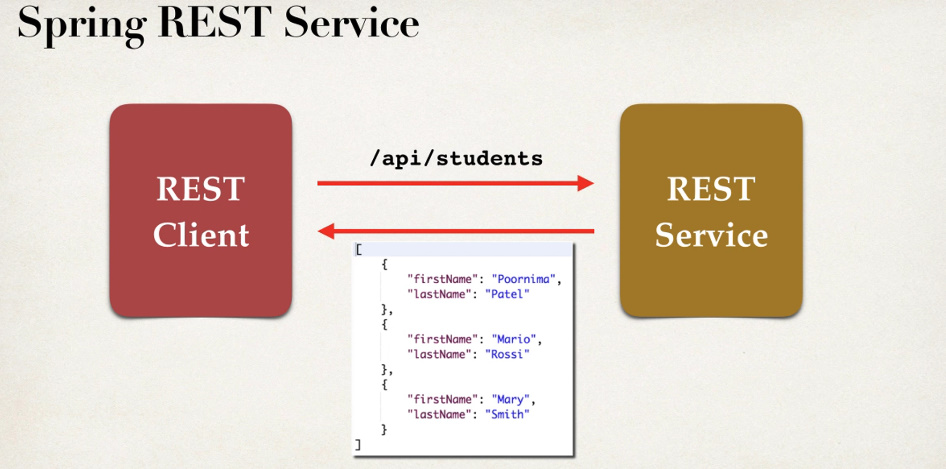
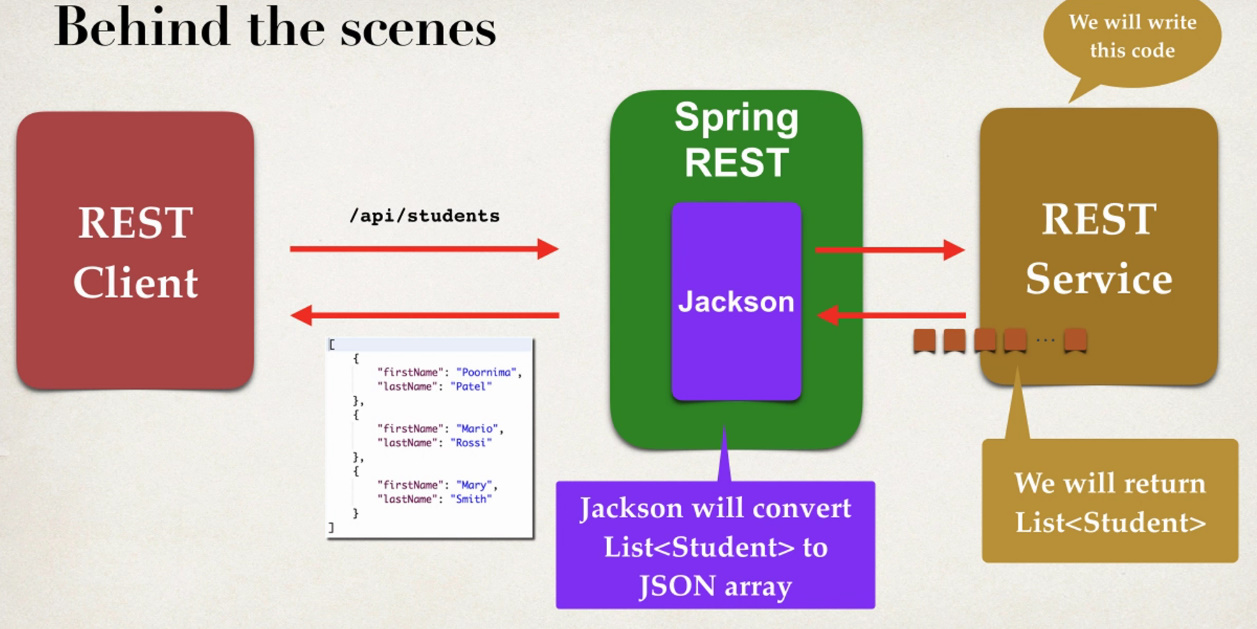
* Rest Service inseamna ca folosind diferite request, obtinem diferite date.



