REST

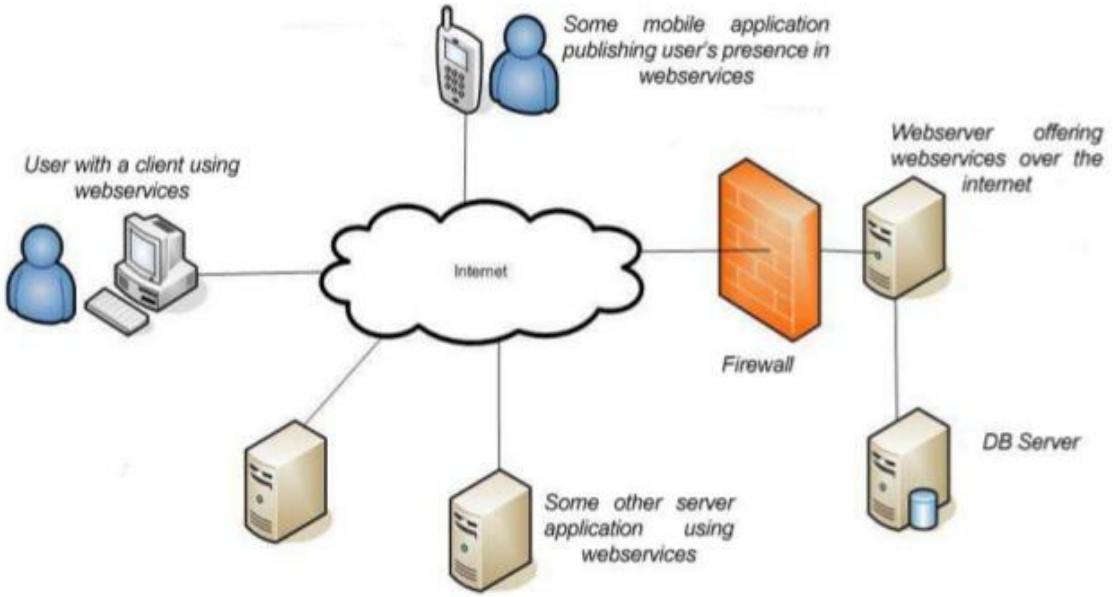
Голуб Виталий
Java-developer

2017

О чем пойдет речь

- Web services
- REST & RESTful
- REST vs SOAP

Need for web services



- **Web service (Веб-сервис, веб-служба)** - сервис (набор методов), предоставляемый приложением и доступный для использования по сети.
- Стандартизированный способ для взаимодействия разнородных приложений
- Без ограничений на ОС, язык программирования или устройство
- Предоставление услуг для любого приложения

Что же такое REST?

- Технологию REST впервые представил один из авторов HTTP-протокола **Рой Филдинг** (Roy Fielding) в своей диссертации в Калифорнийском университете в Ирвайне (2000 год).

- REST (Representational state transfer) – это стиль архитектуры программного обеспечения для распределенных систем.

- REST - НЕ протокол

- REST является простым интерфейсом управления информацией

- Системы, поддерживающие REST, называются RESTful-системами

Принципы проектирования RESTful Web-сервисов

- Явное использование HTTP-методов.
- Взаимодействие между сервером и клиентом не хранит состояние (stateless).
- Предоставление URI, аналогичных структуре каталогов.
- Передача данных в XML, JavaScript Object Notation (JSON).

Более подробно про REST
http://www.restapitutorial.ru/lessons/whatisrest.html#code-on-demand

REST : HTTP Methods

SQL	REST(HTTP)
SELECT	GET
INSERT	POST
UPDATE	PUT
DELETE	DELETE

REST : URI Design

URI	HTTP-метод	Действие
/student/	GET	получить список всех студентов
/student/4	GET	получить студента с id=4
/student/4	POST	изменить студента (данные в теле запроса)
/student/	PUT	добавить студента (данные в теле запроса)
/student/4	DELETE	удалить студента

REST vs SOAP

SOAP веб-сервисы

- XML, WSDL
- Работа с методами
- Поддержка транзакций, уровней безопасности и пр.
- Большое кол-во спецификаций
- Различные транспортные уровни
- Сложнее в разработке

Java™ API for XML Web Services (JAX-WS)

RESTful веб-сервисы

- Ресурс ориентированная технология
- HTTP запросы
- Для несложной бизнес-модели
- Работа с ресурсами, а не методами
- Легче разрабатывать
- Сложнее в разработке

