INFORME DE LIGAMIENTO DE POPULATE.XML CON LOS TEST FUNCIONALES

1. Introducción

En este informe explicaremos como ligar los identificadores de bean del populate con los objetos usados en los tests funcionales. Para que los test funcionales siguán funcionando tras cambios en el populate.xml.

2. Resumen de la configuración

Para ello debemos realizar unos cambios en el PopulateDatabase.Java

En el que añadiremos el siguiente método persist:

```java
protected static void persist(final DatabaseUtil databaseUtil, final List<Entry<String, Object>> sortedEntities) {
    String name;
    DomainEntity entity;
    //Properties objecto definition
    final Properties properties = new Properties();
    OutputStream output = null;

    try {
        //output definition
        output = new FileOutputStream("src\\main\\resources\\populate.properties");
        System.out.println();
        databaseUtil.openTransaction();
        for (final Entry<String, Object> entry : sortedEntities) {
            name = entry.getKey();
            entity = (DomainEntity) entry.getValue();

            System.out.printf("> %s", name);
            databaseUtil.persist(entity);
            System.out.printf(": %s%n", entity.toString());
            // TODO: print the entity using SchemaPrinter.  This should get a map in which
            // every persisted entity is mapped onto the corresponding bean name in the
            // PopulateDatabase.xml file; otherwise traceability will be a nightmare.

            //Register of the bean id against the identifier in te properties
            properties.setProperty(name, String.valueOf(entity.getId()));
        }

        //Storafe of the file
        properties.setProperty("noExist", "0");
        properties.store(output, null);
        databaseUtil.closeTransaction();
        System.out.println();
    } catch (final IOException e) {
        e.printStackTrace();
    } finally {
        if (output != null)
            try {
                output.close();
            } catch (final IOException e2) {
                e2.printStackTrace();
            }
    }
}
```

Con el cual se creará un archivo llamado populate.properties que guarda la relación entre los bean id con su identificador único. Este archivo se guardará en src\\main\\resources.

```
Register9=2505
Register8=2504
Register7=2503
Register6=2502
Register5=2501
Register4=2500
Register3=2499
Register2=2498
Register1=2497
chirp4Chorbi1Received=2416
event1Manager3=2469
event1Manager2=2467
attachment6Chirp1Chorbi1Copy=2435
likes3Chorbi1=2481
coordinatesChorbi8=2452
coordinatesChorbi7=2450
coordinatesChorbi6=2448
coordinatesChorbi5=2446
attachment3Chirp1Chorbi1=2426
coordinatesChorbi4=2444
searchTemplateChorbi8=2493
manager6=2404
coordinatesChorbi3=2442
searchTemplateChorbi7=2492
manager5=2403
coordinatesChorbi2=2440
searchTemplateChorbi6=2491
manager4=2402
chirp3Chorbi1Received=2415
coordinatesChorbi1=2438
searchTemplateChorbi5=2490
manager3=2401
searchTemplateChorbi4=2489
manager2=2400
searchTemplateChorbi3=2488
manager1=2399
searchTemplateChorbi2=2487
searchTemplateChorbi1=2486
```

Por último, en la clase AbstracTest, completaremos el método setUpClass lo que cargará cada vez que se ejecute un test el archivo properties, del cual se extrae el Id correspondiente.

```
@Before
public void setUp() {
    AbstractTest.prop = new Properties();
    InputStream input = null;

    try {
        input = new FileInputStream("src/main/resources/populate.properties");

        //load a properties file
        AbstractTest.prop.load(input);

    } catch (final IOException e) {

        e.printStackTrace();
    } finally {
        if (input != null)
            try {
                input.close();
            } catch (final Exception e2) {
                e2.printStackTrace();
            }
    }
}
```

Y añadiendo el método extract:

```
public int extract(final String beanName) {
    int result;
    result = Integer.valueOf(AbstractTest.prop.getProperty(beanName));
    return result;
}
```

Ya podríamos realizar los test sin depender de las Ids de los objetos en base de datos:

```
protected void templateCreateRegister(final String username, final Boolean sorted, final String eventBeanName, final Class<?> expected) {
    Class<?> caught;
    final Register register;
    final Event event;
    caught = null;
    try {
        this.authenticate(username);

        if (!sorted)
            this.eventService.findNextMonthEventsWithPlacesAndFreePlaces();
        else
            this.eventService.findNextMonthEventsWithPlacesSortedAndFreePlaces();

        event = this.eventService.findOne(this.extract(eventBeanName));
        register = this.registerService.create(event);
        this.registerService.save(register);
    } catch (final Throwable oops) {
        caught = oops.getClass();
    }
    this.checkExceptions(expected, caught);
}
```