

**Spring API Rest MVC with Authentication**

**Design and Testing II**

**Group 6**

Ángel Delgado Luna

Belen Garrido López

Ezequiel Portillo Jurado

Alejandro Rodríguez Díaz

Mª de Gracia Piñero Pastor

Índice

Tabla de contenido

[Foundations 3](#_Toc9789880)

[Security 4](#_Toc9789881)

[CRUD Actions 5](#_Toc9789882)

[Create 6](#_Toc9789883)

[Edit/Get 7](#_Toc9789884)

[List 8](#_Toc9789885)

[Delete 9](#_Toc9789886)

[Bibliography 10](#_Toc9789887)

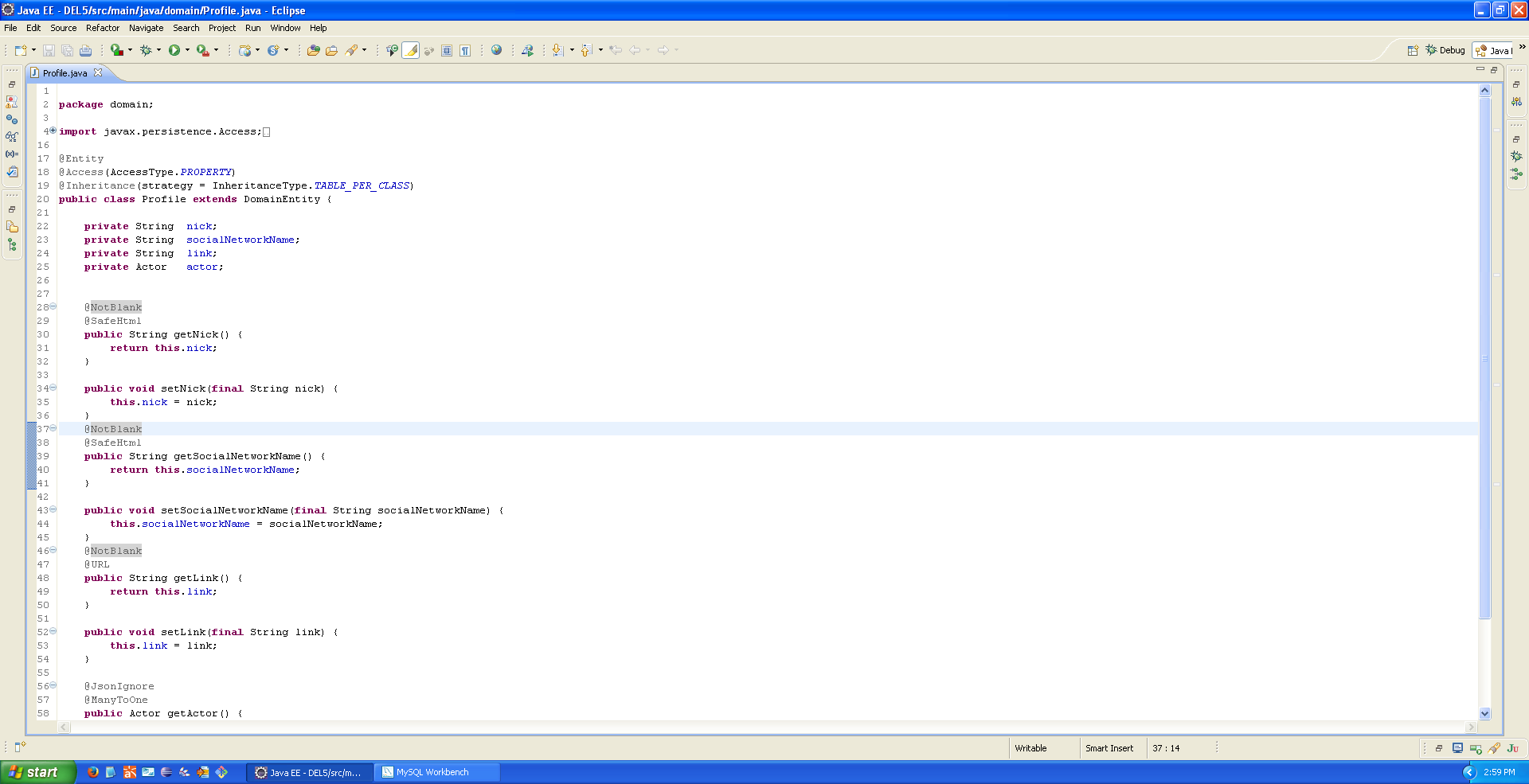
# Foundations

Nowadays, we can find a lot of platforms for data consuming. An example, google calendar. Google calendar has an API (Application Programming Interface) which lets users to consumes its calendars in any application, taking into account that you must have an account previously.

