DataJPA
=======
1. Mulitple DBIntegration
2. Association(Bi-Directional)
3. Different type ofJoins
4. SpringMail(demonstration)
================================

Note::
 youtube :: https://www.youtube.com/@najaficode3154
 SpringSecurity and Junit with HttpMockito(Notes+videos :: Youtube(Najafi code))

Note: Microservices is not for Freshers
Microservices(From tommo)[Monday would be the last day]
	a. Microservices architecture
	b. ApacheKafka Intergration
	c. RedisCache Integration
	d. Hystric circuit breaker
	e. EurekaClient and EuerkaServer[StockMarket Application,more attributes of application.properties]

Multiple DBIntegration
======================
SpringBoot
	In autoconfiguration style, we can make SpringBoot to communicate with Only one Database

Inputs will be supplied through application.properties
	a. url
	b. username
	c. password
	d. driver-class-name
	e. hibernate specific properties

How to connect with Multiple databases?
 only Application - SpringBoot Data Jpa

To connect with Multiple databases, We need to write the code in SpringStyle(Java Based Configuration)

We need to define below objects(beans) in following order
	a. DataSource
	b. EntityManagerFactory
	c. TransactionManager
		by using Spring Java Configuration Style

```
@Configuration
public class .....{
	//no of methods = no of objects
	@Bean
	public <class/interface> <objName>(){

	}
}
```

Configuration to get a dataSource object for Customer
===
```
@Configuration
```

```
@EnableTransactionManagement
@EnableJpaRepositories(
                  entityManagerFactoryRef = "db1EntityManagerFactory",
                  transactionManagerRef = "db1TransactionManager",
                  basePackages = "in.ineuron.repo.customer"
            )
public class Db1Config {

      // Datasource
      @Bean
      @ConfigurationProperties(prefix = "db1.datasource")
      public DataSource db1DataSource() {
            return DataSourceBuilder.create().build();
      }

      // EntityManagerFactory
      @Bean
      public LocalContainerEntityManagerFactoryBean
db1EntityManagerFactory(EntityManagerFactoryBuilder emfb) {

            HashMap<String, Object> properties = new HashMap<>();
            properties.put("hibernate.hbm2ddl.auto", "update");
            properties.put("hibernate.dialect",
"org.hibernate.dialect.MySQL8Dialect");
            properties.put("hibernate.show_sql", "true");
            properties.put("hibernate.format_sql", "true");

            return
emfb.dataSource(db1DataSource()).packages("in.ineuron.config.model.customer").prope
rties(properties)
                        .persistenceUnit("octbatch").build();

      }

      // TransactionManagement
      @Bean
      public PlatformTransactionManager db1TransactionManager(
                  @Qualifier("db1EntityManagerFactory") EntityManagerFactory
factory) {
            return new JpaTransactionManager(factory);
      }

}
```

application.properites

```
db1.datasource.jdbc-url=jdbc:mysql:///octbatch
db1.datasource.username=root
db1.datasource.password=root123
db1.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

db2.datasource.jdbc-url=jdbc:postgresql://localhost:5432/ineuronDb
db2.datasource.username=postgres
db2.datasource.password=root123
db2.datasource.driver-class-name=org.postgresql.Driver

server.port=4444
```

Association in SpringDataJPA
============================
 Need of keeping databales in association
	=> Keeping multiple entities/items data in single db table
		a. Data Redundency Problem(Duplication problem)
		b. Data Management Problem(Data Inconsistency Problem)

	refer:: .png

Solution1:: keep mulitple entities info in mulitple db tables like user information in seperate db table and phone number details in
		seperate db table

Drawback
	=> Now data redundency problem is gone,but datamanagement/navigation problem is created, we cannot access phone number details
	 from customer tables records and viceversa

Solution2::Take dbtables to maintain multiple entities info and keep them in relatioship using FK(ForeignKey Column)
Advantages
	=> No Data Redundency Problem
	=> No Data Navigation problem,using FK column we canaccess child table records from parent db table records and vice-versa

 If db tables are in relationship then associated entity classes should be taken having relationship,but we keep db tables in relationship using FK
 column where we keep entity class in relationship using Composition and Collection.
 The way dbtables are in relationship is different from the way entity classes in relationship.
 To avoid this mismatching,We use the advanced o-r mapping called "Association Mapping".

SpringDataJPA supports 4 types of Association
	a. One to One
	b. One to Many
	c. Many to One
	d. Many to Many

Association in SpringDataJpa can be implemented in 2 modes
	a. UniDirectional Association [Either Parent to Child or Child to Parent is accessible]
	b. BiDirectional Association [Parent to Child and Child to Parent is accessible]

Association in SpringDataJpa can be build in Entity class in 2 ways
	a. Using Reference type properties in Composition[Non Collection Property] ===> 1 to 1, M to 1
	b. Using Collection type properties in Composition ===> 1 to M, M to M

We can keep dbtables either in unidirectional or in bidirectional association using single FK column, but in java no FK column is available,
so to keep Entity class in association we need to use Composition either collection or non collection type properties.

Cascading in Association
========================

Cascading means, the non select operations performed on the main object will be propogogated to the Associated object.

```
public enum CascadeType {
	/**
	 * Includes all types listed here.
	 */
	ALL,
	/**
	 * Corresponds to {@link javax.persistence.CascadeType#PERSIST}.
	 */
	PERSIST,
	/**
	 * Corresponds to {@link javax.persistence.CascadeType#MERGE}.
	 */
	MERGE,
	/**
	 * Corresponds to {@link javax.persistence.CascadeType#REMOVE}.
	 */
	REMOVE,
	/**
	 * Corresponds to {@link javax.persistence.CascadeType#REFRESH}.
	 */
	REFRESH,
	/**
	 * Corresponds to the Hibernate native DELETE action.
	 */
	DELETE,

}
```