JAX-RS: Java™ API for RESTful Web Services

REST в Spring

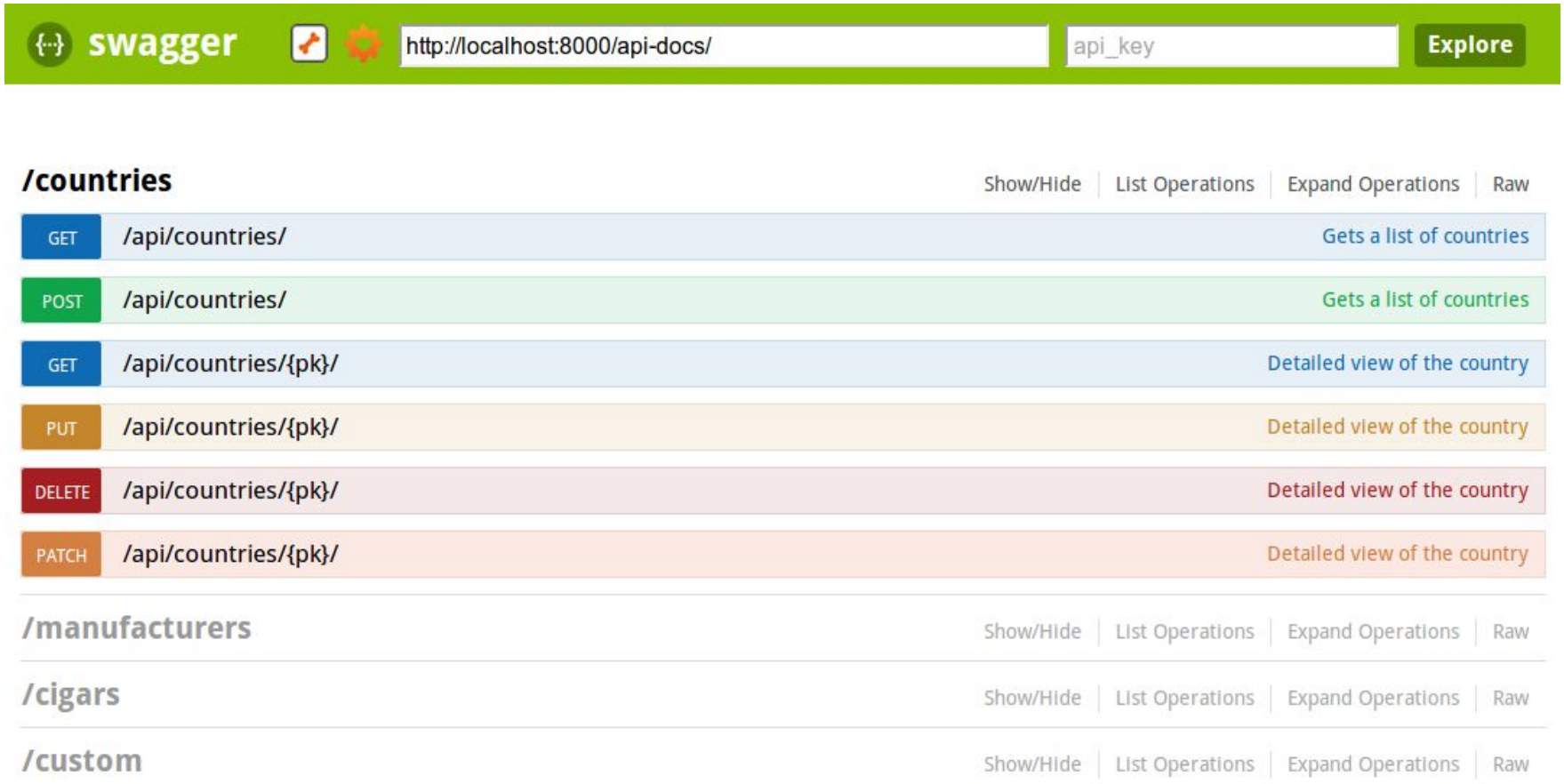
- @RestController
- @RequestMapping
- @ResponseBody
- @RequestParam
- @PathVariable

```
@RestController
@RequestMapping(value = "/student")
public class StudentController {
     @RequestMapping(method = RequestMethod.POST)
     public String saveStudent(@RequestParam String name, @RequestParam Integer age) {
          // сохранить пользователя в бд
          return "Student created";
     }

    @RequestMapping(value = "/{id}",  method = RequestMethod.GET)
    @ResponseBody
    public Person getStudent(@PathVariable Long id) {
          // найти студента по id в бд
         // создать объект Person и вернуть
        return person;
     }

}
```

Rest-документация (Swagger)



Подключение Swagger в SpringBoot

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.4.0</version>
</dependency>

<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.4.0</version>
</dependency>
```

Создать бин конфигурации для сваггера (рядом с бином запуска @SpringBootApplication)

```
@Configuration
@EnableSwagger2
public class SwaggerConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
          .select()
          .apis(RequestHandlerSelectors.any())
          .paths(PathSelectors.any())
          .build();
    }
}
```

Практическое задание

1. Развернуть Spring Boot проект из предыдущего занятия
2. Создать/добавить в PersonController 5 методов:
   - savePerson()
   - getParson()
   - updatePerson()
   - deletePerson()
   - getPeople()
3. Указать в @RequestMapping для каждого метода
   - соответствующий HTTP-метод
   - value = "/{id}" для методов, требующих id и добавить @PathVariable к параметру long id
4. Заполнить таблицу 3-4 записями с помощью resources/data.sql
5. Запустить проект, создать/удалить/изменить пользователя, а также получить одного пользователя по id, и список всех пользователей, с помощью соответствующих REST-запросов.

   Дополнительно
6. Добавить зависимости для сваггера
7. Создать конфиг-бин для сваггера
8. Запустить проект, перейти на *localhost:8080/swagger-ui.html*
9. Повторить 5 пункт теперь уже со сваггером