We must take into account that using a secure API will allow developers to build a comfortable architecture without entering on important details for each implementation. JSON is a data format which helps us to make this action; because, we only have to take a model that is going to be used later by our application.

In this example, we have chosen social profiles entity from any user. Let´s see a short schema:





According to this, we are going to generate our own api with this entity. Let´s move on the next items from this document!

Maven Dependencies

Dependency 1 – Jackson (New on the project)

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.5.3</version>

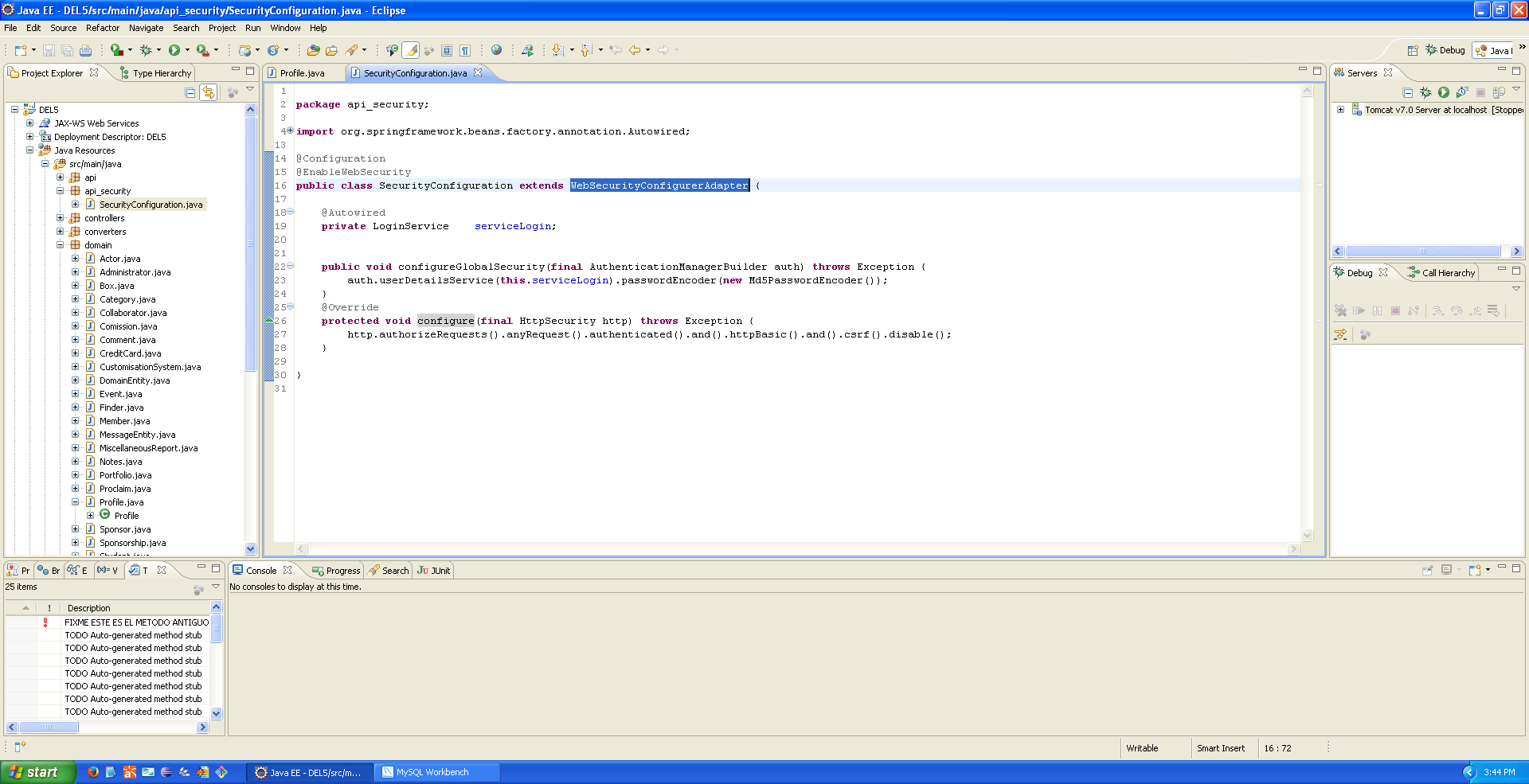
</dependency>

Dependency 2 – Spring Dependencies (Core – WebMVC – ORM)

