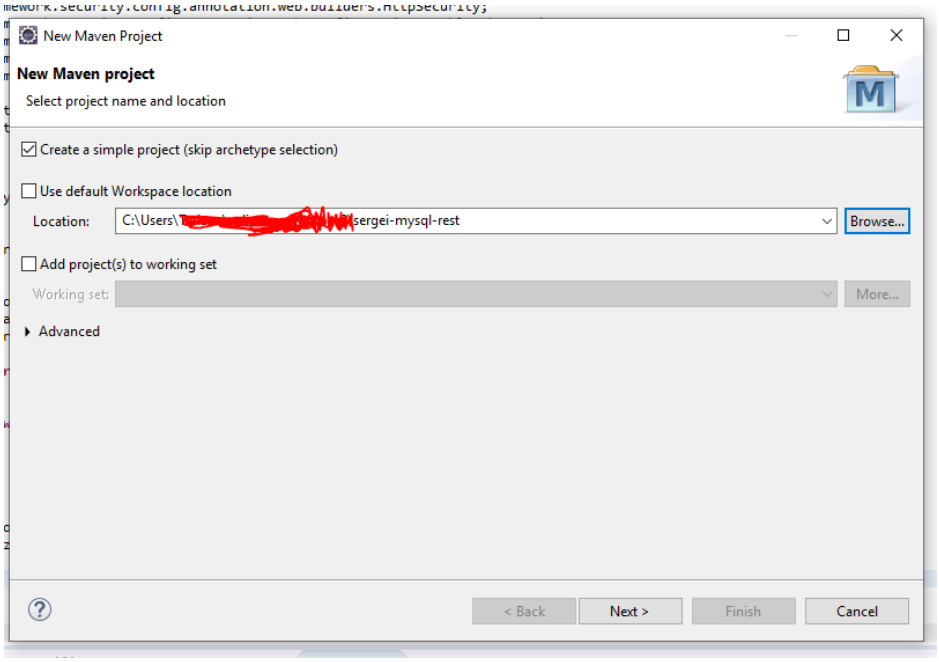
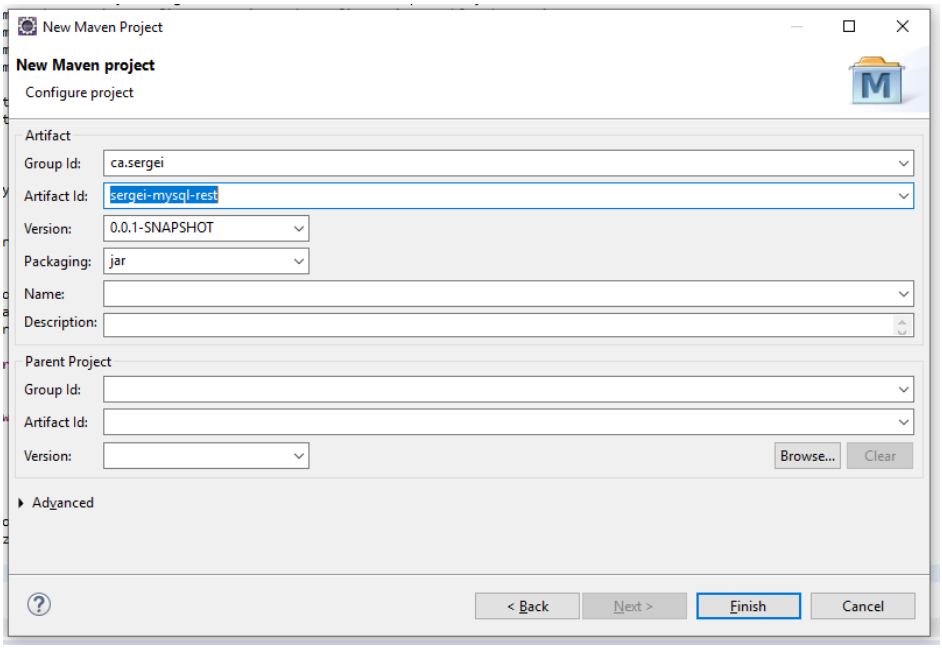
Step 1 (creation):

Create a new simple Maven Project:





Step 2 (dependences):

Open the pom.xml and add dependences:

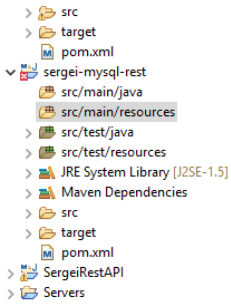
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>ca.sergei</groupId>
  <artifactId>sergei-mysql-rest</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <!-- Define the parent pom -->
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>1.5.8.RELEASE</version>
        </parent>
```

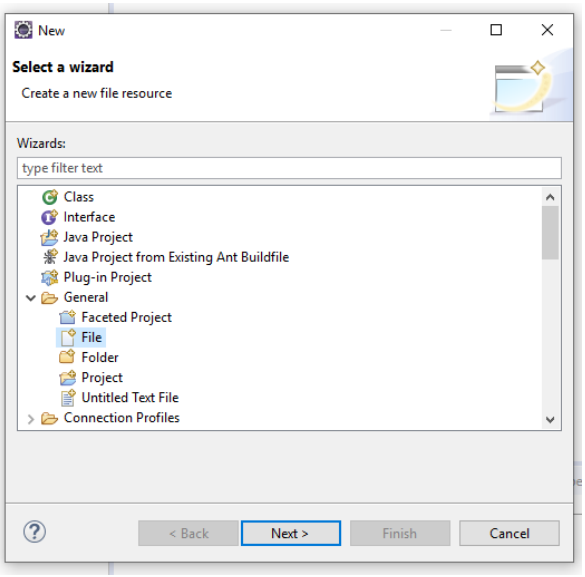
```
	<!-- Set the Java version -->
	<properties>
		<java.version>1.8</java.version>
	</properties>

	<dependencies>
		<!-- Spring Boot -->
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<!-- JPA Data -->
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>

		<!-- Data REST -->
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-rest</artifactId>
		</dependency>

		<!-- MySQL Connector-J -->
		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
		</dependency>

	</dependencies>

	<!-- Spring Boot Maven Plugin -->
	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

</project>
```

Step 3 (connection to the database):

Create a configuration file with connection to the database.

Create a new file named application.properties inside the src/main/resources directory.

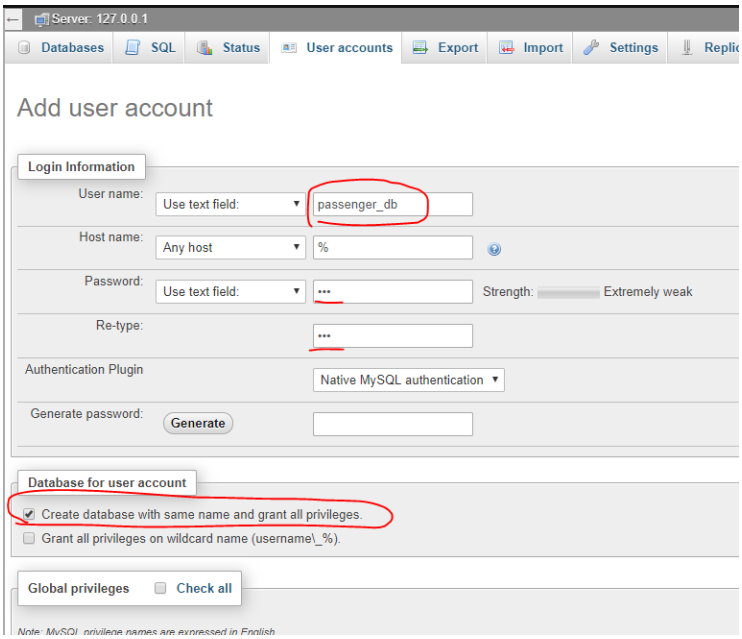




