REPORT A+

Deliverable 02 – Domain Model

María Jiménez Vega
Antonio Nolé Anguita
Álvaro Calle González
Julia García Gallego
Fernando Manuel Ruiz Pliego

Contenido

INTRODUCTION

In the first place, we investigate about the two options that advise us in the project statement. After researching we decided on JSON because is faster than YAML and more interoperable whit more system and it could be more useful in the future.

Secondly, we investigate the different libraries that can be used to implement what is requested. We saw that there are several, including Jackson [1], GSON [2] among others. [3]

Finally, we opted for GSON because although Jackson provided it to us spring [4], GSON seemed easier to implement. In addition, facing a different library because of Jackson already knew her.

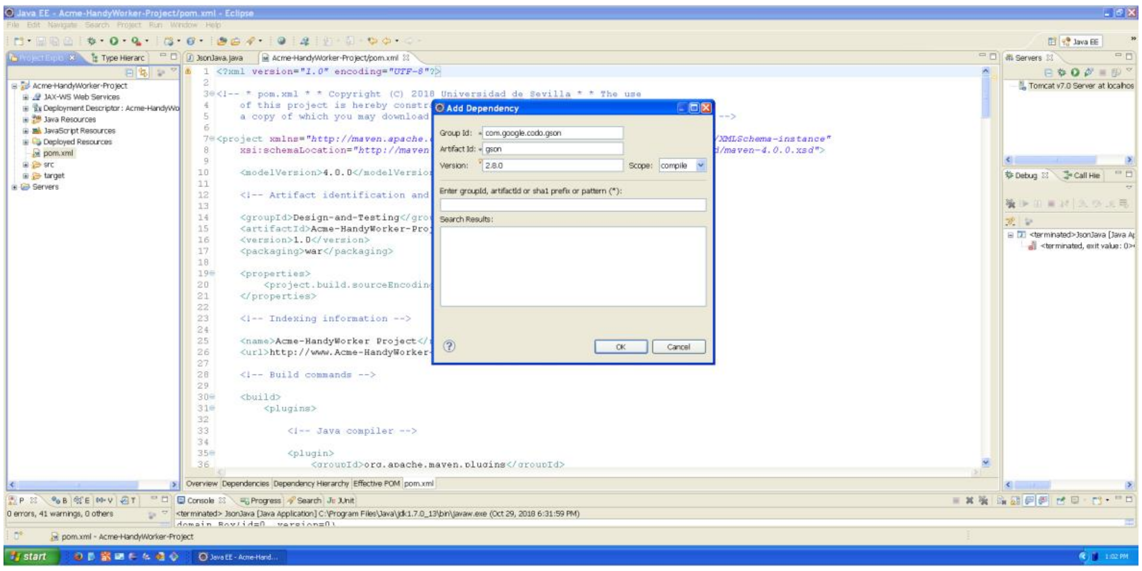
GSON INSTALLATION IN THE PROJECT

To install GSON in the project we must open the pom.xml file [5]. We right click on the file and fill in the fields as it appears in the following picture.

