Bài 20
Web service và RESTful

Module: BOOTCAMP WEB-BACKEND DEVELOPMENT

Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài “Session và Cookie"

Tóm tắt lại các phần đã học từ bài “Session và Cookie”

Mục tiêu

- Trình bày được khái niệm Web service
- Trình bày được các loại web service cơ bản
- Mô tả được RESTful Web service
- Triển khai được RESTful Web service
- Sử dụng được POSTMAN để kiểm thử web service
- Trình bày được lớp RestTemplate
- Sử dụng được RestTemplate để thao tác với RESTful Web service

Thảo luận

Web service

Web Service (1)

- Web service (Dịch vụ web) là các thành phần ứng dụng được hiển thị dưới dạng các dịch vụ trên WWW.
- Dịch vụ
  - Là một thành phần phần mền
  - Chứa một vàu logic nghiệp vụ bên trong đó
  - Được hiển thị trên Web cho nhiều loại client
  - Được truy cập bởi các client tại các vị trí khác nhau
- Web Service có thể sử dụng để tích hợp với các ứng dụng được viết bằng các ngôn ngữ khác nhau và chạy trên các nền tảng khác nhau.
- Xây dựng các chuẩn mở và sử dụng các giao thức mở để giao tiếp.

Web Service (2)

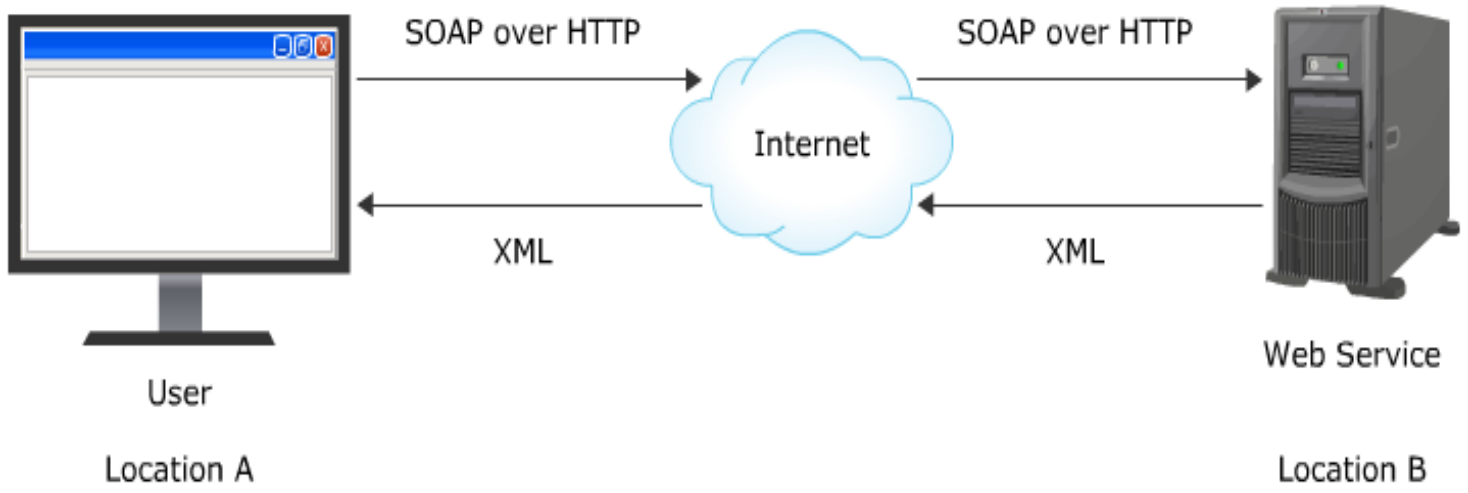
- Web service hoạt động như một server trong mô hình ứng dụng client server sử dụng giao thức HTTP/HTTPs và chỉ thực hiện một tác vụ cụ thể.
- Với dữ liệu đầu vào xác định, web server xử lý và trả ra dữ liệu đầu ra theo chuẩn đảm bảo mọi ứng dụng có thể hiểu và sử dụng mà không quan tâm đến loại thiết bị, hệ điều hành, kiến trúc phần mềm hay ngôn ngữ được sử dụng.
- Kiểu dữ liệu đầu ra phổ biến của một web service thường là XML hoặc JSON.

Các thành phần của Web Service

3 thành phần chính (dựa trên XML)

1. Simple Object Access Protocol (SOAP): dùng để trao đổi thông tin\thông điệp giữa user và Web Service
2. Web Service Description Language (WSDL): dùng để mô tả thông tin về Web Service
3. Universal Description, Discovery and Integration (UDDI): dùng cho việc công bố Web Service, user sử dụng UDDI để xác định\khám phá các nghiệp vụ mà Web Service cung cấp và triệu gọi chúng

Giao tiếp giữa user và Web Service



1. User gọi Web Service bằng cách gửi (request) một thông điệp SOAP qua giao thức HTTP tới Web Service
2. Web Service tiếp nhận\xử lí và trả về (response) cho user một thông điệp SOAP trên nền tảng XML

Phân biệt Website và Web Service (1)

Website	Web Service
Có giao diện người dùng hoặc GUI	Không có giao diện người dùng
Nói đến Website được hiểu sử dụng bởi con người	Nói đến Web Service được hiểu sử dụng bởi các ứng dụng được tương tác với nhau qua internet
Website hoạt động đa nền tảng, vì chúng yêu cầu tinh chỉnh để hoạt động trên các trình duyệt hay hệ điều hành khác nhau.	Web Service độc lập về nền tảng và tất cả dạng truyền thông đều sử dụng giao thức chuẩn.
Website được truy cập bởi các thành phần trong giao diện người dùng như button, textbox, form ...	Webservice được truy cập bởi phương thức HTTP – PUT, GET, POST, DELETE ...