```
# Automatically update the database
spring.jpa.hibernate.ddl-auto=update

# The database connection URL
spring.datasource.url=jdbc:mysql://localhost:3306/passenger_db?useSSL=false

# Username
spring.datasource.username= passenger_db

# Password
spring.datasource.password=123

# Define the database platform
spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect

# Define the naming strategy
spring.jpa.hibernate.naming-strategy = org.hibernate.cfg.ImprovedNamingStrategy

# Define the default schema
spring.jpa.properties.hibernate.default_schema=schema
```

Step 4 (create an database):

Open PhpMyAdmin or whatever the MySQL client you prefer and create an empty database passenger_db.



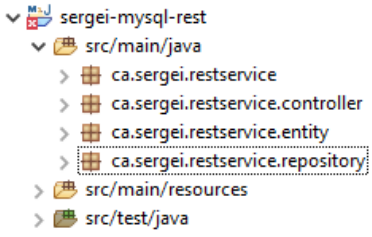
We have just created the database. Tables and data will be automatically created at the runtime based on java entity classes.

Step 5 (packages):

Create a packages:

ca.yourname.restservice
ca.yourname.restservice.controller
ca.yourname.restservice.entity
ca.yourname.restservice.repository

in the src/main/java



Step 6 (class Application):

Create a class Application inside the ca.yourname.restservice package

```
package ca.sergei.restservice;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

import ca.sergei.restservice.entity.Passenger;
import ca.sergei.restservice.repository.PassengerRepository;

@SpringBootApplication
public class Application {

	public static void main(String[] args) {
		SpringApplication.run(Application.class, args);
	}

}
```

This class is the starting point of our application. Running this class will run the application as a web service.

Step 7 (entity - class Passenger):

Lets create entity - class Passenger in the ca.yourname.restservice.entity.

```
package ca.sergei.restservice.entity;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Passenger {

	@Id
	private String login;
	private String password;
	private String name;
	private String email;

// Getters and Setters (Omitted for brevity)
```

This class is decorated by the @Entity annotation and the primary key is decorated with the @Id annotation.

Step 8 (interface PassengerRepository):

Create an interface PassengerRepository in the ca.yourname.restservice.repository.

```
package ca.sergei.restservice.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.rest.core.annotation.RestResource;

import org.springframework.data.rest.core.annotation.RepositoryRestResource;

import ca.sergei.restservice.entity.Passenger;

@RepositoryRestResource(path = "/passengers")

public interface PassengerRepository extends JpaRepository<Passenger, String> {

}
```

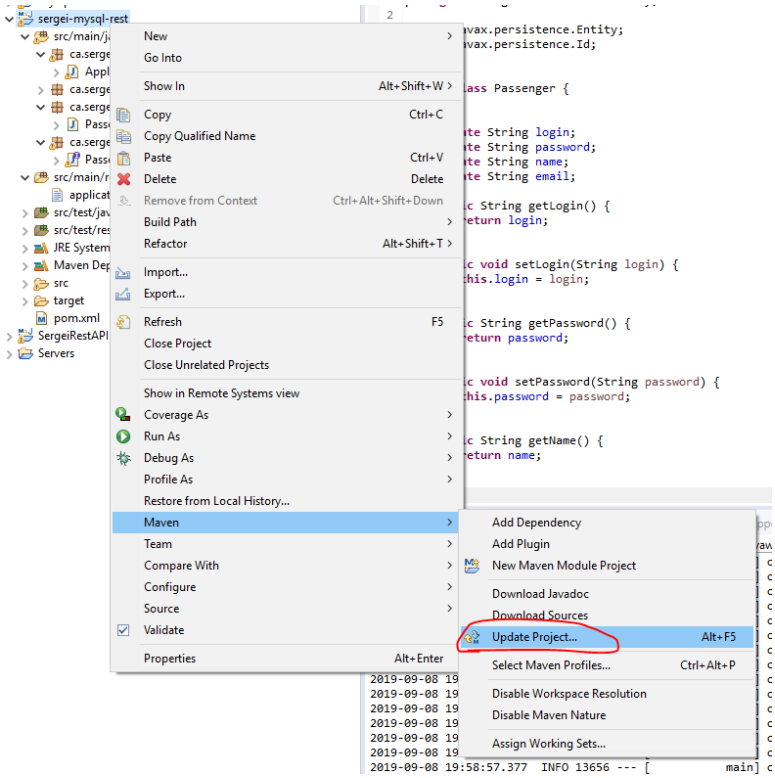
This interface extends JpaRepository and the two generic types defined are the entity class and the type of its primary key. In this example, the entity class is Passenger and its primary key login is String. The @ RepositoryRestResource annotation marks this interface as a REST resource serving the /passengers URL.

Now, a basic RESTful CRUD (create, read, update and delete) web service is ready!

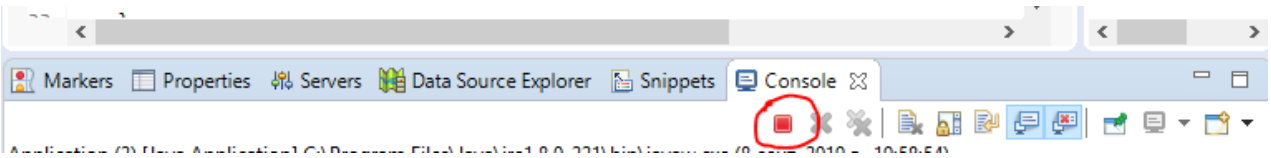
Let's test it:

Step 9 (RESTful CRUD is ready. Test):

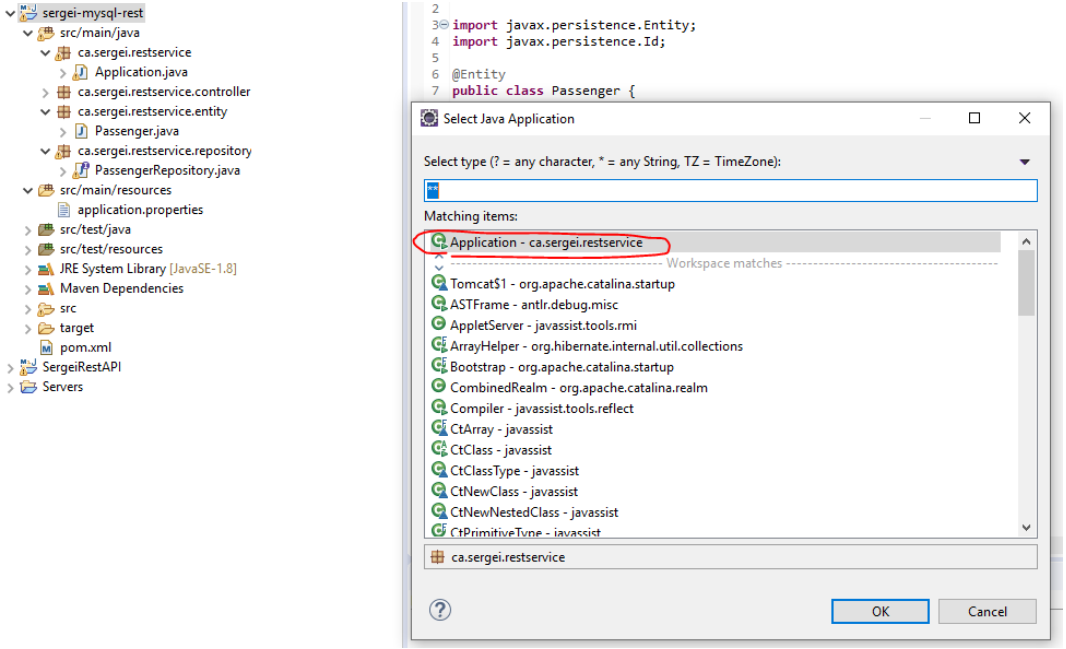
Right click on the project and select Maven -> Update Project…



Terminate all projects if they are running



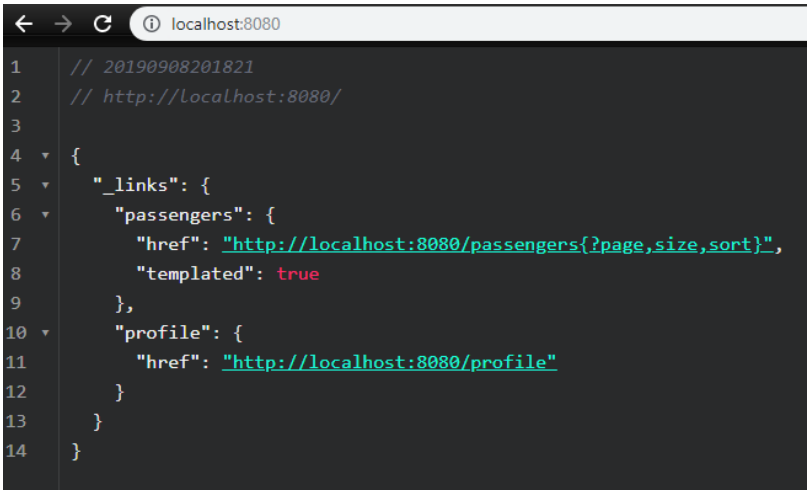
After, right click on the project and select Run As → Java Application. If asked, select the Application class as the main class to execute.



On the console log you have to see:

```
2019-09-08 20:16:19.740  INFO 11432 --- [          main] o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on startup
2019-09-08 20:16:19.773  INFO 11432 --- [          main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
2019-09-08 20:16:19.776  INFO 11432 --- [          main] ca.sergei.restservice.Application        : Started Application in 2.996 seconds (JVM running for 3.221)
```

On the browser for localhost:



Try the REST APIs exposed by the PassengerRepository. You can create a new passenger using POST request and read all the passengers by sending a GET request to the same URL.

POST http://localhost:8080/passengers

Authorization | Headers (2) | Body | Pre-request Script | Tests

form-data | x-www-form-urlencoded | raw | binary | JSON (application/json)

```
{
    "login":"passenger1",
    "password":"passenger1",
    "name":"Passenger1",
    "email":"passenger1@passenger1.ca"
}
```

Body | Cookies | Headers (4) | Test Results

Pretty | Raw | Preview | JSON

```
{
    "password": "passenger1",
    "name": "Passenger1",
    "email": "passenger1@passenger1.ca",
    "_links": {
        "self": {
            "href": "http://localhost:8080/passengers/passenger1"
        },
        "passenger": {
            "href": "http://localhost:8080/passengers/passenger1"
        }
    }
}
```

GET http://localhost:8080/passengers

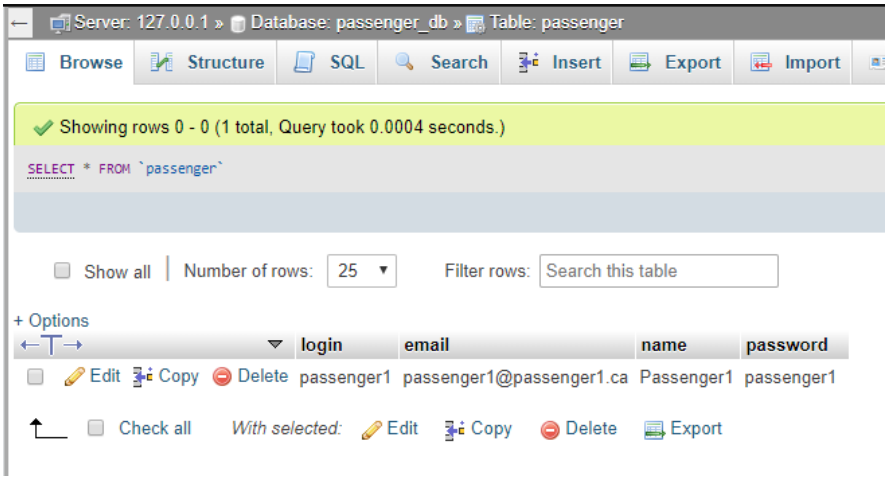
Authorization | Headers (2) | Body | Pre-request Script | Tests

Type: No Auth

Body | Cookies | Headers (3) | Test Results

Pretty | Raw | Preview | JSON

```
{
    "_embedded": {
        "passengers": [
            {
                "password": "passenger1",
                "name": "Passenger1",
                "email": "passenger1@passenger1.ca",
                "_links": {
                    "self": {
                        "href": "http://localhost:8080/passengers/passenger1"
                    },
                    "passenger": {
                        "href": "http://localhost:8080/passengers/passenger1"
                    }
                }
            }
        ]
    },
    "_links": {
        "self": {
            "href": "http://localhost:8080/passengers{?page,size,sort}",
            "templated": true
        },
        "profile": {
            "href": "http://localhost:8080/profile/passengers"
        }
    },
    "page": {
        "size": 20,
        "totalElements": 1,
        "totalPages": 1,
        "number": 0
    }
}
```



Step 10 (PassengerController):

Even though our RESTful web service is up and running, we need more control over the API without losing the advantage of auto-generated CRUD operations.

Before implementing our own controller, let's hide the PassengerRepository from the public.

We need to replace the @RepositoryRestResource(path = "/users") annotation with @RestResource(exported = false) annotation. This annotation informs Spring Boot not to export the methods as REST endpoints.

```
@RestResource(exported = false)

public interface PassengerRepository extends JpaRepository<Passenger, String> {

}
```

Create a new class PassengerController inside the package ca.yourname.restservice.controller

```
package ca.sergei.restservice.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import ca.sergei.restservice.entity.Passenger;
import ca.sergei.restservice.repository.PassengerRepository;

import javassist.tools.web.BadHttpRequest;

@RestController
@RequestMapping(path = "/passengers")
public class PassengerController {

	@Autowired
	private PassengerRepository repository;

	@GetMapping
	public Iterable<Passenger> findAll() {
		return repository.findAll();
	}

	@GetMapping(path = "/{login}")
	public Passenger find(@PathVariable("login") String login) {
		return repository.findOne(login);
	}

	@PostMapping(consumes = "application/json")
	public Passenger create(@RequestBody Passenger passenger) {
		return repository.save(passenger);
```

```
	}

	@DeleteMapping(path = "/{login}")
	public void delete(@PathVariable("login") String login) {
		repository.delete(login);
	}

	@PutMapping(path = "/{login}")
	public Passenger update(@PathVariable("login") String login, @RequestBody Passenger passenger) throws BadHttpRequest {
		if (repository.exists(login)) {
			passenger.setLogin(login);
			return repository.save(passenger);
		} else {
			throw new BadHttpRequest();
		}
	}
}
```

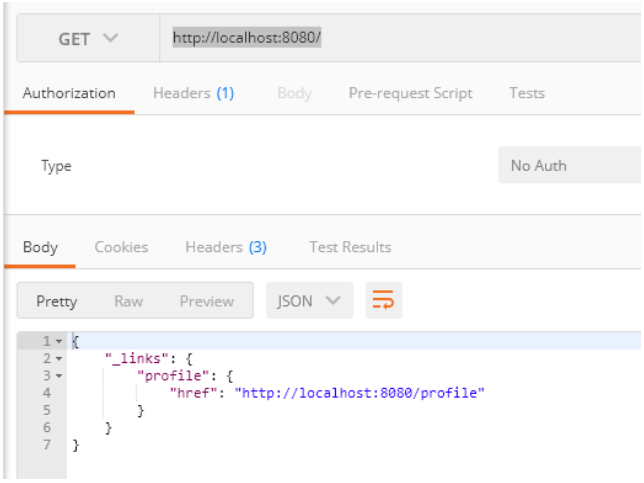
Step 11 (Full test):

Right click on the project and select Maven -> Update Project…
Terminate all projects if they are running
After, right click on the project and select Run As → Java Application. If asked, select the Application class as the main class to execute.

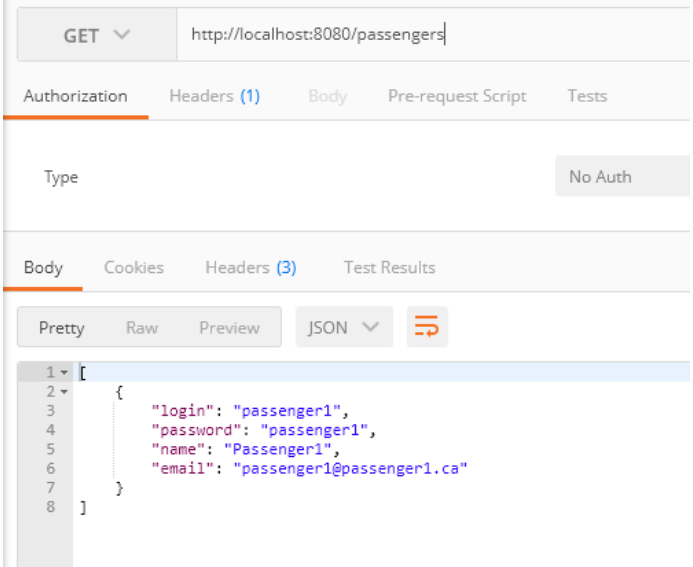
At the console log you must see:

```
2019-09-08 20:39:58.048  WARN 9668 --- [          main] .c.m.RepositoryCollectionResourceMapping : @RestResource detected to customize the repository resource for ca.sergei.restservice.repository.PassengerRepository!
2019-09-08 20:39:58.266  INFO 9668 --- [          main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationConte
2019-09-08 20:39:58.316  INFO 9668 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/passengers/{login}],methods=[PUT]}" onto public ca.sergei.restservice.entity.Passenger ca.sergei.restservi
2019-09-08 20:39:58.317  INFO 9668 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/passengers/{login}],methods=[GET]}" onto public ca.sergei.restservice.entity.Passenger ca.sergei.restservi
2019-09-08 20:39:58.317  INFO 9668 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/passengers/{login}],methods=[DELETE]}" onto public void ca.sergei.restservice.controller.PassengerControll
2019-09-08 20:39:58.317  INFO 9668 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/passengers],methods=[POST],consumes=[application/json]}" onto public ca.sergei.restservice.entity.Passenge
2019-09-08 20:39:58.317  INFO 9668 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/passengers],methods=[GET]}" onto public java.lang.Iterable<ca.sergei.restservice.entity.Passenger> ca.serg
2019-09-08 20:39:58.318  INFO 9668 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Obj
2019-09-08 20:39:58.318  INFO 9668 --- [          main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView org.springframework
```

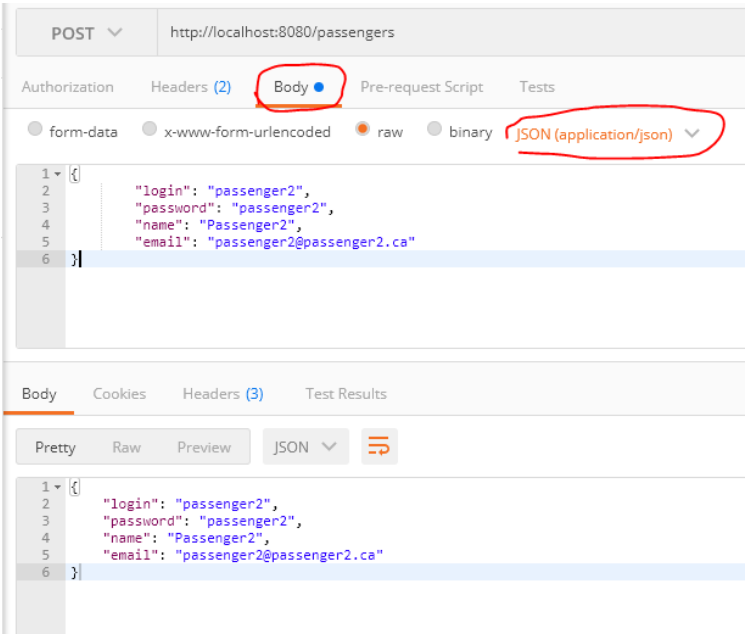
GET http://localhost:8080/



GET http://localhost:8080/passengers GET all list of passengers



POST http://localhost:8080/passengers create new passenger



GET http://localhost:8080/passengers GET all list of passengers

GET http://localhost:8080/passengers

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (3) Test Results

Pretty Raw Preview JSON


```
[
    {
        "login": "passenger1",
        "password": "passenger1",
        "name": "Passenger1",
        "email": "passenger1@passenger1.ca"
    },
    {
        "login": "passenger2",
        "password": "passenger2",
        "name": "Passenger2",
        "email": "passenger2@passenger2.ca"
    }
]
```


PUT http://localhost:8080/passengers/passenger2 update passenger2

PUT http://localhost:8080/passengers/passenger2

Authorization Headers (2) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)


```
{
        "password": "test",
        "name": "Test",
        "email": "test@test.ca"
}
```


Body Cookies Headers (3) Test Results

Pretty Raw Preview JSON


```
{
    "login": "passenger2",
    "password": "test",
    "name": "Test",
    "email": "test@test.ca"
}
```


GET info about passenger2 http://localhost:8080/passengers/passenger2

GET http://localhost:8080/passengers/passenger2

Authorization Headers (2) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (3) Test Results

Pretty Raw Preview JSON


```
{
    "login": "passenger2",
    "password": "test",
    "name": "Test",
    "email": "test@test.ca"
}
```


DELETE passenger2 http://localhost:8080/passengers/passenger2

DELETE http://localhost:8080/passengers/passenger2

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (2) Test Results

Pretty Raw Preview Text

GET all list of passengers http://localhost:8080/passengers

GET http://localhost:8080/passengers

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (3) Test Results

Pretty Raw Preview JSON


```
[
    {
        "login": "passenger1",
        "password": "passenger1",
        "name": "Passenger1",
        "email": "passenger1@passenger1.ca"
    }
]
```

Step 12 (Adding the basic authentication):

Adding the basic authentication to the REST API using the database.
Add the following dependency to the pom.xml file.

```
<!-- Spring Security -->
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Create a new class SecurityConfig in the ca.yourname.restservice package.

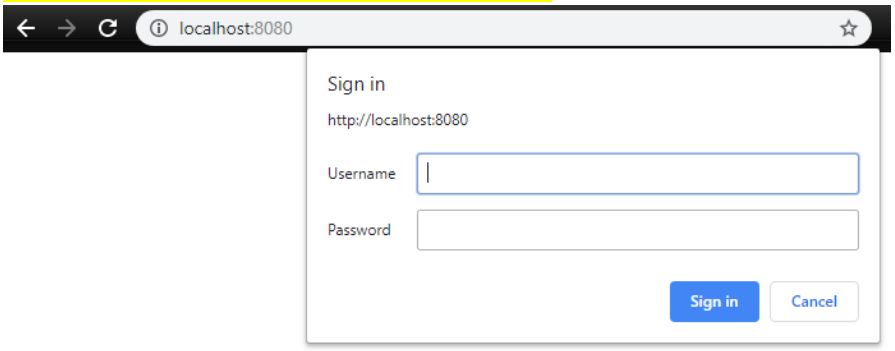
```
package ca.sergei.restservice;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.userdetails.UsernameNotFoundException;

import ca.sergei.restservice.entity.Passenger;
import ca.sergei.restservice.repository.PassengerRepository;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

	@Autowired
	private PassengerRepository userRepository;

	@Autowired
	protected void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
		auth.userDetailsService(username -> {
			Passenger user = userRepository.findOne(username);
			if (user != null) {
				return new org.springframework.security.core.userdetails.User(user.getLogin(), user.getPassword(),
						true, true, true, true, AuthorityUtils.createAuthorityList("USER"));
			} else {
				throw new UsernameNotFoundException("Could not find the user '" + username + "'");
			}
		});
	}

	@Override
	protected void configure(HttpSecurity http) throws Exception {
		http.authorizeRequests().anyRequest().fullyAuthenticated().and().httpBasic().and().csrf().disable();
	}
}
```

This class contains the Autowired PassengerRepository to retrieve User objects using the login and add a UserDetailsService using the login and password. As the authority list, you can return any user roles. If you have user roles defined in your database, it is better to use them. Otherwise, you can hardcode any user roles as I have defined USER here.
The configure method configures the authentication process. It enables basic authentication to all requests and disables the CSRF (Cross Site Request Forgery) header check since it will cause 403 Forbidden HTTP error when sending any requests other than GET.