# Security

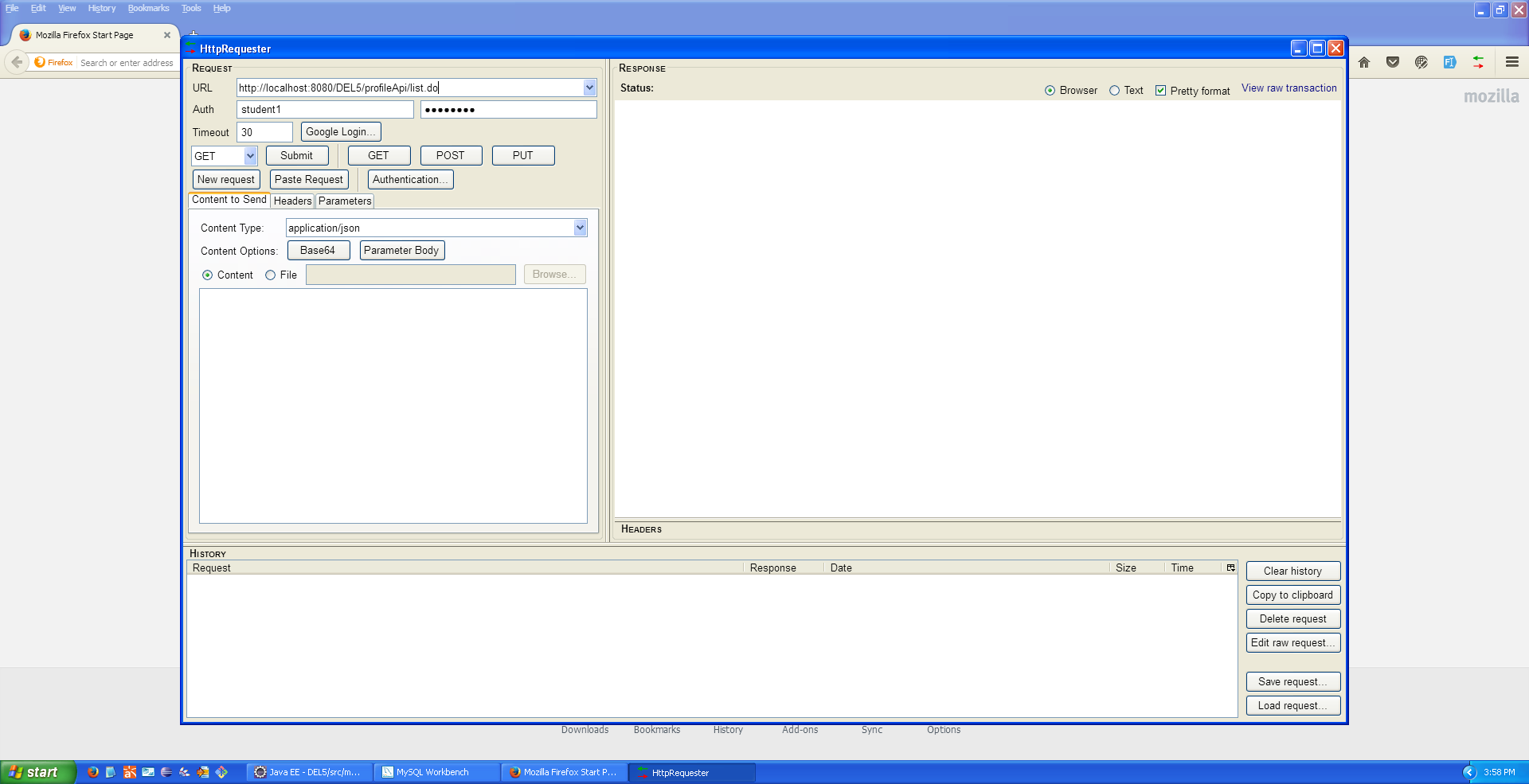
The first step we must do, it´s to configure a security system for users login. In our case, we have chosen a basic authentication: username and password (encrypted with MD5). Having a look to the package structure on the project attached to this deliverable, you are supposed to see two packages called “api” and “api\_security”; on this item we will focus on the second named.

Api\_security packaged contains a class named as “SecurityConfiguration” which extends from “WebSecurityConfigurerAdapter”.



The first method call “configureGlobalSecurity” receives as parameter an object of authentication manager. This will let us to build the authentication in our system according to the parameters receive from LoginService class. This class extends from UserDetailsService (native from Spring) which gives us the main parameter to make this configuration (username and password). As indicated, a MD5PasswordEncoder instance is given as password enconder for the auth.

In the second method, we are saying that every http call must be authenticated if it is required.