Group ID = com.google.code.gson

Artifact ID = gson

Version = 2.8.0



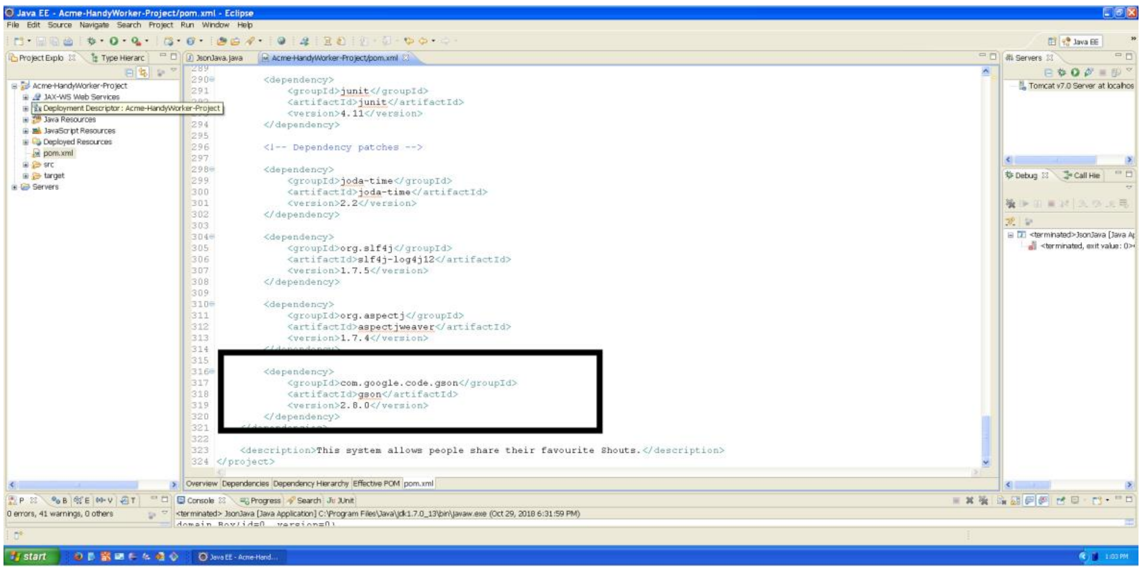
Add Dependency

Group Id: com.google.codo.gson
Artifact Id: gson
Version: 2.8.0
Scope: compile

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!-- * pom.xml * * Copyright (C) 2018 Universidad de Sevilla * * The use
    of this project is hereby constr
    a copy of which you may download                                  -->

<project xmlns="http://maven.apache.                                 /XMLSchema-instance"
    xsi:schemaLocation="http://maven                                 d/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersio

    <!-- Artifact identification and

    <groupId>Design-and-Testing</gro
    <artifactId>Acme-HandyWorker-Pro
    <version>1.0</version>
    <packaging>war</packaging>

    <properties>
        <project.build.sourceEncodin
    </properties>

    <!-- Indexing information -->

    <name>Acme-HandyWorker Project</
    <url>http://www.Acme-HandyWorker-

    <!-- Build commands -->

    <build>
        <plugins>

            <!-- Java compiler -->

            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
```

```
<terminated> JsonJava [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Oct 29, 2018 6:31:59 PM)
```

We click ok and the following code appears automatically:



```xml
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.11</version>
        </dependency>

        <!-- Dependency patches -->

        <dependency>
            <groupId>joda-time</groupId>
            <artifactId>joda-time</artifactId>
            <version>2.2</version>
        </dependency>

        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.7.5</version>
        </dependency>

        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.7.4</version>
        </dependency>

        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.8.0</version>
        </dependency>
    </dependencies>

    <description>This system allows people share their favourite Shouts.</description>
</project>
```