Phân biệt Website và Web Service (2)

Website	Web Service
Ví dụ. ArtOfTesting.com là trang web có bộ sưu tập các trang web có liên quan chứa hướng dẫn	Ví dụ: Google Maps API là một dịch vụ web có thể được sử dụng bởi các trang web để hiển thị bản đồ bằng cách chuyển tọa độ tới đó.
Website là một ứng dụng đầu cuối. Người sử dụng truy cập website qua URL qua đó nhận được những dữ liệu text, hình ảnh, âm thanh… có thể dễ dàng hiểu được.	Web service là một khái niệm rộng hơn, dữ liệu trả ra từ web service người sử dụng thông thường khó để hiểu. Dữ liệu đó được các ứng dụng (trong đó có web site) sử dụng và chế biến thành dữ liệu có thể đọc cho người sử dụng.

Ưu điểm của web service

- Hoạt động trên các ứng dụng, nền tảng, hệ điều hành, ngôn ngữ khác nhau.
- Khả năng tái sử dụng cao.
- Tạo mối quan hệ tương tác, mềm dẻo trong hệ thống phần mềm, dễ dàng cho việc phát triển ứng dụng phân tán.
- Giảm sự phức tạp của hệ thống, giảm thời gian phát triển hệ thống, hạ giá thành hoạt động, dễ dàng tương tác giữa các hệ thống với nhau.

Nhược điểm của web service

- Khi một web service chết hoặc dừng hoạt động sẽ gây lỗi, thiệt hại lớn trên tất cả các hệ thống, thiết bị đang sử dụng web service đó.
- Cần quan tâm đến vấn đề an toàn và bảo mật nhiều hơn khi sử dụng web service.
- Việc có quá nhiều chuẩn cho web service dẫn đến người sử dụng khó nắm bắt.

Các loại web service cơ bản

- SOAP (Simple Object Access Protocol) là giao thức sử dụng XML để định nghĩa dữ liệu dạng thuần văn bản (plain text) và truyền dữ liệu thông qua HTTP.
- REST (Representational State Transfer) là một kiểu cấu trúc cung cấp các quy tắc để xây dựng web service.
  - REST định nghĩa dữ liệu dưới dạng XML hoặc JSON và truyền thông qua mạng internet sử dụng giao thức HTTP.
  - Các web service xây dựng dựa trên REST được gọi là RESTful, chúng chủ yếu nhằm xử lý các hoạt động CRUD (Create/ Read/ Update/ Delete) trên dữ liệu.

Thảo luận

RESTful web service

REST

- REST (Representational State Transfer) được sử dụng để tạo ra và giao tiếp với Web Service thay thế mô hình SOAP.
- REST có kiến trúc đón giản, định rõ các ràng buộc nhằm tạo ra ứng dụng Web Service đạt được những tính chất mong muốn về hiệu suất, khả năng mở rộng, khả năng điều chỉnh v.v..
- REST hướng tới việc xây dựng ứng dụng Web Service có khả năng làm việc tốt nhất trên môi trường WWW.
- Dữ liệu và các tính năng được coi như tài nguyên và được truy xuất thông qua các URI (Uniform Resource Identifier)
- Sử dụng 4 phương thức chính của HTTP là POST, GET, PUT và DELETE để thực hiện các hành động CRUD đối với cá tài nguyên.

RESful Web Service (1)

- RESTful web service là một web service được xây dựng dựa trên cấu trúc REST.
- RESTful web service tuân thủ các quy tắc cơ bản của cấu trúc REST gồm:
  - Sử dụng các phương thức HTTP một cách rõ ràng
  - Phi trạng thái
  - Hiển thị cấu trúc thư mục như các URIs
  - Truyền tải **JavaScript Object Notation (JSON)**, **XML** hoặc cả hai.

RESful Web Service (2)

- Sử dụng các phương thức HTTP một cách rõ ràng
  - Để tạo một tài nguyên trên máy chủ, sử dụng phương thức **POST**.
  - Để truy xuất một tài nguyên, sử dụng phương thức **GET**.
  - Để thay đổi trạng thái một tài nguyên hoặc để cập nhật nó, sử dụng phương thức **PUT**.
  - Để huỷ bỏ hoặc xoá một tài nguyên, sử dụng phương thức **DELETE**.

- Chú ý: Những quy tắc nêu trên là không bắt buộc nhưng REST khuyến nghị sử dụng chúng để tạo sự rõ ràng, dễ hiểu.

RESful Web Service (3)

- Phi trạng thái (Stateless)
  - RESTfull web service không lưu lại thông tin, phiên làm việc của client.
  - Mỗi lần gọi RESTful web service, client phải cung cấp đầy đủ các thông tin đầu vào. Web service sẽ xử lý và trả ra dữ liệu mà không quan tâm tới các lần request trước đó từ client.
- Ưu điểm:
  - Các thành phần máy chủ phi trạng thái sẽ có thiết kế ít phức tạp hơn.
  - Dễ dàng viết và phân bổ client đến server thông qua cân bằng tải.
- Nhược điểm:
  - Tiêu tốn tài nguyên băng thông hơn, do mỗi lần gọi web service, client phải truyền đầy đủ các thông tin.
  - Server thực thi web service phải hoạt động nhiều hơn do mỗi yêu cầu từ client đều phải xử lý như nhau, không tận dụng được tài nguyên, tính toán trước đó.