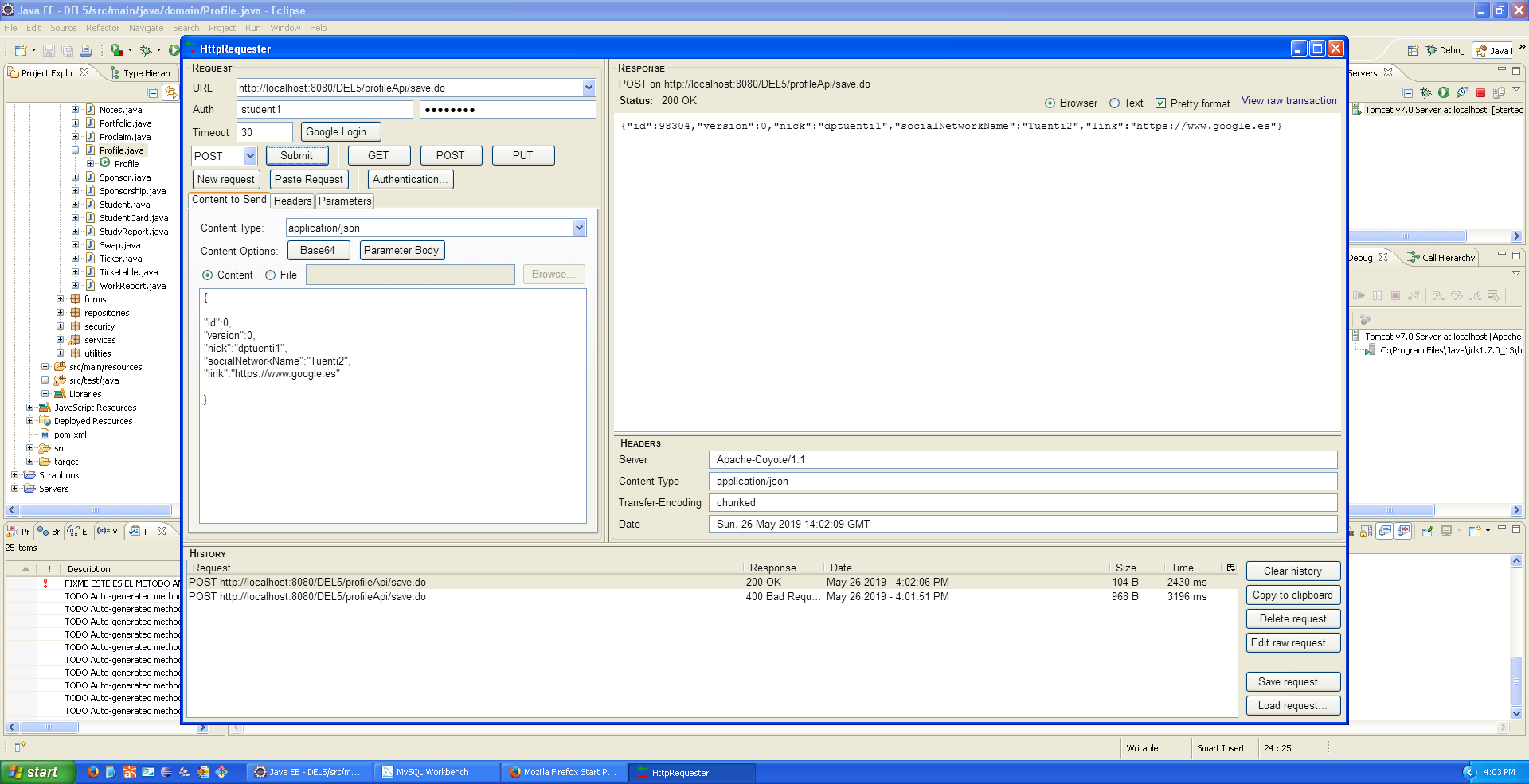
# CRUD Actions

In the first package called “api”, we can find our entity configured for external use. Let see first some important annotations:

* @RestController: With this annotation we are indicating to spring that this is a controller which uses a json data structure.
* @ResponseBody: The returned items will be given as JSON data.
* @RequestBody: The post data will have a json data structure.
* @JsonIgnore: Profile has an actor attribute. When we are working with social profiles, we do not need any actor data, as consequence we want that spring ignores this parameter. (Use in java model above the actor get method)