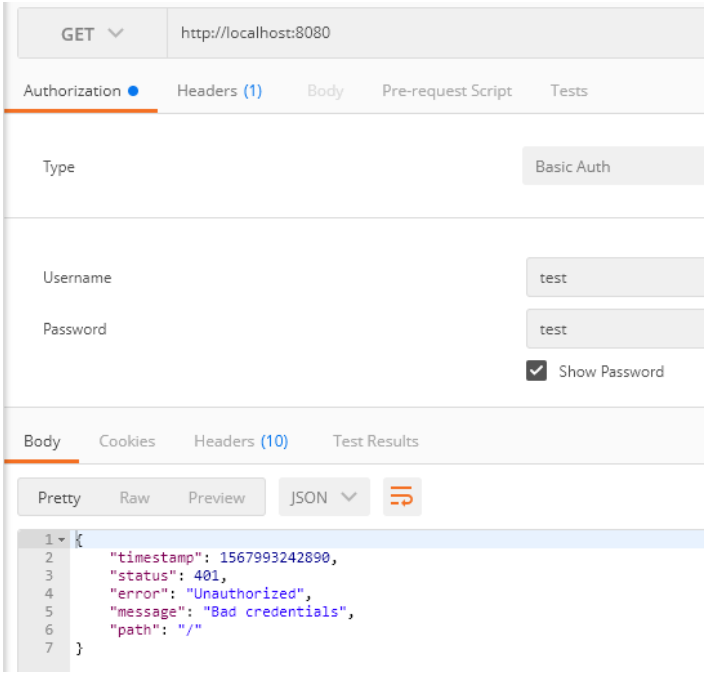
Step 13 (Authentication test):

Update maven project.
Stop web server, run project as java application.

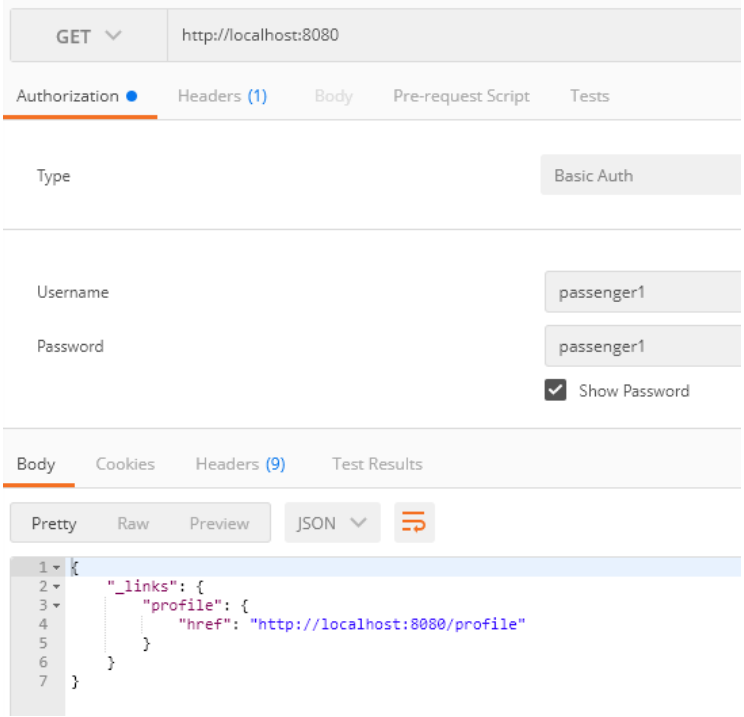
Now, if try to run localhost:8080, you can see:



You will get 401 without / wrong authentication:



Now type correct login / pass:



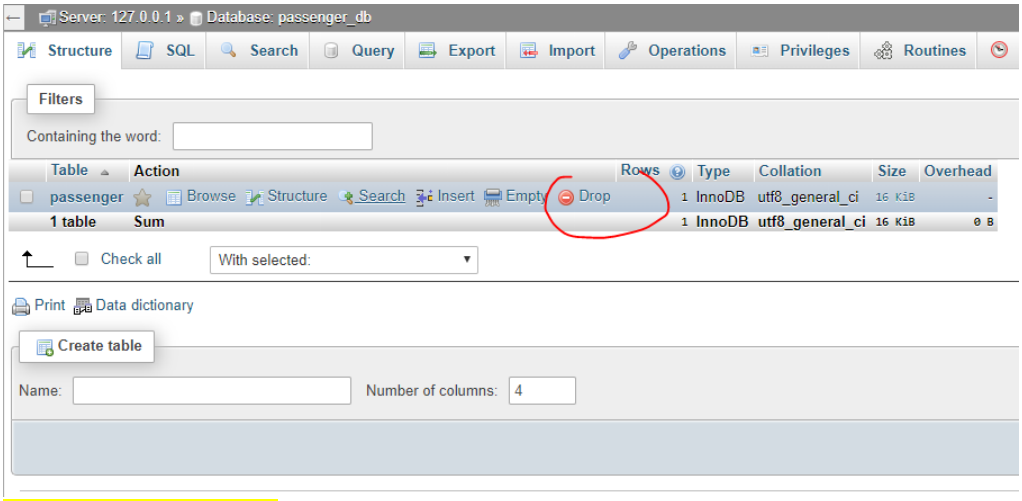
Step 14 (database prefill):

Let's add CommandLineRunner init method to the Application.java to create a db table (passenger) at the system startup.

```
package ca.sergei.restservice;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import ca.sergei.restservice.entity.Passenger;
import ca.sergei.restservice.repository.PassengerRepository;

@SpringBootApplication
public class Application {
	public static void main(String[] args) {
		SpringApplication.run(Application.class, args);
	}
	@Bean
	protected CommandLineRunner init(final PassengerRepository passengerRepository) {
		return args -> {
			Passenger passenger = new Passenger();
			passenger.setLogin("passenger1");
			passenger.setPassword("passenger1");
			passenger.setName("passenger_name1");
			passenger.setEmail("passenger1@passenger.com");
			passengerRepository.save(passenger);
		};
	}
}
```

Test it:

Drop table passenger:



Now we have clear DB.
Update maven project.
Stop web server, run project as java application.
Check your db. Passenger table must be created.

Step 15 (Creating relationships between tables):

JPA / Hibernate One to One Mapping [reference]
JPA / Hibernate One to Many Mapping [reference]
JPA / Hibernate Many to Many Mapping [reference]

A **one-to-one** relationship is defined using JPA's @OneToOne annotation. It accepts several attributes. Let's understand what those attributes are meant for -
fetch = FetchType.LAZY - Fetch the related entity lazily from the database.
cascade = CascadeType.ALL - Apply all cascading effects to the related entity. That is, whenever we update/delete a AAA entity, update/delete the corresponding BBB as well.
mappedBy = "AAA" - We use mappedBy attribute in the AAA entity to tell hibernate that the AAA entity is not responsible for this relationship and It should look for a field named ttt in the BBB entity to find the configuration for the JoinColumn/ForeignKey column.
In a bi-directional relationship, we specify @OneToOne annotation on both the entities but only one entity is the owner of the relationship. Most often, the child entity is the owner of the relationship and the parent entity is the inverse side of the relationship.
The owner of the relationship contains a @JoinColumn annotation to specify the foreign key column, and the inverse-side of the relationship contains a mappedBy attribute to indicate that the relationship is mapped by the other entity.

The idea with bidirectional one-to-many association is to allow you to keep a collection of child entities in the parent, and enable you to persist and retrieve the child entities via the parent entity. This is made possible via Hibernate's entity state transitions and dirty checking mechanism.

We use @ManyToMany annotation to create a many-to-many relationship between two entities. In a bi-directional association, the @ManyToMany annotation is used on both the entities but only one entity can be the owner of the relationship.
The entity that specifies the @JoinTable is the owning side of the relationship and the entity that specifies the mappedBy attribute is the inverse side.

ca.yourname.restservice
ca.yourname.restservice.controller
ca.yourname.restservice.entity
ca.yourname.restservice.repository

Let's create new entities in the ca.yourname.restservice.entity:
Address, Flight and Payment.
But before, We need to update dependency.