RESful Web Service (5)

- Hiển thị cấu trúc thư mục như các URIs
  - REST đưa ra một cấu trúc để người dùng có thể truy cập vào tài nguyên của nó thông qua các URL.
  - Các tài nguyên bao gồm text, hình ảnh, âm thanh, video… được xác định thông qua định danh như một địa chỉ URI (*Uniform Resource Identifier) đảm bảo* rõ ràng, dễ hiểu và dễ đoán cho người sử dụng.
- Ví dụ:
  - URL của một web service lấy thông tin thời tiết của Hà Nội vào ngày 5/6/2018: http://myservice.com/weather/hanoi/2018-06-05

RESful Web Service (6)

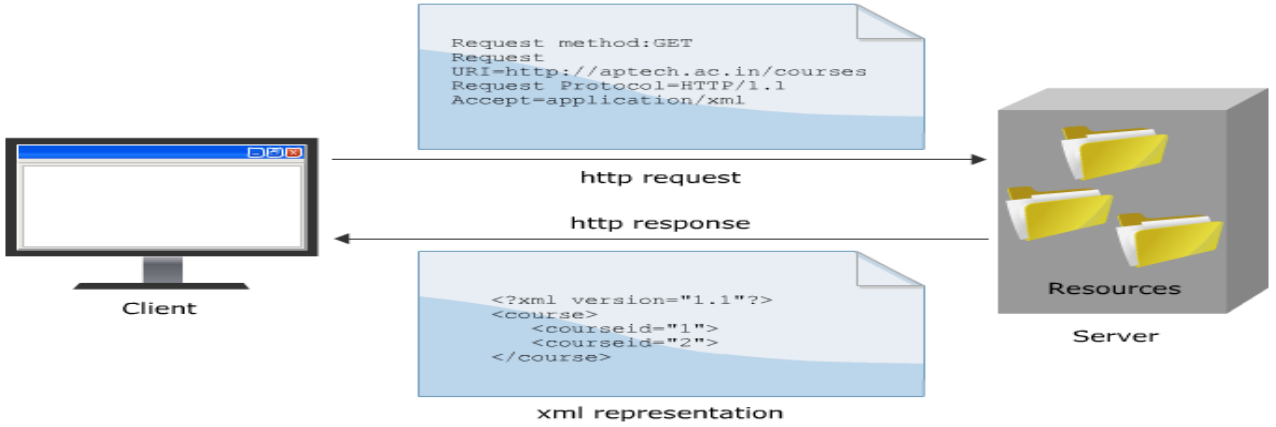
- Truyền tải **JavaScript Object Notation (JSON)**, **XML** hoặc cả hai.
  - Một client gửi yêu cầu đến web service được truyền tải dưới dạng XML hoặc JSON và dữ liệu trả về từ web service thông thường cũng tương tự.
  - Client có thể quyết định dữ liệu trả về từ web service bằng việc thêm chỉ định vào HEADER của request thông qua từ khóa Accept. Các chỉ định này được gọi là kiểu MINE.
- Ví dụ: Client chỉ định nhận dữ liệu trả về là kiểu JSON

  GET /weather/hanoi/2018-06-05 HTTP/1.1

  Host: myservice.com

  Accept: application/json

Giao tiếp User – Web Service



1. Client gửi (request) yêu cầu về một tài nguyên cụ thể nào đó tới Web Service sử dụng các phương thức của HTTP
2. Web Service xác định phương thức, tài nguyên và thực hiện các hành động CRUD tương ứng
3. Web Service gửi (response) về cho client tài nguyên theo định dạng mà client yêu cầu

Thảo luận

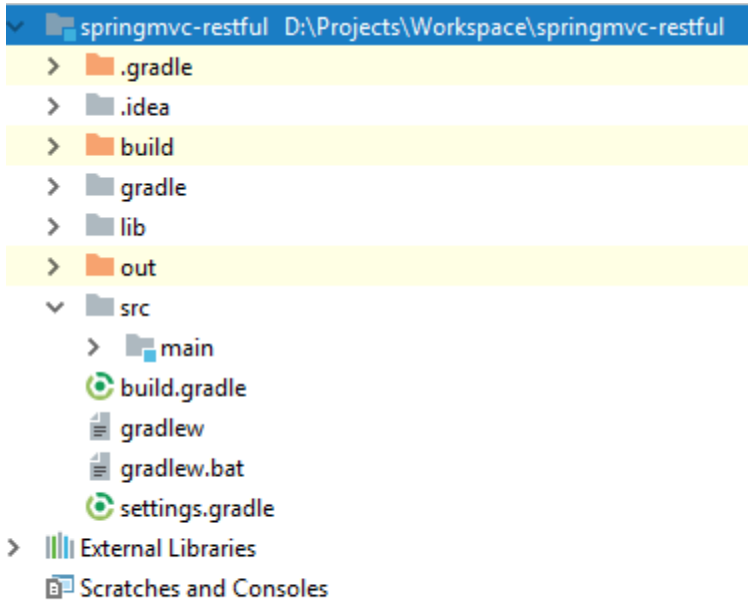
Triển khai RESTful

Mục tiêu

- Triển khai RESTful thực hiện thao tác CRUD cơ bản với một danh sách sinh viên với Spring MVC.
  - CREATE: Web service thực hiện thêm dữ liệu sinh viên mới vào danh sách - RESTful method POST.
  - READ: : Web service đọc dữ liệu sinh viên theo ID hoặc tất cả - RESTful method GET.
  - UPDATE: : Web service sửa dữ liệu sinh viên theo ID - RESTful method PUT.
  - DELETE: : Web service xóa sinh viên trong danh sách theo ID - RESTful method DELETE.