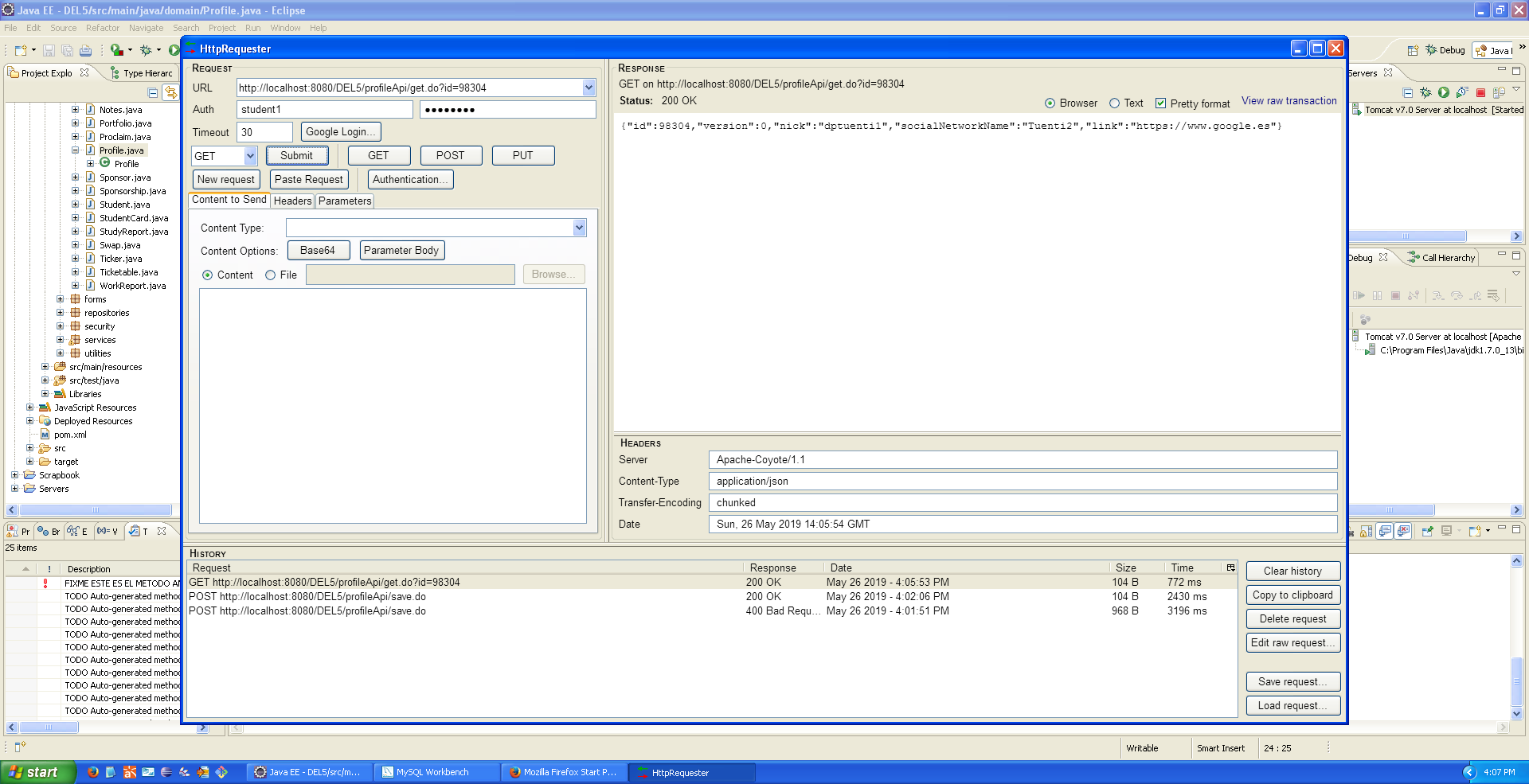
## Create

1. We first make the json structure necessary.
2. We introduce as url to post: <http://localhost:8080/DEL5/profileApi/save.do>
3. We click on submit changing its param to POST action