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
		xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
		<modelVersion>4.0.0</modelVersion>
		<groupId>com.javahelps</groupId>
		<artifactId>mysql-rest-service</artifactId>
		<version>0.0.1-SNAPSHOT</version>
			<parent>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-starter-parent</artifactId>
				<version>2.0.5.RELEASE</version>
				<relativePath/> <!-- lookup parent from repository -->
			</parent>
			<properties>
				<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
				<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
				<java.version>1.8</java.version>
			</properties>

			<dependencies>
			  <!-- Tomcat embedded container-->
			  <dependency>
			    <groupId>org.springframework.boot</groupId>
			      <artifactId>spring-boot-starter-tomcat</artifactId>
```

```
            <scope>provided</scope>
        </dependency>
        <!-- JSTL for JSP -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
        </dependency>
        <!-- Need this to compile JSP -->
        <dependency>
            <groupId>org.apache.tomcat.embed</groupId>
            <artifactId>tomcat-embed-jasper</artifactId>
            <scope>provided</scope>
        </dependency>
        <!-- Need this to compile JSP,
            tomcat-embed-jasper version is not working, no idea why -->
        <dependency>
            <groupId>org.eclipse.jdt.core.compiler</groupId>
            <artifactId>ecj</artifactId>
            <version>4.6.1</version>
            <scope>provided</scope>
        </dependency>
        <!-- Optional, test for static content, bootstrap CSS-->
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>bootstrap</artifactId>
            <version>3.3.7</version>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-devtools</artifactId>
                <scope>runtime</scope>
        </dependency>
        <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
                <scope>runtime</scope>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
        </dependency>
        <!-- Data REST -->
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-rest</artifactId>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Address:

```
package ca.sergei.restservice.entity;

import java.util.Set;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(name = "address")
public class Address {
    private long id;
    private String number;
    private String streetAddress;
    private String city;
    private String state;
```

```
private String country;
private String zipcode;
private Set<Passenger> passenger ;

public Address() {
		super();
}

public Address(long id, String number, String streetAddress, String city, String state, String country,
				String zipcode, Set<Passenger> passenger) {
		super();
		this.id = id;
		this.number = number;
		this.streetAddress = streetAddress;
		this.city = city;
		this.state = state;
		this.country = country;
		this.zipcode = zipcode;
		this.passenger = passenger;
}
@JsonIgnore
@OneToMany(mappedBy="address",cascade = CascadeType.ALL)
public Set<Passenger> getPassenger() {
		return passenger;
}
public void setPassenger(Set<Passenger> passenger) {
		this.passenger = passenger;
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
public long getId() {
		return id;
}
public void setId(long id) {
		this.id = id;
}
@Column(name = "number", nullable = false)
public String getNumber() {
		return number;
}
public void setNumber(String number) {
		this.number = number;
}
@Column(name = "street_address", nullable = false)
public String getStreetAddress() {
		return streetAddress;
}
public void setStreetAddress(String streetAddress) {
		this.streetAddress = streetAddress;
}
@Column(name = "city", nullable = false)
public String getCity() {
		return city;
}
public void setCity(String city) {
		this.city = city;
}
@Column(name = "state", nullable = false)
public String getState() {
		return state;
}
public void setState(String state) {
		this.state = state;
}
@Column(name = "country", nullable = false)
public String getCountry() {
		return country;
}
public void setCountry(String country) {
		this.country = country;
}
@Column(name = "zip_code", nullable = false)
```

```
	public String getZipcode() {
		return zipcode;
	}
	public void setZipcode(String zipcode) {
		this.zipcode = zipcode;
	}
}
```

Flight:

```
package ca.sergei.restservice.entity;

import java.util.Date;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

import org.hibernate.annotations.OnDelete;
import org.hibernate.annotations.OnDeleteAction;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(name = "flight")
public class Flight {

	private long id;
	private String fromCity;
	private Date departureDate;
	private String departureTime;
	private String toCity;
	private Date destinationDate;
	private String destinationTime;
	private Set<Passenger> passengers;

	public Flight() {
	}

	public Flight(long id, String fromCity, Date departureDate, String departureTime, String toCity,
		Date destinationDate, String destinationTime, Set<Passenger> passengers) {
		super();
		this.id = id;
		this.fromCity = fromCity;
		this.departureDate = departureDate;
		this.departureTime = departureTime;
		this.toCity = toCity;
		this.destinationDate = destinationDate;
		this.destinationTime = destinationTime;
		this.passengers = passengers;
	}

	@JsonFormat(pattern="HH:mm:ss")
	@Column(name = "departure_time", nullable = false)
	public String getDepartureTime() {
		return departureTime;
	}
	public void setDepartureTime(String departureTime) {
		this.departureTime = departureTime;
	}
	@JsonFormat(pattern="HH:mm:ss")
	@Column(name = "destination_time", nullable = false)
	public String getDestinationTime() {
		return destinationTime;
	}
```

```
	public void setDestinationTime(String destinationTime) {
		this.destinationTime = destinationTime;
	}
	@Column(name = "from_city")
	public String getFromCity() {
		return fromCity;
	}
	public void setFromCity(String fromCity) {
		this.fromCity = fromCity;
	}
	@Column(name = "to_city")
	public String getToCity() {
		return toCity;
	}
	public void setToCity(String toCity) {
		this.toCity = toCity;
	}

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	@JsonFormat(pattern="yyyy-MM-dd")
	@Column(name = "departure_date", nullable = false)
	public Date getDepartureDate() {
		return departureDate;
	}
	public void setDepartureDate(Date departureDate) {
		this.departureDate = departureDate;
	}
	@JsonFormat(pattern="yyyy-MM-dd")
	@Column(name = "destination_date", nullable = false)
	public Date getDestinationDate() {
		return destinationDate;
	}
	public void setDestinationDate(Date destinationDate) {
		this.destinationDate = destinationDate;
	}
	@JsonIgnore
	 @ManyToMany(fetch = FetchType.LAZY,
		mappedBy = "flightList",  cascade = {CascadeType.ALL})
	@OnDelete(action = OnDeleteAction.CASCADE)
	public Set<Passenger> getPassengers() {
		return passengers;
	}
	public void setPassengers(Set<Passenger> passengers) {
		this.passengers = passengers;
	}
}
```

Payment:

```
package ca.sergei.restservice.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

enum CardType {
	Visa, MasterCard
}
@Entity
@Table(name = "payment")
public class Payment {
	private long id;
	private String cardNumber;
```

```
	private String cardType;

	public Payment() {
	}

	public Payment(long id, String cardNumber, String cardType) {
		super();
		this.id = id;
		this.cardNumber = cardNumber;
		this.cardType = cardType;
	}
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	@Column(name = "card_number", nullable = false)
	public String getCardNumber() {
		return cardNumber;
	}
	public void setCardNumber(String cardNumber) {
		this.cardNumber = cardNumber;
	}
	@Column(name = "card_type", nullable = false)
	public String getCardType() {
		return cardType;
	}
	public void setCardType(String cardType) {
		this.cardType = cardType;
	}
}
```

And we need to change our class Passenger

Passenger:

```
package ca.sergei.restservice.entity;

import java.util.Date;
import java.util.Set;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import org.hibernate.annotations.OnDelete;
import org.hibernate.annotations.OnDeleteAction;
import com.fasterxml.jackson.annotation.JsonFormat;

@Entity
@Table(name = "passenger")

public class Passenger {
	private long id;
	private String login;
	private String password;
	private String name;
	private String family;
	private Date birthdate;
	private String email;
	private String phone;
	private Payment payment;
```

```
private Address address;
private Set<Flight> flightList;
private String gender;

public Passenger() {
}
public Passenger(long id, String login, String password, String name, String family, Date birthdate, String email, String phone, Payment payment,
                Address address, Set<Flight> flightList, String gender) {
        super();
        this.id = id;
        this.login = login;
        this.password = password;
        this.name = name;
        this.family = family;
        this.birthdate = birthdate;
        this.email = email;
        this.phone = phone;
        this.payment = payment;
        this.address = address;
        this.flightList = flightList;
        this.gender = gender;
}
@Column(name = "gender", nullable = false)
public String getGender() {
        return gender;
}
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
public long getId() {
        return id;
}
public void setId(long id) {
        this.id = id;
}
public void setGender(String gender) {
        this.gender = gender;
}
@ManyToOne(cascade = CascadeType.ALL)
@JoinColumn(name = "address_id", nullable = false)
public Address getAddress() {
        return address;
}
public void setAddress(Address address) {
        this.address = address;
}
@Column(name = "login", nullable = false)
public String getLogin() {
        return login;
}
public void setLogin(String login) {
        this.login = login;
}
@Column(name = "password", nullable = false)
public String getPassword() {
        return password;
}
public void setPassword(String password) {
        this.password = password;
}
@Column(name = "name", nullable = false)
public String getName() {
        return name;
}
public void setName(String name) {
        this.name = name;
}
@Column(name = "family", nullable = false)
public String getFamily() {
        return family;
}
public void setFamily(String family) {
        this.family = family;
}
```

```
	@JsonFormat(pattern = "yyyy-MM-dd")
	@Column(name = "birthdate", nullable = false)
	public Date getBirthdate() {
		return birthdate;
	}
	public void setBirthdate(Date birthdate) {
		this.birthdate = birthdate;
	}
	@Column(name = "email", nullable = false)
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	@Column(name = "phone", nullable = false)
	public String getPhone() {
		return phone;
	}
	public void setPhone(String phone) {
		this.phone = phone;
	}
	@OneToOne(cascade = CascadeType.ALL)
	@JoinColumn(unique = true)
	public Payment getPayment() {
		return payment;
	}
	public void setPayment(Payment payment) {
		this.payment = payment;
	}
	@OnDelete(action = OnDeleteAction.CASCADE)
	@ManyToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
	@JoinTable(name = "passenger_flight", joinColumns = {
		@JoinColumn(name = "passenger_id", referencedColumnName = "id") }, inverseJoinColumns = {
		@JoinColumn(name = "flight_id", referencedColumnName = "id") })
	public Set<Flight> getFlightList() {
		return flightList;
	}
	public void setFlightList(Set<Flight> flightList) {
		this.flightList = flightList;
	}
}
```