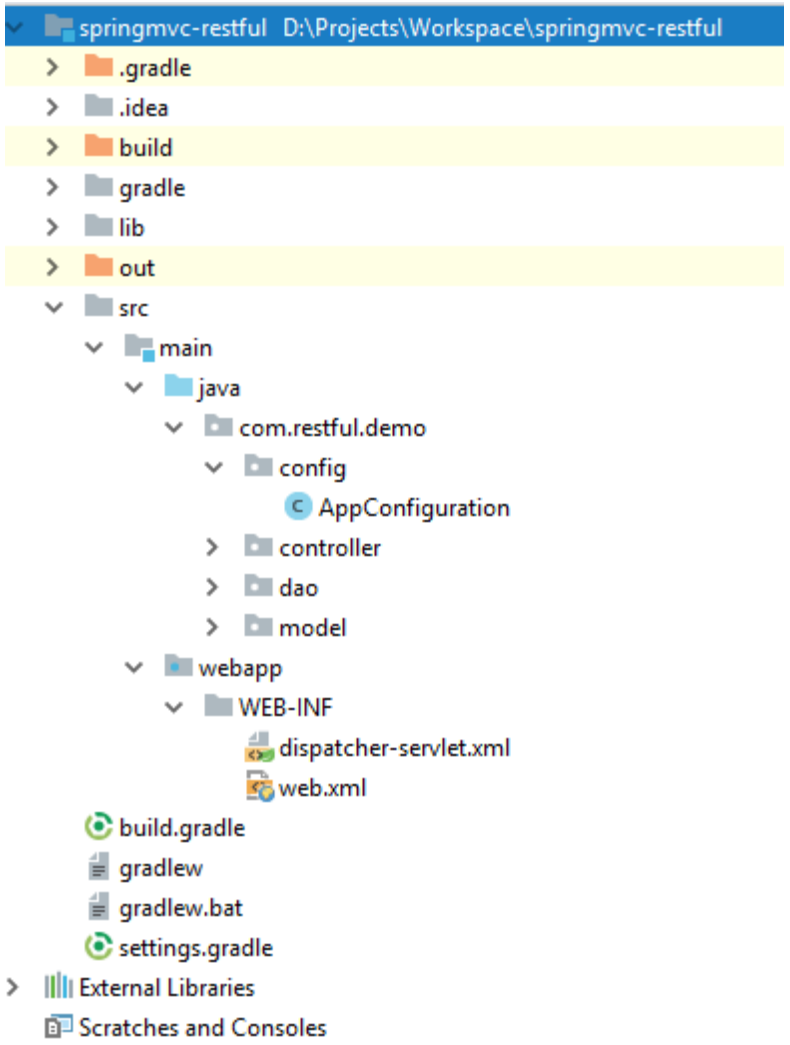
Triển khai RESTful

- Khởi tạo New project với Spring MVC.
- Cấu trúc ban đầu của project như sau:



Triển khai RESTful

- Cấu hình project Spring MVC (Đã học).
- Cấu trúc của project sau khi cấu hình.



Triển khai RESTful

- Cấu hình file ***build.gradle***
  - Thêm thư viện chuyển đổi XML ⇔ JAVA và JSON ⇔ JAVA
  - JSON ⇔ JAVA: Sử dụng **jackson-databind**
  - XML ⇔ JAVA: Sử dụng ***jackson-dataformat-xml***

```
compile group: 'com.fasterxml.jackson.core', name: 'jackson-databind', version: '2.8.3'
compile group: 'com.fasterxml.jackson.dataformat', name: 'jackson-dataformat-xml', version: '2.8.3'
```

Triển khai RESTful

- Tạo model
  - Tạo đối tượng Student với các thuộc tính cơ bản id, name, age, point cùng Getter/Setter và Contructor tương ứng.
  - Tạo interface ifStudentDAO định nghĩa các phương thức CRUD với một danh sách student.
  - Tạo đối tượng StudentDAO implement ifStudentDAO.

Triển khai RESTful

- Tạo REST Controller
  - Tạo class MainController, class MainController được xác định là RestController bằng anovation @RestController

```
1  package com.restful.demo.controller;
2
3  import com.restful.demo.dao.StudentDAO;
4  import com.restful.demo.model.Student;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.http.MediaType;
7  import org.springframework.web.bind.annotation.*;
8
9  import java.util.List;
10
11 @RestController
12 public class RESTfulmainController {
13     @Autowired
14     private StudentDAO studentDAO;
15
16
```