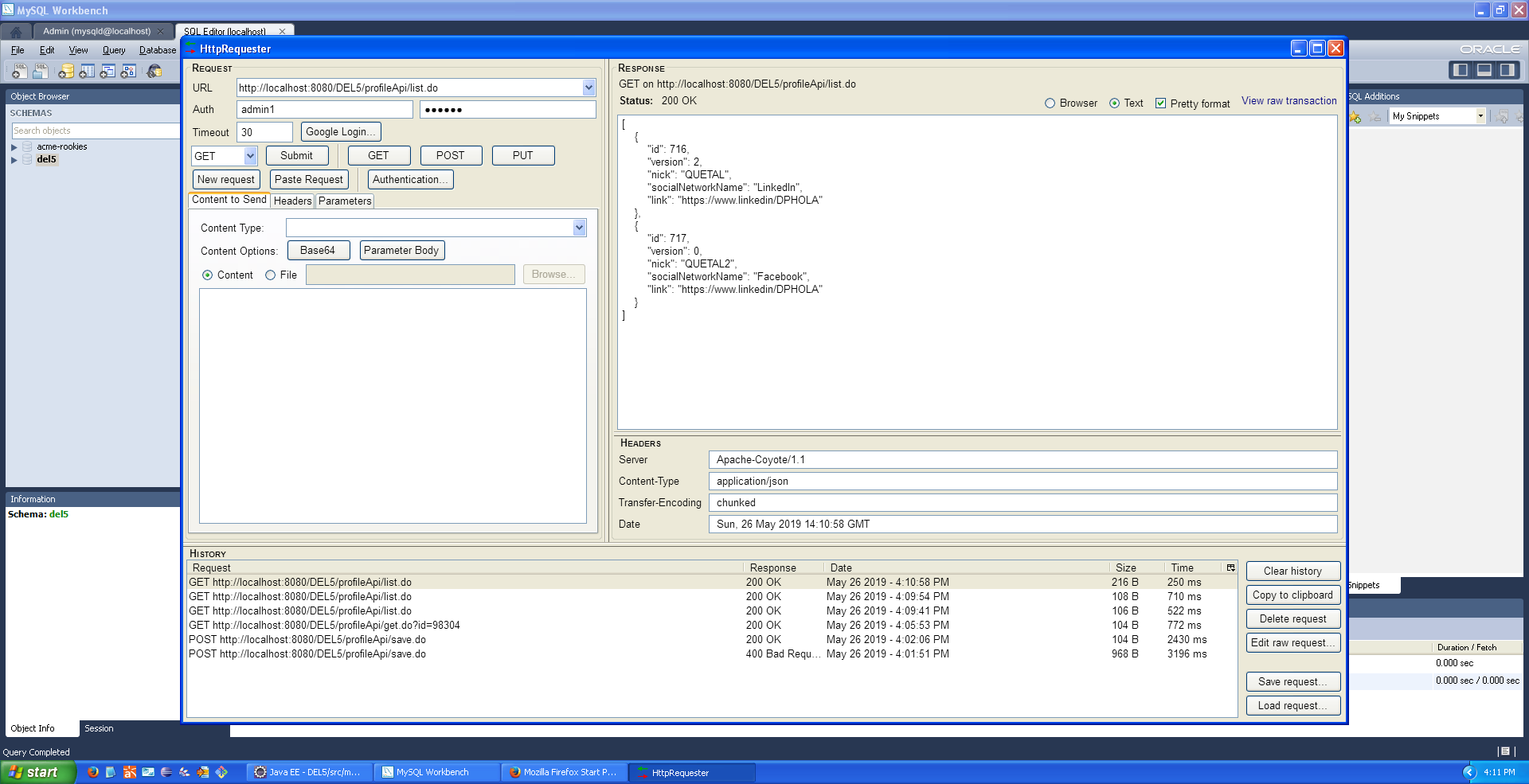
## Edit/Get

1. We introduce the following url: <http://localhost:8080/DEL5/profileApi/get.do?id=98304> in order to get the object with the id requested (In our example, the one created on the item before). Id = 98304.
2. We submit the request taking into account that we must use GET HTTP Request.