Asa, obtinem toate datele in format JSON

* Deci, REST Service va returna o lista
* Jackson, cand transforma POJO in JSON, foloseste getters methods din obiect si cand din JSON in POJO, setters

**Cum functioneaza un REST API**



* Deci, Spring REST si Jackson lucreaza mereu impreuna
* Spring REST ia requesturile si le da pe mana lui Jackson.
* Jackson e cel care ia lista de obiecte de la REST SERVICE si o transforma apoi in JSON si invers.

**@RequestMapping vs @GetMapping**

**RequestMapping** – Anotatie care mapeaza HTTP requests de metode specifice. E folosit pentru a administra requesturi generice unde putem folosi GET, PUT, POST. Ea este mai mult folosita la nivel de clasa.

**GetMapping** – anotatie care mapeaza HTTP GET request de metode. E prescurtare pentru RequestMapping(method = RequestMethod.GET). Este folosita doar la nivel de metoda.

**Rest Service Creation**

1. Clasa Student

public class Student {  
 private String firstName;  
 private String lastName;  
  
 public Student() { }  
  
 public Student(String firstName, String lastName) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
}

1. RestController:

@RestController  
@RequestMapping("/api")  
public class Controller {  
  
 @GetMapping("/students")  
 public List<Student> getStudents(){  
 List<Student> students = new ArrayList<Student>();  
 students.add(new Student("Mititiuc","Eduard"));  
 students.add(new Student("Goncear","Alex"));  
 students.add(new Student("Petru","Turcan"));  
  
 return students;  
 }  
}

Cand vom da request /api/students, se va afisa o lista de studenti in JSON. Remarcam ca noi nu am transformat nimic in JSON cu ObjectMapper, dar am returnat doar lista. Jackson si Spring Rest s-au ocupat de transformarea obiectelor din lista in JSON

**@PathVariable**

* @PathVariable este utilizat cand dorim sa primim in forma de JSON doar un singur obiect, in dependenta de un parametru, ca id cum ar fi. De ex:

.../api/students/{id} --> ../api/students/5



* De acum vom face asa:

@GetMapping(“/students/{studentId}”)

* Trebuie sa legam @PathVariable de un parametru local din metoda,cu numele identic cu cel dintre {}.

@GetMapping("/students/{studentId}")  
public Student getStudents(@PathVariable int studentId){  
 List<Student> students = new ArrayList<Student>();  
 students.add(new Student("Mititiuc","Eduard"));  
 students.add(new Student("Goncear","Alex"));  
 students.add(new Student("Petru","Turcan"));  
  
 return students.get(studentId);  
}



**@PathVariable poate fi folosita si pentru un simplu @Controller**

**@PostConstruct**

* **Controller tot este un bean!!!**
* **Putem avea doar un singur @PostConstruct per class!**
* Mai sus, nu e deloc bine ca de fiecare data cand metoda getStudents e apelata, se creaza mereu o lista si se tot insereaza in ea studentii.
* Pentru a rezolva asta, folosim @PostConstruct pentru a genera datele o singura data. Cand Controllerul va fi creat, la rularea programului, tot ce are in fata @PostConstruct va fi automat apelat, si deci o singura data.
* Pentru ca @PostConstruct sa lucreze, adaugam in pom.xml:

<dependency>  
 <groupId>javax.annotation</groupId>  
 <artifactId>javax.annotation-api</artifactId>  
 <version>1.3.2</version>  
</dependency>

@RestController  
@RequestMapping("/api")  
public class Controller {  
   
 private List<Student> students;  
   
 @PostConstruct  
 public void getStudents(){  
 students = new ArrayList<Student>();  
 students.add(new Student("Mititiuc","Eduard"));  
 students.add(new Student("Goncear","Alex"));  
 students.add(new Student("Petru","Turcan"));  
 }  
 @GetMapping("/students/{studentId}")  
 public Student getStudents(@PathVariable int studentId){  
  
 return students.get(studentId);  
 }  
}