Triển khai RESTful

- Triển khai web service **get all students**
  - @RequestMapping
    - map giữa URL của web service và phương thức được thực thi.
    - Xác định phương thức của web service bằng từ khóa method
    - Xác định loại dữ liệu bằng từ khóa produces, trong ví dụ bên dưới, web service được xác định thao tác với cả 2 loại dữ liệu là XML và JSON.
  - @ResponseBody
    - Thực hiện logic của web service và trả ra dữ liệu cho client.

```
29
30      @RequestMapping(value = "/students",
31              method = RequestMethod.GET,
32              produces = {MediaType.APPLICATION_JSON_VALUE, MediaType.APPLICATION_XML_VALUE})
33      @ResponseBody
34      public List<Student> getAllStudent() { return studentDAO.findAllStudent(); }
37
```

Triển khai RESTful

- Triển khai web service **get student theo ID**
  - @RequestMapping xác định 1 parameter {id} sẽ được truyền vào URL của web service.
  - @ResponseBody lấy giá trị {id} trong URL truyền vào phương thức thực thi của web service.

```
21
22      @RequestMapping(value = "/student/{id}",
23              method = RequestMethod.GET,
24              produces = {MediaType.APPLICATION_JSON_VALUE, MediaType.APPLICATION_XML_VALUE})
25      @ResponseBody
26      public Student getStudent(@PathVariable String id) { return studentDAO.findStudent(id); }
29
```

Triển khai RESTful

- Triển khai web service **thêm mới 1 student**
  - @RequestMapping xác đinh phương thức của web service là POST tuân thủ theo cấu trúc REST.
  - Đối với phương thức POST web service xác định client cần truyền thêm dữ liệu vào nội dung của request. Kiểu dữ liệu có thể là XML, JSON…
  - @ResponseBody lấy thông tin đối tượng student trong @RequestBody đẩy vào phương thức thực thi của web service.

```
38      @RequestMapping(value = "/student",
39              method = RequestMethod.POST,
40              produces = {MediaType.APPLICATION_JSON_VALUE, MediaType.APPLICATION_XML_VALUE})
41      @ResponseBody
42      public String addStudent(@RequestBody Student st){
43          studentDAO.addStudent(st);
44          return "Successfullly added";
45      }
```

Triển khai RESTful

- Triển khai web service **xóa 1 student**
  - @RequestMapping xác đinh phương thức của web service là DELETE

```
56      @RequestMapping(value = "/student/{id}",
57              method = RequestMethod.DELETE,
58              produces = {MediaType.APPLICATION_JSON_VALUE, MediaType.APPLICATION_XML_VALUE})
59      @ResponseBody
60      public String updStudent(@PathVariable("id") String id){
61          studentDAO.delStudent(id);
62          return "Successfullly deleted";
63      }
```

Triển khai RESTful

- Triển khai web service **sửa dữ liệu student**
  - @RequestMapping xác định phương thức của web service là PUT

```
47      @RequestMapping(value = "/student",
48              method = RequestMethod.PUT,
49              produces = {MediaType.APPLICATION_JSON_VALUE, MediaType.APPLICATION_XML_VALUE})
50      @ResponseBody
51      public String updStudent(@RequestBody Student st){
52          studentDAO.updStudent(st);
53          return "Successfullly updated";
54      }
```

Demo

Triển khai RESTful

Thảo luận

Kiểm thử Web service

Kiểm thử Web service

- Khởi chạy project
  - Trang welcome được định nghĩa trong controller sẽ hiển thị đầu tiên, điều đó có nghĩa là server đã chạy thành công và web service đã sẵn sàng.
  - Bản chất trang welcome là 1 web service có URI rỗng, được xác định trong main controller thông qua anovation @RequestMapping(“/”)

```
16      @RequestMapping("/")
17      @ResponseBody
18      public String welcome() { return "Welcome to RESTful webservice"; }
```

Kiểm thử Web service

- Đối với các web service đơn giản sử dụng phương thức GET, có thể kiểm thử bằng cách chạy trực tiếp trên trình duyệt thông qua URL.
- Với web service get thông tin student URL như sau:
  - http://localhost:8080/students
  - http://localhost:8080/student/0001

Kiểm thử Web service

- Kết quả
  - http://localhost:8080/students

Kiểm thử Web service

- Kết quả
  - http://localhost:8080/student/0001

Kiểm thử Web service

- Với cách chạy trực tiếp qua trình duyệt sẽ gặp khó khăn trong việc custom RequestHeader, RequestBody và chỉ định kiểu dữ liệu trả về.
- Việc không custom được RequestHeader, RequestBody dẫn đến không thể kiểm thử nhưng web service sử dụng phương thức POST, PUT, DELETE…

