# @Dependent Test

@Test (dependsOnMethod = {“test_name”})

# @DataProviders

## @DataProviders

Los data providers proveen de datos a los tests, de forma tal, de que un mismo test puede ser ejecutado múltiples veces, recibiendo diferentes valores.

```
@DataProvider(name = "personas")
public Object[][] crearPersonas() {
    return new Object[][] {
            { "Juan", new Integer( value: 36) },
            { "Maria", new Integer( value: 37)},
    };
}
```

La forma de declararlo es a través de la notación @DataProvider

```java
@DataProvider(name = "personas")
public Object[][] crearPersonas() {
    return new Object[][] {
            { "Juan", new Integer( value: 36) },
            { "Maria", new Integer( value: 37)},
    };
}
```

La forma de declararlo es a través de la notación @DataProvider y en el método de test, se agrega como parámetro (dataProvider)

```java
@Test(dataProvider = "personas")
public void verifyData1(String nombre, Integer edad) {
    System.out.println(nombre + " " + edad);
}
```



# Clase DataProvider

Se puede crear una clase llamada DataGenerator donde se establezcan diferentes conjuntos de datos a ser utilizados

```java
@DataProvider(name="paises")
public Object[][] paisesConCapitales(){
    return new Object[][] {
            {"Buenos Aires", "Argetina"},
            {"Montevideo", "Uruguay"},
            {"Santiago", "Chile"}
    };
}
```

```java
@Test(dataProvider = "personas", dataProviderClass = DataProvidersClass.class)
public void mostrarDatosPersonas(String nombre, String apellido, String edad)
{
    System.out.println("Mi nombre es" + nombre  + " " + apellido + " y tengo " + edad);
}
```



# @DataProviders

Los data providers no crean una nueva instancia por cada conjunto de datos, sino que ejecutan en una única instancia

# Fakers

## Motivación

Generar datos de forma aleatoria.

Cada vez que ejecuto los tests, quiero tener diferentes datos para enviar

# Libreria Javafakers

```xml
<!-- https://mvnrepository.com/artifact/com.github.javafaker/javafaker -->
<dependency>
    <groupId>com.github.javafaker</groupId>
    <artifactId>javafaker</artifactId>
    <version>1.0.2</version>
</dependency>
```



# Utilizando Fakers

```java
Faker faker = new Faker();

String name = faker.name().fullName(); // Miss Samanta Schmidt
String firstName = faker.name().firstName(); // Emory
String lastName = faker.name().lastName(); // Barton

String streetAddress = faker.address().streetAddress(); // 60018 Sawayn Brooks Suite 449
```

http://dius.github.io/java-faker/apidocs/index.html

https://github.com/DiUS/java-faker



# Crear una clase específica para generar Datos

```
public static Faker faker = new Faker();

public static String getFirstName() {
    //Generating the first name
    String firstName = faker.name().firstName();
    return firstName;
}
```

## Crear una clase específica para generar Datos

```java
public static String getLastName() {
    //Generating last name
    String lastName = faker.name().lastName();
    return lastName;
}
```



## Crear una clase específica para generar Datos

```java
public static String getPhone() {
    //Generating password
    Random random = new Random();

    return String.valueOf(random.nextInt( bound: 1000000) +1000000000);
}
```

# Crear una clase específica para generar Datos

```
public static String getEmail() {
    //Generating email Id
    String emailId = faker.internet().emailAddress();
    return emailId;
}
```