf4j)

15. Después del main, escriba:

```
@Bean
CommandLineRunner start(JournalService service) {

return args -> {
     log.info("> Deleting existing data...");
     service.deleteAllJournals();
     log.info("> Inserting new data...");
     service.saveJournal(new Journal("Get to know Spring Boot","Today
I will learn Spring Boot","01/02/2016"));
     service.saveJournal(new Journal("Simple Spring Boot Project","I
will do my first Spring Boot Project","01/03/2016"));
     service.saveJournal(new Journal("Spring Boot Reading","Read more
about Spring Boot","02/02/2016"));
     service.saveJournal(new Journal("Spring Boot in the
Cloud","Spring Boot using Cloud Foundry","03/01/2016"));
     log.info("> Getting all data...");
     service.getAllJournals().forEach(entry ->
log.info(entry.toString()));
     log.info("> Getting data using like...");
     service.findJournalsByTitle("Cloud").forEach(entry ->
log.info(entry.toString()));
          };
     }
```

16. En el archivo application.properties coloque las siguientes propiedades:

```
spring.data.mongodb.host= localhost
spring.data.mongodb.port=27017
```

13. Abra el visor de bases de datos NoSql vea el contenido de la BD. Luego, Dele Run a esta aplicación en Spring Boot y refresque. Vea el contenido.

PARTE II

Ahora vamos a crear un Nuevo Proyecto para publicar un api basado en Mongo.

1. Creamos un nuevo proyecto de Spring Boot (File → New → Spring Starter Project). Name: MongoApi. Group com.cenfotec, Artifact MongoApi, descripcion mongo Api. Paquete: com.cenfotec.mongoapi.
2. Seleccinamos como dependencias Spring Web y Spring Data MongoDb.
3. Ya con el proyecto creado, haga el paquete de **domain**. Crea ahí la clase Professional. Le pone los siguientes atributos y constructores:

```
@Id
public String id;
public String name;
public Professional () {
}

public Professional(String name) {
      this.name = name;
}

@Override
public String toString() {
      return "Colleague{" +
            ", name='" + name + '\'' +
      '}';
}
```

4. Agregue el paquete repository. Agregue la interfaz ProfessionalRepository

```
public interface ProfessionalRepository extends MongoRepository<
Professional, String> {
public List< Professional > findByName(String name);
}
```

5. Hacemos un nuevo paquete com.cenfotec.mongoapi.service. En él creamos la interfaz:

```
import com.cenfotec.mongoapi.domain.Professional;

public interface ProfessionalService {

	public void saveProfessional(Professional newProfessional);
}
```

6. En el mismo paquete agregamos la clase ProfessionalServiceImpl

```
@Service
public class ProfessionalServiceImpl implements ProfessionalService {

	@Autowired
	ProfessionalRepository professionalRepo;

	@Override
	public void saveProfessional(Professional newProfessional) {
		professionalRepo.save(newProfessional);
	}

}
```

7.Hagamos ahora el paquete Web. Y le creamos la clase ProfessionalController. Agregamos el siguiente código

```
@RestController
public class ProfessionalController {

@Autowired
private ProfessionalService professionalService;
}
```

6. Despues de private ProfessionalRepository repository… escribimos el siguiente método.

```
@PostMapping("/professional")
public ResponseEntity<String> addColleague(@RequestBody Professional
professional) {
	professionalService.saveProfessional(professional);
	return new ResponseEntity<>(HttpStatus.CREATED);
}
```