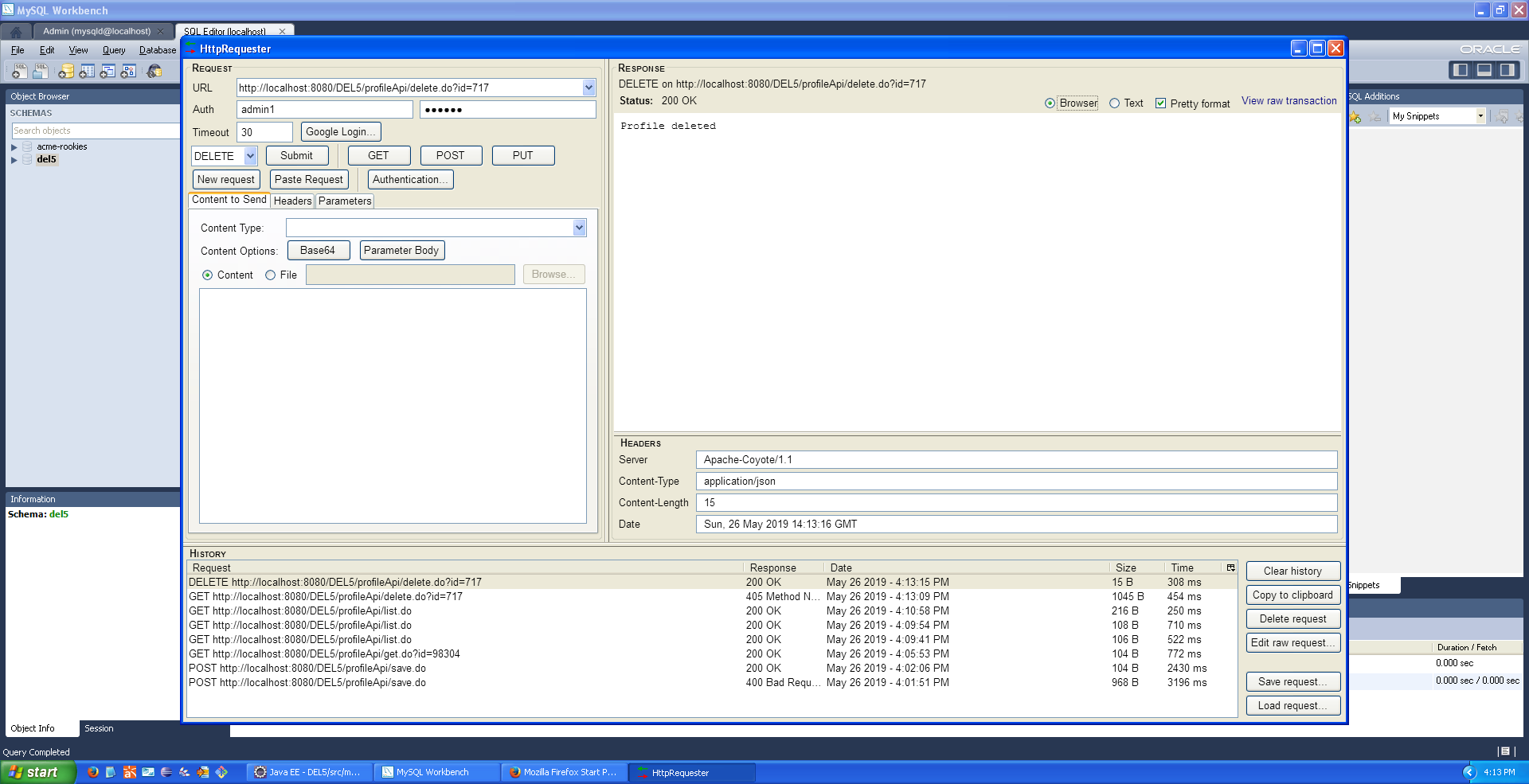
## List

1. We introduce the following url: http://localhost:8080/DEL5/profileApi/list.do in order to get the profiles from the user logged.
2. We submit the request taking into account that we must use GET HTTP Request.

