[Bài đọc] Spring và các RESTful web services

REST là viết tắt của Representational State Transfer. Nó là một kiểu kiến ​​trúc có thể được sử dụng để thiết kế các dịch vụ web, có thể được sử dụng từ nhiều khách hàng khác nhau. Ý tưởng cốt lõi là, thay vì sử dụng các cơ chế phức tạp như CORBA, RPC hoặc SOAP để kết nối giữa các máy, HTTP đơn giản được sử dụng để thực hiện các cuộc gọi giữa chúng.

Các tài nguyên được xử lý bằng cách sử dụng một bộ động từ chung:

* Để tạo tài nguyên: sử dụng HTTP POST
* Để truy xuất tài nguyên: sử dụng HTTP GET
* Để cập nhật tài nguyên: sử dụng HTTP PUT
* Để xóa tài nguyên: sử dụng HTTP DELETE

Điều đó có nghĩa là, với tư cách là nhà phát triển dịch vụ REST, chúng ta nên tuân thủ các tiêu chí trên.

Các dịch vụ Web dựa trên REST thường trả về JSON hoặc XML dưới dạng phản hồi, mặc dù nó không chỉ giới hạn ở các loại này. Khách hàng có thể chỉ định (sử dụng HTTP) loại tài nguyên mà họ quan tâm và máy chủ có thể trả về tài nguyên, chỉ định loại nội dung của tài nguyên mà nó đang phục vụ.

**Rest Based Controller**

Sau đây là một controller dựa trên REST, thực hiện API REST. Đây là những gì API REST của chúng ta thực hiện:

* GET yêu cầu tới / api / user /  trả về danh sách user
* GET yêu cầu tới / api / user / 1 trả về user có ID = 1
* POST yêu cầu tới / api / user / với một đối tượng user khi JSON tạo một user mới
* PUT yêu cầu tới / api / user / 3 với đối tượng user khi JSON cập nhật user với ID = 3
* DELETE yêu cầu tới / api / user / 4 xóa user có ID = 4
* DELETE yêu cầu tới / api / user / xóa tất cả user

@RestController  
public class HelloWorldRestController {  
  
 @Autowired  
 UserService userService; //Service which will do all data retrieval/manipulation work  
  
  
 //-------------------Retrieve All Users--------------------------------------------------------  
  
 @RequestMapping(value = "/user/", method = RequestMethod.*GET*)  
 public ResponseEntity<List<User>> listAllUsers() {  
 List<User> users = userService.findAllUsers();  
 if(users.isEmpty()){  
 return new ResponseEntity<List<User>>(HttpStatus.NO\_CONTENT);//You many decide to return HttpStatus.NOT\_FOUND  
 }  
 return new ResponseEntity<List<User>>(users, HttpStatus.OK);  
 }  
  
  
 //-------------------Retrieve Single User--------------------------------------------------------  
  
 @RequestMapping(value = "/user/{id}", method = RequestMethod.*GET*, produces = MediaType.APPLICATION\_JSON\_VALUE)  
 public ResponseEntity<User> getUser(@PathVariable("id") long id) {  
 System.*out*.println("Fetching User with id " + id);  
 User user = userService.findById(id);  
 if (user == null) {  
 System.*out*.println("User with id " + id + " not found");  
 return new ResponseEntity<User>(HttpStatus.NOT\_FOUND);  
 }  
 return new ResponseEntity<User>(user, HttpStatus.OK);  
 }  
  
  
  
 //-------------------Create a User--------------------------------------------------------  
  
 @RequestMapping(value = "/user/", method = RequestMethod.*POST*)  
 public ResponseEntity<Void> createUser(@RequestBody User user, UriComponentsBuilder ucBuilder) {  
 System.*out*.println("Creating User " + user.getName());  
  
 if (userService.isUserExist(user)) {  
 System.*out*.println("A User with name " + user.getName() + " already exist");  
 return new ResponseEntity<Void>(HttpStatus.CONFLICT);  
 }  
  
 userService.saveUser(user);  
  
 HttpHeaders headers = new HttpHeaders();  
 headers.setLocation(ucBuilder.path("/user/{id}").buildAndExpand(user.getId()).toUri());  
 return new ResponseEntity<Void>(headers, HttpStatus.CREATED);  
 }  
  
  
 //------------------- Update a User --------------------------------------------------------  
  
 @RequestMapping(value = "/user/{id}", method = RequestMethod.*PUT*)  
 public ResponseEntity<User> updateUser(@PathVariable("id") long id, @RequestBody User user) {  
 System.*out*.println("Updating User " + id);  
  
 User currentUser = userService.findById(id);  
  
 if (currentUser==null) {  
 System.*out*.println("User with id " + id + " not found");  
 return new ResponseEntity<User>(HttpStatus.NOT\_FOUND);  
 }  
  
 currentUser.setName(user.getName());  
 currentUser.setAge(user.getAge());  
 currentUser.setSalary(user.getSalary());  
  
 userService.updateUser(currentUser);  
 return new ResponseEntity<User>(currentUser, HttpStatus.OK);  
 }  
  
 //------------------- Delete a User --------------------------------------------------------  
  
 @RequestMapping(value = "/user/{id}", method = RequestMethod.*DELETE*)  
 public ResponseEntity<User> deleteUser(@PathVariable("id") long id) {  
 System.*out*.println("Fetching & Deleting User with id " + id);  
  
 User user = userService.findById(id);  
 if (user == null) {  
 System.*out*.println("Unable to delete. User with id " + id + " not found");  
 return new ResponseEntity<User>(HttpStatus.NOT\_FOUND);  
 }  
  
 userService.deleteUserById(id);  
 return new ResponseEntity<User>(HttpStatus.NO\_CONTENT);  
 }  
  
  
 //------------------- Delete All Users --------------------------------------------------------  
  
 @RequestMapping(value = "/user/", method = RequestMethod.*DELETE*)  
 public ResponseEntity<User> deleteAllUsers() {  
 System.*out*.println("Deleting All Users");  
  
 userService.deleteAllUsers();  
 return new ResponseEntity<User>(HttpStatus.NO\_CONTENT);  
 }  
  
}

**Trong đó:**

* **@RestController** là kết hợp của @Controller và @ResponseBody
* **@RequestBody**: Nếu tham số phương thức được chú thích bằng @RequestBody, Spring sẽ liên kết phần thân yêu cầu HTTP đến với tham số đó.
* **ResponseEntity** đại diện cho toàn bộ phản hồi HTTP
* **@PathVariable** chỉ ra rằng tham số phương thức sẽ được liên kết với URI (id).

Last modified: Thursday, 21 March 2019, 3:16 PM