```
<terminated> JsonJava [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Oct 29, 2018 6:31:59 PM)
```

To make sure everything is correct, we right click ok the proyect mave>update maven project and select check “focer update of snapshots/releases”.

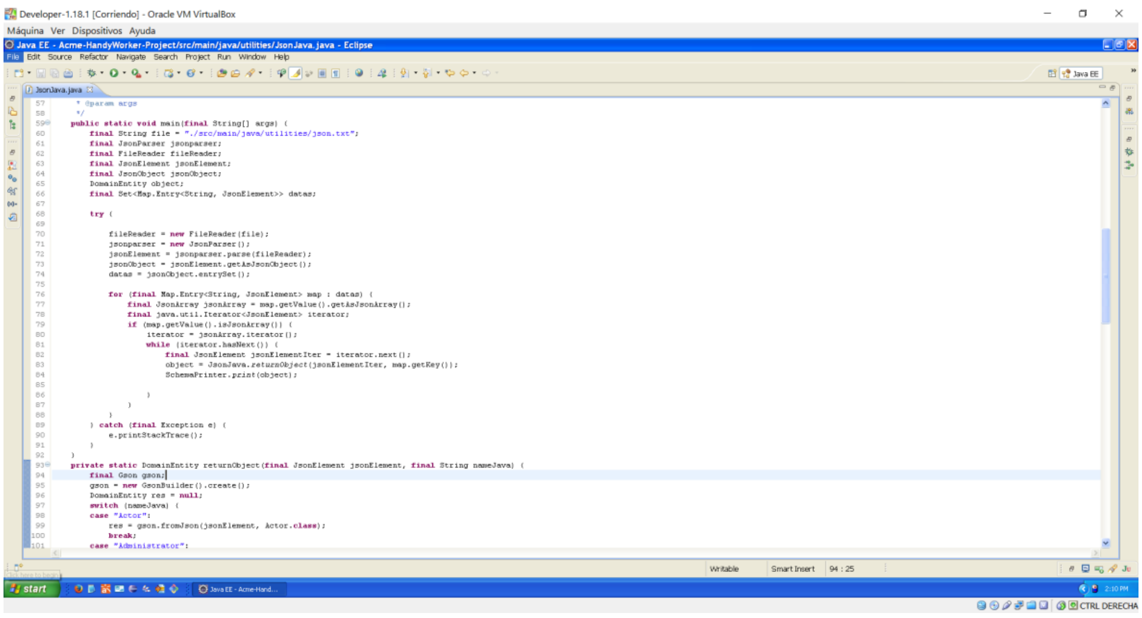
CODE

To implement the functionality, you must create a java class and a file transformed in an object. Both are in src>main>java>utilities.

We have called the java class which implements the code that will be executed. This class is composed of a main method in which the following steps are performed within a try/catch:

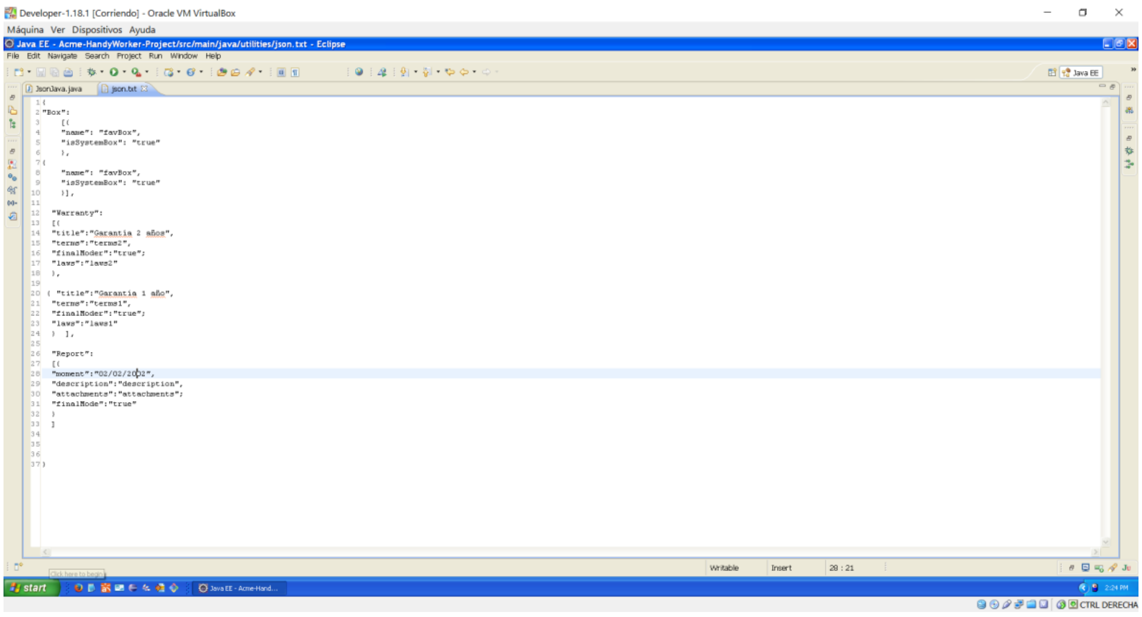
1. Read json.txt file.
2. The file is passed by jsonParser so that they become data.
3. We transform it into an object.
4. The object through the method entrySet() we have a set of data.
5. Through a method we run the data which after making certain checks and making use of “Iterator” to go through the data that is within the map.
6. Once we have the piece of the JSON file that only contains what corresponded to a java entity, it passes through a private method that returns this data converted into a java object. This is done through the method offered by the GSON library “fromJson”, to which an element JSON is passed and the class to which it corresponds. To know which java class, it belongs to we do it with a switch method, by passing the name of the class in string we obtain its java class.

While we were doing tests, we verified that it does not detect well the format of the dates. To solve this problem, we have used setDateFormat to format the dates [6].