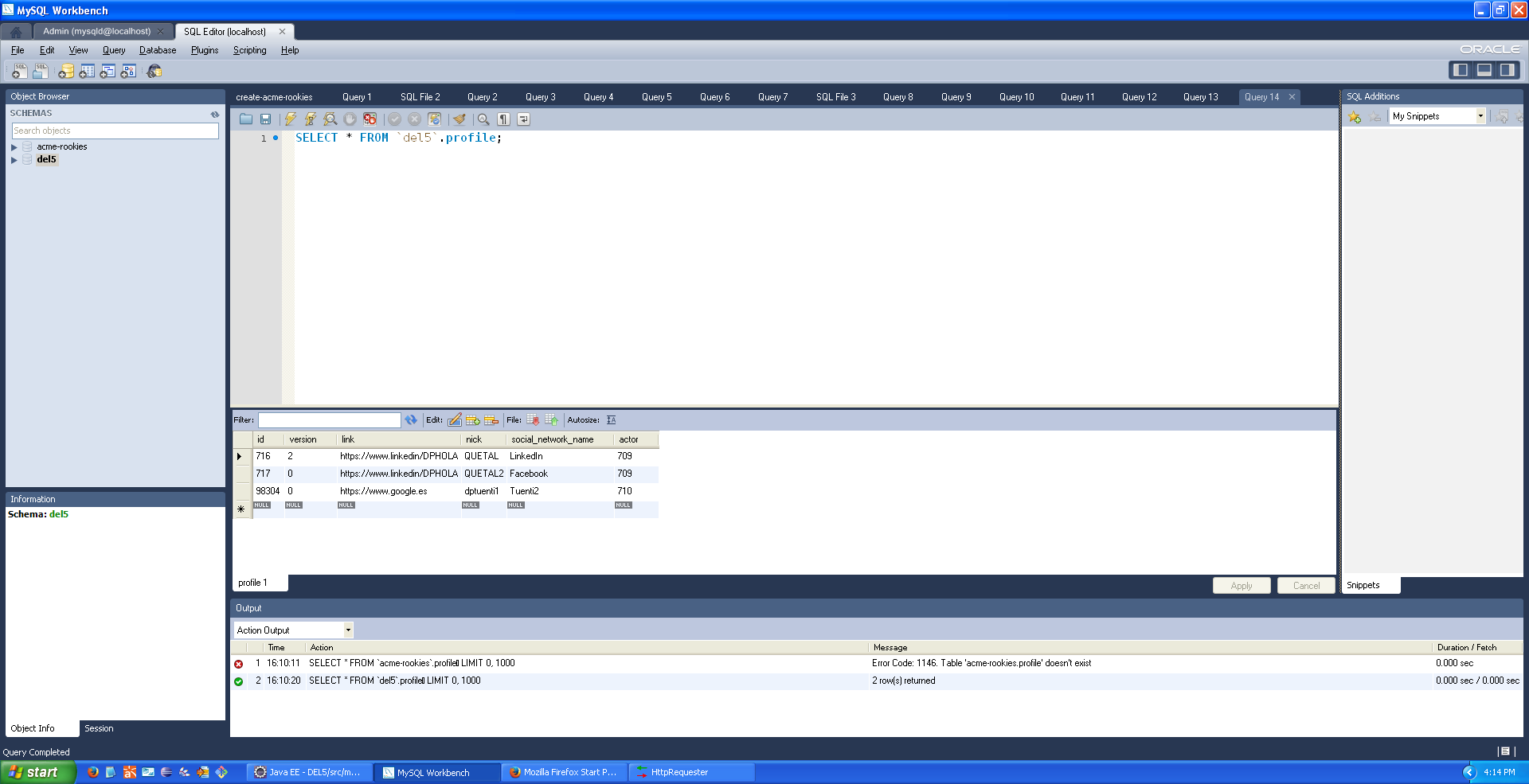


## Delete

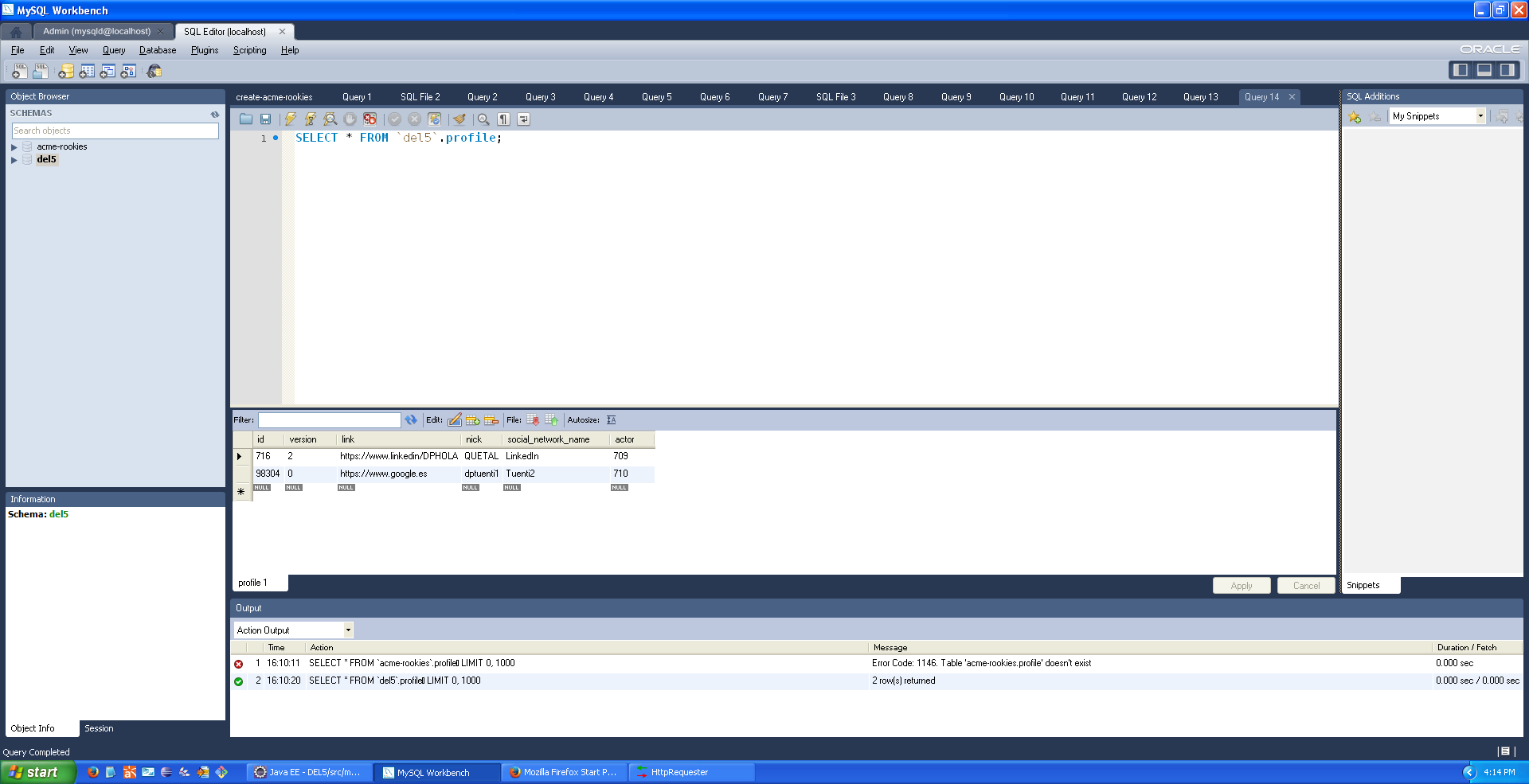
1. We introduce the following url: http://localhost:8080/DEL5/profileApi/delete.do?id=717 in order to delete the profile with the id indicated.
2. We submit the request taking into account that we must use DELETE HTTP Request.



Before



After



# Bibliography

* [1] <https://stackoverflow.com/questions/29838960/how-to-dynamically-remove-fields-from-a-json-response>
* [2] <https://medium.com/@ziatheonlyone/spring-rest-api-part-3-spring-security-basic-authentication-3fd20342745b>