Thảo luận

Sử dụng POSTMAN

Giới thiệu POSTMAN

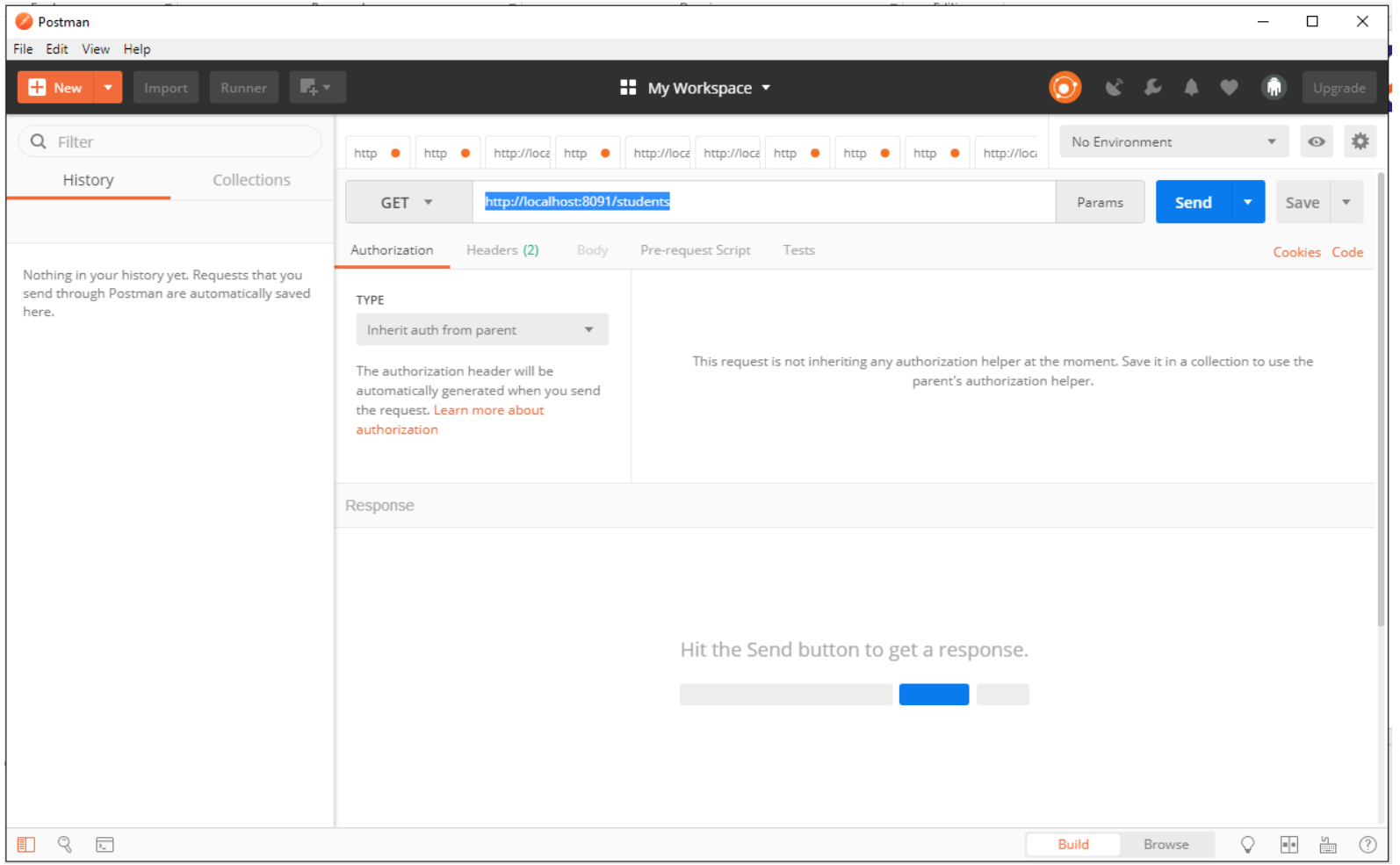
- POSTMAN là một App Extensions, cho phép làm việc với các API, điển hình là REST.
- Hỗ trợ tất cả các phương thức HTTP (GET, POST, PUT, DELETE, OPTIONS, HEAD ...)

Cài đặt POSTMAN

- POSTMAN được cung cấp miễn phí qua web site https://www.getpostman.com/

Sử dụng POSTMAN

- Giao diện chính



Sử dụng POSTMAN

- B1: Nhập URL của web service vào thanh địa chỉ của POSTMAN.
- B2: Chọn phương thức của web service.
- B3: Sửa, thêm các thông tin params phụ thuộc vào web service.
- B4: Nhấn SEND và theo dõi kết quả web service trả ra.

Sử dụng POSTMAN

- Với web service trên, khi kiểm thử với POSTMAN cho kết quả trả về với kiểu chỉ định JSON như sau:
- http://localhost:8080/students

Sử dụng POSTMAN

- Để kiểm thử web service sử dụng method POST, PUT…cần phải custom RequesHeader và RequestBody.
- Mục đích của việc custom là truyền thêm dữ liệu cần thiết cho web service.

Sử dụng POSTMAN

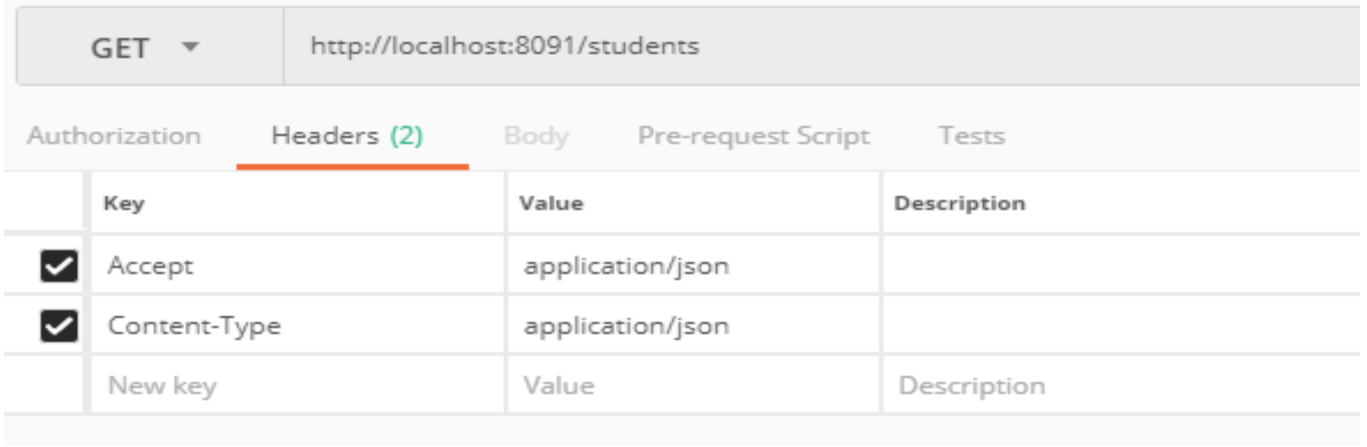
- Custom RequestHeader
  - Trên màn hình POSTMAN chuyển sang tab Header.
  - Để chỉ định kiểu dữ liệu trả về từ web service là JSON hoặc XML.