```java
     * @param args
     */
    public static void main(final String[] args) {
        final String file = "./src/main/java/utilities/json.txt";
        final JsonParser jsonparser;
        final FileReader fileReader;
        final JsonElement jsonElement;
        final JsonObject jsonObject;
        DomainEntity object;
        final Set<Map.Entry<String, JsonElement>> datas;

        try {

            fileReader = new FileReader(file);
            jsonparser = new JsonParser();
            jsonElement = jsonparser.parse(fileReader);
            jsonObject = jsonElement.getAsJsonObject();
            datas = jsonObject.entrySet();

            for (final Map.Entry<String, JsonElement> map : datas) {
                final JsonArray jsonArray = map.getValue().getAsJsonArray();
                final java.util.Iterator<JsonElement> iterator;
                if (map.getValue().isJsonArray()) {
                    iterator = jsonArray.iterator();
                    while (iterator.hasNext()) {
                        final JsonElement jsonElementIter = iterator.next();
                        object = JsonJava.returnObject(jsonElementIter, map.getKey());
                        SchemaPrinter.print(object);

                    }
                }
            }
        } catch (final Exception e) {
            e.printStackTrace();
        }
    }
    private static DomainEntity returnObject(final JsonElement jsonElement, final String nameJava) {
        final Gson gson;
        gson = new GsonBuilder().create();
        DomainEntity res = null;
        switch (nameJava) {
        case "Actor":
            res = gson.fromJson(jsonElement, Actor.class);
            break;
        case "Administrator":
```

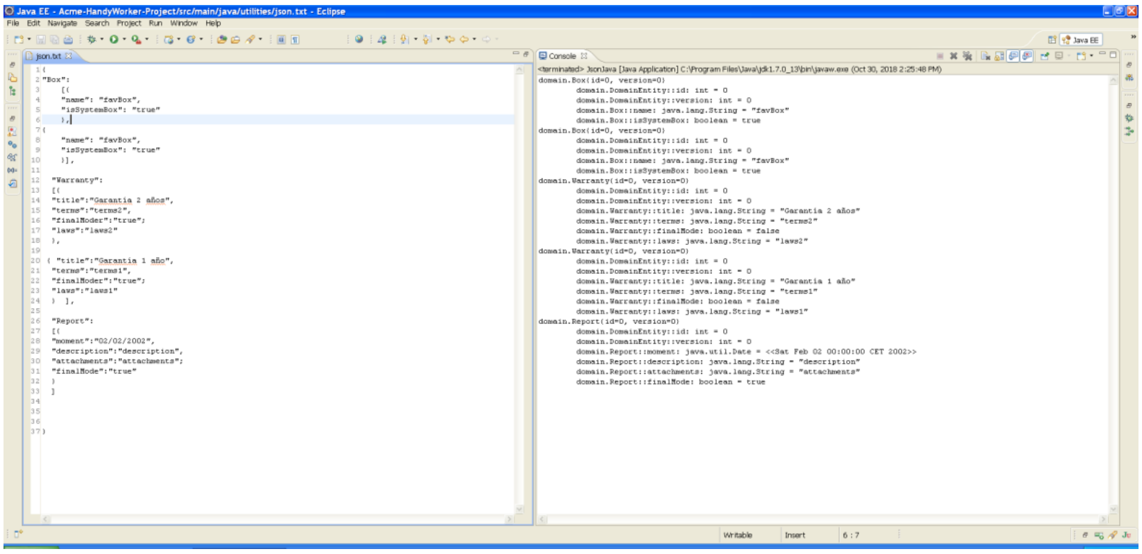
Finally, we created a file in which we created JSON objects and that we later used to test the implemented.