PassengerRepository:

```
package ca.sergei.restservice.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import org.springframework.web.bind.annotation.RequestMapping;
import ca.sergei.restservice.entity.Passenger;

@Repository
//@RequestMapping(path = "/passengers")
public interface PassengerRepository extends JpaRepository<Passenger, Long>{
}
```

FlightRepository:

```
package ca.sergei.restservice.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import org.springframework.web.bind.annotation.RequestMapping;

import ca.sergei.restservice.entity.Flight;

@Repository
//@RequestMapping(path = "/flights")
public interface FlightRepository extends JpaRepository<Flight, Long>{
```

```
}
```

AddressRepository:

```
package ca.sergei.restservice.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import org.springframework.web.bind.annotation.RequestMapping;
import ca.sergei.restservice.entity.Address;
@Repository
//@RequestMapping(path = "/address")
public interface AddressRepository extends JpaRepository<Address, Long>{
}
```

Now we need to change our Application class, for renovate first init of data to our database.

Application :

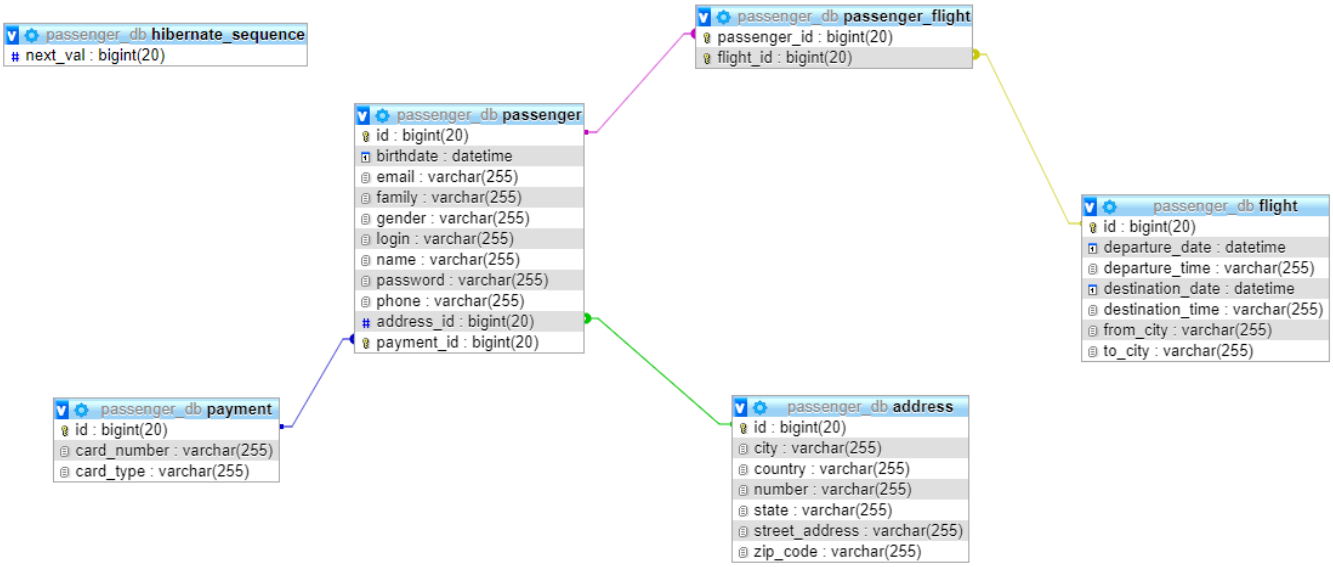
```
package ca.sergei.restservice;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import ca.sergei.restservice.entity.Passenger;
import ca.sergei.restservice.repository.PassengerRepository;
import ca.sergei.restservice.entity.MyEnum;
import ca.sergei.restservice.entity.Address;
import ca.sergei.restservice.entity.Flight;
import ca.sergei.restservice.repository.AddressRepository;
import ca.sergei.restservice.repository.FlightRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import java.util.Calendar;
import java.util.Date;

@SpringBootApplication
public class Application  {

	public static void main(String[] args) {
		SpringApplication.run(Application.class, args);
	}
	@Bean
	protected CommandLineRunner init(final PassengerRepository passengerRepository) {
		return args -> {
	    Passenger passenger = new Passenger (Long.valueOf("1"), "admin", "admin", "AdminName", "AdminFamily", new Date(), "admin@email.ca",
"514-111-1111", null,
				new Address(1, "1A", "1e Avenue", "Montreal", "QC", "Canada", "A1A1A1", null), null, "M");
	    passengerRepository.save(passenger);
		};
	}
}
```

Step 16 (Test relationships):

Drop all tables in the Database.

As we haven't finished PassengerController yet, we can test program by adding @RequestMapping(path = "/passengers") annotation to the PassengerRepository

Update maven project.
Stop web server, run project as java application.
Check your db. All tables and relationships must be created.



Server: 127.0.0.1 » Database: passenger_db » Table: passenger

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking | Triggers

Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

```
SELECT * FROM `passenger`
```

Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

	id	birthdate	email	family	gender	login	name	password	phone	address_id	payment_id
Edit Copy Delete	2	2019-09-11 22:47:42	admin@email.ca	AdminFamily	M	admin	AdminName	admin	514-111-1111	1	NULL

Check all | With selected: Edit Copy Delete Export

Server: 127.0.0.1 » Database: passenger_db » Table: address

Browse | Structure | SQL | Search | Insert | Export | Import | Priv

Showing rows 0 - 0 (1 total, Query took 0.0005 seconds.)

```
SELECT * FROM `address`
```

Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

	id	city	country	number	state	street_address	zip_code
Edit Copy Delete	1	Montreal	Canada	1A	QC	1e Avenue	A1A1A1

Check all | With selected: Edit Copy Delete Export

Now we can add all necessary data to the tables:

Server: 127.0.0.1 » Database: passenger_db

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Routines | Events

Filters

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
address	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_general_ci	16 KiB	-
flight	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_general_ci	16 KiB	-
hibernate_sequence	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8_general_ci	16 KiB	-
passenger	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_general_ci	48 KiB	-
passenger_flight	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_general_ci	32 KiB	-
payment	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_general_ci	16 KiB	-
6 tables	Sum	9	InnoDB	utf8_general_ci	144 KiB	0 B

Let's run it in the browser:

localhost:8080/passengers


```
// 20190911230547
// http://localhost:8080/passengers

[
  {
    "id": 2,
    "login": "admin",
    "password": "admin",
    "name": "AdminName",
    "family": "AdminFamily",
    "birthdate": "2019-09-12",
    "email": "admin@email.ca",
    "phone": "514-111-1111",
    "payment": null,
    "address": {
      "id": 1,
      "number": "1A",
      "streetAddress": "1e Avenue",
      "city": "Montreal",
      "state": "QC",
      "country": "Canada",
      "zipcode": "A1A1A1"
    },
    "flightList": [
      {
        "id": 1,
        "fromCity": "Montreal",
        "departureDate": "2019-09-12",
        "departureTime": "02:25",
        "toCity": "Vancouver",
        "destinationDate": "2019-09-13",
        "destinationTime": "03:30"
      }
    ],
    "gender": "M"
  }
]
```