    Accept: Application/json

    Accept: Application/xml
  - Để chỉ định kiểu dữ liệu client gửi lên server là JSON, XML…

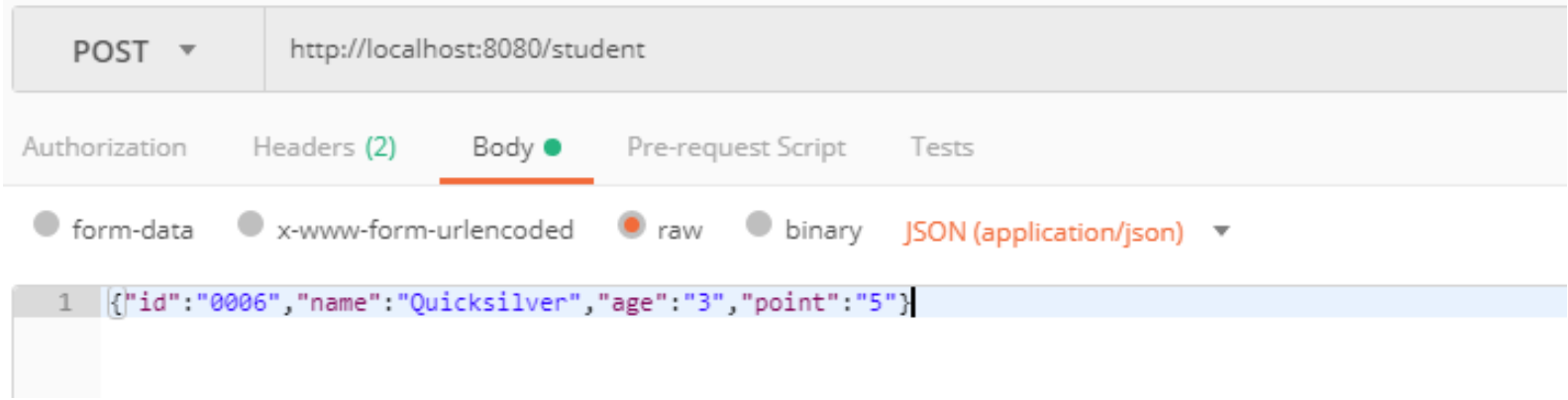
    Content-Type: Application/json

    Content-Type: Application/xml



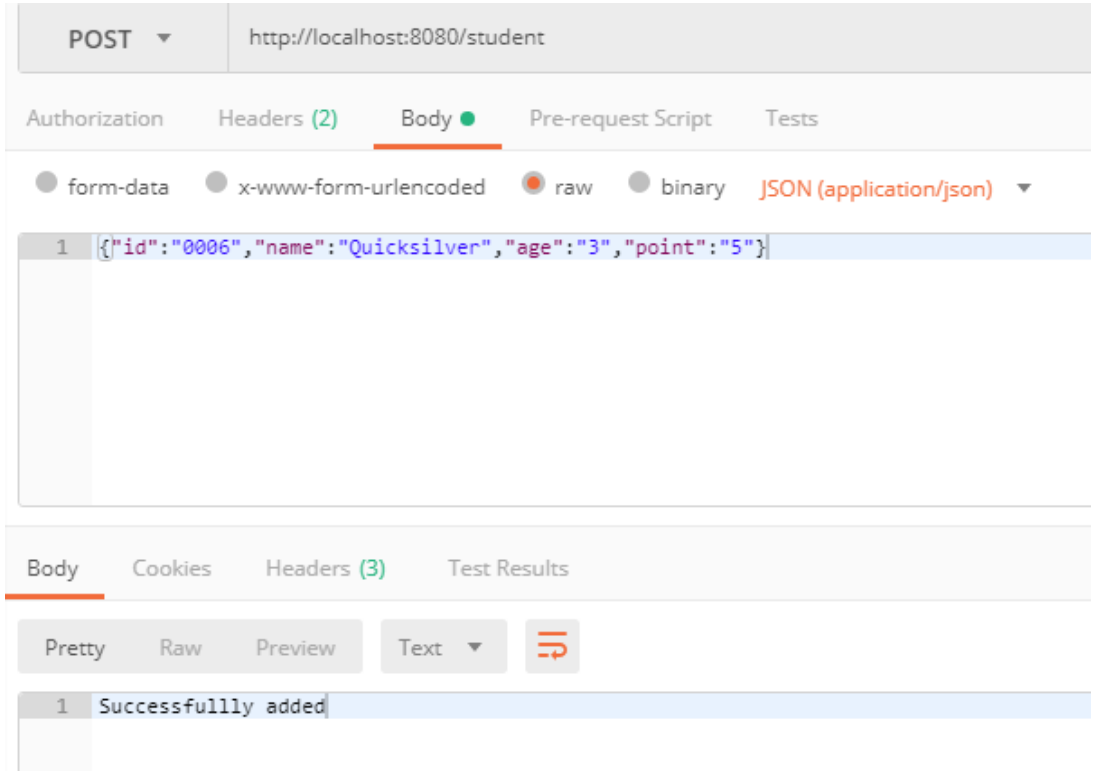
Sử dụng POSTMAN

- Custom RequestBody
  - Trên màn hình POSTMAN chuyển sang tab Body.
  - Xác định kiểu dữ liệu của request body, trong ví dụ là kiểu dữ liệu application/json. Nội dung của request body là chuỗi json của một đối tượng student.



Sử dụng POSTMAN

- Sau khi hoàn tất việc custom nhấn SEND để gọi web service thêm 1 đối tượng student mới.



- Message báo thành công, sử dụng web service getAllStudent để kiểm tra danh sách.

Demo

Sử dụng POSTMAN

Thảo luận

Class RestTemplate

RestTemplate

- RestTemplate là một class quan trọng của Spring Framework, giúp đồng bộ hóa những truy cập từ phía client vào RESTful web service trên server.
- RestTemplate đơn giản hóa việc giao tiếp với server bằng cách thực thi cấu trúc REST.
- RestTemplate xử lý các kết nối kiểu HTTP chỉ bằng các URLs và nhận lại kết quả.

RestTemplate

- RestTemplate contructors.

Tên phương thức	**Mô tả**
RestTemplate()	Tạo một thể hiện của RestTemplate dử dụng contructor mặc định.
RestTemplate(ClientHttpRequestFactory rqFactory)	Tạo 1 thể hiện của RestTemplate dựa trện 1 đối tượng ClientHttpRequestFactory.
RestTemplate(java.util.List<HttpMessageConverter<?>> msgConverter)	Tạo 1 thể hiện của RestTemplate sử dựa trên 1 danh sách HttpMessageConverter

RestTemplate

- RestTemplate cung cấp các phương thức tương ứng với phương thức của HTTP.

HTTP method	RestTemplate method
DELETE	delete(java.lang.String, java.lang.Object...)
GET	getForObject(java.lang.String, java.lang.Class<T>, java.lang.Object...)
	getForEntity(java.lang.String, java.lang.Class<T>, java.lang.Object...)
HEAD	headForHeaders(java.lang.String, java.lang.Object...)
OPTIONS	optionsForAllow(java.lang.String, java.lang.Object...)
POST	postForLocation(java.lang.String, java.lang.Object, java.lang.Object...)
	postForObject(java.lang.String, java.lan.Object, java.lang.Class<T>, java.lang.Object...)
PUT	put(java.lang.String, java.lang.Object, java.lang.Object...)

Triển khai RestTemplate