```
{
"Box":
    [{
    "name": "favBox",
    "isSystemBox": "true"
    },
{
    "name": "favBox",
    "isSystemBox": "true"
    }],

  "Warranty":
  [{
  "title":"Garantia 2 años",
  "terms":"terms2",
  "finalModer":"true";
  "laws":"laws2"
  },

 { "title":"Garantia 1 año",
  "terms":"terms1",
  "finalModer":"true";
  "laws":"laws1"
  }  ],

  "Report":
  [{
  "moment":"02/02/2002",
  "description":"description",
  "attachments":"attachments";
  "finalMode":"true"
  }
  ]



}
```

TEST

Continued you can see the tests that we have done.


Java EE - Acme-HandyWorker-Project/src/main/java/utilities/json.txt - Eclipse
File Edit Navigate Search Project Run Window Help
json.txt
Console
Java EE
Writable
Insert
6 : 7

```
{
"Box":
    [{
    "name": "favBox",
    "isSystemBox": "true"
    },
{
    "name": "favBox",
    "isSystemBox": "true"
    }],

  "Warranty":
  [{
  "title":"Garantia 2 años",
  "terms":"terms2",
  "finalModer":"true";
  "laws":"laws2"
  },

 { "title":"Garantia 1 año",
  "terms":"terms1",
  "finalModer":"true";
  "laws":"laws1"
  }  ],

  "Report":
  [{
  "moment":"02/02/2002",
  "description":"description",
  "attachments":"attachments";
  "finalMode":"true"
  }
  ]



}
```

```
<terminated> JsonJava [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Oct 30, 2018 2:25:48 PM)
domain.Box{id=0, version=0}
        domain.DomainEntity::id: int = 0
        domain.DomainEntity::version: int = 0
        domain.Box::name: java.lang.String = "favBox"
        domain.Box::isSystemBox: boolean = true
domain.Box{id=0, version=0}
        domain.DomainEntity::id: int = 0
        domain.DomainEntity::version: int = 0
        domain.Box::name: java.lang.String = "favBox"
        domain.Box::isSystemBox: boolean = true
domain.Warranty{id=0, version=0}
        domain.DomainEntity::id: int = 0
        domain.DomainEntity::version: int = 0
        domain.Warranty::title: java.lang.String = "Garantia 2 años"
        domain.Warranty::terms: java.lang.String = "terms2"
        domain.Warranty::finalMode: boolean = false
        domain.Warranty::laws: java.lang.String = "laws2"
domain.Warranty{id=0, version=0}
        domain.DomainEntity::id: int = 0
        domain.DomainEntity::version: int = 0
        domain.Warranty::title: java.lang.String = "Garantia 1 año"
        domain.Warranty::terms: java.lang.String = "terms1"
        domain.Warranty::finalMode: boolean = false
        domain.Warranty::laws: java.lang.String = "laws1"
domain.Report{id=0, version=0}
        domain.DomainEntity::id: int = 0
        domain.DomainEntity::version: int = 0
        domain.Report::moment: java.util.Date = <<Sat Feb 02 00:00:00 CET 2002>>
        domain.Report::description: java.lang.String = "description"
        domain.Report::attachments: java.lang.String = "attachments"
        domain.Report::finalMode: boolean = true
```

BIBLIOGRAPHY

1. https://www.adictosaltrabajo.com/2011/02/09/jackson-deserialize-with-constructor/
2. https://www.adictosaltrabajo.com/2012/09/17/gson-java-json/
3. https://www.baeldung.com/jackson-vs-gson
4. https://springframework.guru/processing-json-jackson/
5. https://stackoverflow.com/questions/26701452/gson-dependency-which-repository
6. https://stackoverflow.com/questions/7910734/gsonbuilder-setdateformat-for-2011-10-26t202959-0700