Step 17 (Changing PassengerController):

PassengerController:

```
package ca.sergei.restservice.controller;

import java.util.HashMap;
import java.util.Map;
```

```
import java.util.Optional;
import javax.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import ca.sergei.restservice.entity.Address;
import ca.sergei.restservice.entity.Passenger;
import ca.sergei.restservice.repository.AddressRepository;
import ca.sergei.restservice.repository.PassengerRepository;
import javassist.tools.web.BadHttpRequest;

@RestController
@RequestMapping(path = "/passengers")
public class PassengerController {
	@Autowired
	private PassengerRepository repository;
	@GetMapping
	public Iterable<Passenger> findAll() {
		return repository.findAll();
	}
	@GetMapping(path = "/{id}")
	public Optional<Passenger> find(@PathVariable("id") Long id) {
		return repository.findById(id);
	}
	@PostMapping(consumes = "application/json")
	public Passenger create(@Valid @RequestBody Passenger passenger) {
		return repository.save(passenger);
	}
	@PutMapping(path = "/{id}")
	public Passenger update(@PathVariable("id") Long id, @RequestBody Passenger passenger) throws BadHttpRequest {
		if (repository.existsById(id)) {
			passenger.setId(id);
			return repository.save(passenger);
		} else {
			throw new BadHttpRequest();
		}
	}
	@DeleteMapping(path = "/{id}")
	public void delete(@PathVariable("id") Long id) throws BadHttpRequest {
		if (repository.existsById(id)) {
			repository.deleteById(id);
		} else {
			throw new BadHttpRequest();
		}
	}
}
```

Step 18 (Final test):

GET to http://localhost:8080/ - OK

GET http://localhost:8080/

Authorization | Headers (1) | Body | Pre-request Script | Tests

Type: No Auth

Body | Cookies | Headers (3) | Test Results

Pretty | Raw | Preview | JSON


```
{
    "_links": {
        "passengers": {
            "href": "http://localhost:8080/passengers{?page,size,sort}",
            "templated": true
        },
        "flights": {
            "href": "http://localhost:8080/flights{?page,size,sort}",
            "templated": true
        },
        "addresses": {
            "href": "http://localhost:8080/addresses{?page,size,sort}",
            "templated": true
        },
        "profile": {
            "href": "http://localhost:8080/profile"
        }
    }
}
```


GET to the http://localhost:8080/passengers - OK

GET http://localhost:8080/passengers

Pretty | Raw | Preview | JSON


```
[
    {
        "id": 2,
        "login": "admin",
        "password": "admin",
        "name": "AdminName",
        "family": "AdminFamily",
        "birthdate": "2019-09-12",
        "email": "admin@email.ca",
        "phone": "514-111-1111",
        "payment": {
            "id": 1,
            "cardNumber": "AA1234-2345-3456",
            "cardType": "VISA"
        },
        "address": {
            "id": 1,
            "number": "1A",
            "streetAddress": "1e Avenue",
            "city": "Montreal",
            "state": "QC",
            "country": "Canada",
            "zipcode": "A1A1A1"
        },
        "flightList": [
            {
                "id": 1,
                "fromCity": "Montreal",
                "departureDate": "2019-09-12",
                "departureTime": "02:25",
                "toCity": "Vancouver",
                "destinationDate": "2019-09-13",
                "destinationTime": "03:30"
            }
        ],
        "gender": "M"
    }
]
```

POST to the http://localhost:8080/passengers - OK

POST http://localhost:8080/passengers

Authorization | Headers (2) | Body | Pre-request Script | Tests

form-data | x-www-form-urlencoded | raw | binary | JSON (application/json)


```
{
    "id": 3,
    "login": "user1",
    "password": "user1",
    "name": "user1Name",
    "family": "user1Family",
    "birthdate": "2019-09-12",
    "email": "user1@email.ca",
    "phone": "514-111-1111",
    "payment": {
        "id": 2,
        "cardNumber": "bb123456789",
        "cardType": "VISA"
    },
    "address": {
        "id": 1,
        "number": "1A",
        "streetAddress": "1e Avenue",
        "city": "Montreal",
        "state": "QC",
        "country": "Canada",
        "zipcode": "A1A1A1"
    },
    "flightList": [
        {
            "id": 1,
            "fromCity": "Montreal",
            "departureDate": "2019-09-12",
            "departureTime": "02:25",
```


Body | Cookies | Headers (3) | Test Results

Pretty | Raw | Preview | JSON


```
{
    "id": 4,
    "login": "user1",
    "password": "user1",
    "name": "user1Name",
    "family": "user1Family",
    "birthdate": "2019-09-12",
    "email": "user1@email.ca",
    "phone": "514-111-1111",
    "payment": {
        "id": 2,
        "cardNumber": "bb123456789",
        "cardType": "VISA"
    },
    "address": {
        "id": 1,
```


Server: 127.0.0.1 » Database: passenger_db » Table: passenger

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking | Triggers

Showing rows 0 - 1 (2 total, Query took 0.0005 seconds.)

SELECT * FROM `passenger`

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

	id	birthdate	email	family	gender	login	name	password	phone	address_id	payment_id
Edit Copy Delete	2	2019-09-11 22:47:42	admin@email.ca	AdminFamily	M	admin	AdminName	admin	514-111-1111	1	1
Edit Copy Delete	4	2019-09-11 20:00:00	user1@email.ca	user1Family	M	user1	user1Name	user1	514-111-1111	1	2

Check all | With selected: Edit Copy Delete Export

PUT to the http://localhost:8080/passengers/4 - OK

PUT http://localhost:8080/passengers/4

Authorization Headers (2) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)


```
{
    "login": "blabla",
    "password": "blabla",
    "name": "blablaName",
    "family": "blablaFamily",
    "birthdate": "2019-09-12",
    "email": "blabla@email.ca",
    "phone": "514-111-1111",
    "payment": {
        "id": 2,
        "cardNumber": "bb123456789",
        "cardType": "VISA"
    },
    "address": {
        "id": 1,
        "number": "1A",
        "streetAddress": "1e Avenue",
        "city": "Montreal",
        "state": "QC",
        "country": "Canada",
        "zipcode": "A1A1A1"
    },
    "flightList": [
        {
            "id": 1,
            "fromCity": "Montreal",
            "departureDate": "2019-09-12",
            "departureTime": "02:25",
            "toCity": "Vancouver",
```


Body Cookies Headers (3) Test Results

Pretty Raw Preview JSON


```
{
    "id": 4,
    "login": "blabla",
    "password": "blabla",
    "name": "blablaName",
    "family": "blablaFamily",
    "birthdate": "2019-09-12",
    "email": "blabla@email.ca",
    "phone": "514-111-1111",
    "payment": {
        "id": 2,
        "cardNumber": "bb123456789",
        "cardType": "VISA"
    },
    "address": {
```


Server: 127.0.0.1 » Database: passenger_db » Table: passenger

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Showing rows 0 - 1 (2 total, Query took 0.0005 seconds.)

```
SELECT * FROM `passenger`
```

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

		id	birthdate	email	family	gender	login	name	password	phone	address_id	payment_id
Edit Copy Delete		2	2019-09-11 22:47:42	admin@email.ca	AdminFamily	M	admin	AdminName	admin	514-111-1111	1	1
Edit Copy Delete		4	2019-09-11 20:00:00	blabla@email.ca	blablaFamily	M	blabla	blablaName	blabla	514-111-1111	1	2

Check all With selected: Edit Copy Delete Export

DELETE to the http://localhost:8080/passengers/4 - OK

DELETE http://localhost:8080/passengers/4

Authorization Headers (2) Body Pre-request Script Tests

1

Server: 127.0.0.1 » Database: passenger_db » Table: passenger

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Trig

Showing rows 0 - 0 (1 total, Query took 0.0005 seconds.)

```
SELECT * FROM `passenger`
```

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	id	birthdate	email	family	gender	login	name	password	phone	address_id
Edit Copy Delete	6	2019-09-11 23:35:30	admin@email.ca	AdminFamily	M	admin	AdminName	admin	514-111-1111	5

Check all *With selected:* Edit Copy Delete Export