- Trong phần trước, đã triển khai được 5 web service thực hiện CRUD dữ liệu student như sau:
  - **GET** http://localhost:8080/student/{id}
  - **GET** http://localhost:8080/students
  - **POST** http://localhost:8080/student

    Content-Type: Application/json

    {"id":"0006","name":"Quicksilver","age":"3","point":"5"}

  - **PUT** http://localhost:8080/student

    Content-Type: Application/json

    {"id":"0006","name":"Quicksilver","age":"3","point":"5"}

  - **DELETE** http://localhost:8080/student/{id}

Triển khai RestTemplate

- Từ phía client sử dụng RestTemplate để thao tác với từng web service như sau:
  - **GET** http://localhost:8080/student/{id}

```
static final String URL_GET_ALL_STUDENT = "http://localhost:8080/students.json";
RestTemplate restTemplate = new RestTemplate();
String result = restTemplate.getForObject(URL_GET_STUDENT_BY_ID, String.class);
```

Triển khai RestTemplate

- Từ phía client sử dụng RestTemplate để thao tác với từng web service như sau:
  - **POST** http://localhost:8080/student

    Content-Type: Application/json

    {"id":"0006","name":"Quicksilver","age":"3","point":"5"}

```
final String URL_ADD_STUDENT = "http://localhost:8080/student";
Student st = new Student( id: "0006",  name: "Quicksilver",  age: 5,  point: 2);
HttpHeaders headers = new HttpHeaders();
headers.add( headerName: "Accept", MediaType.APPLICATION_JSON_VALUE);
  headers.setContentType(MediaType.TEXT_PLAIN);
RestTemplate restTemplate = new RestTemplate();
// Dữ liệu đính kèm theo yêu cầu.
HttpEntity<Student> requestBody = new HttpEntity<Student>(st, headers);
// Gửi yêu cầu với phương thức POST.
String result = restTemplate.postForObject(URL_ADD_STUDENT, requestBody, String.class);
System.out.println(result +  "\n");
```

Demo

Triển khai RestTemplate

Tóm tắt bài học

- Web service giúp cho việc phát triển ứng dụng, hệ thống trở lên dễ dàng và tiết kiệm thời gian và chi phí.
- RESTful web service là loại web service dựa trên cấu trúc REST đang rất phổ biết và phát triển hiện nay.
- RestTemplate là một class quan trọng của Spring Framwork, nó hỗ trợ phát triển các ứng dụng phía client. Giao tiếp với web service phía server.

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: I18N