RESTful API

REpresentational State Transfer
Application Programming Interface

SC363204

Java Web Application Development

การพัฒนาโปรแกรมประยุกต์บนเว็บด้วยภาษาจาวา

Jakkrit Kaewyotha, Ph.D.
College of Computing, Khon Kaen University

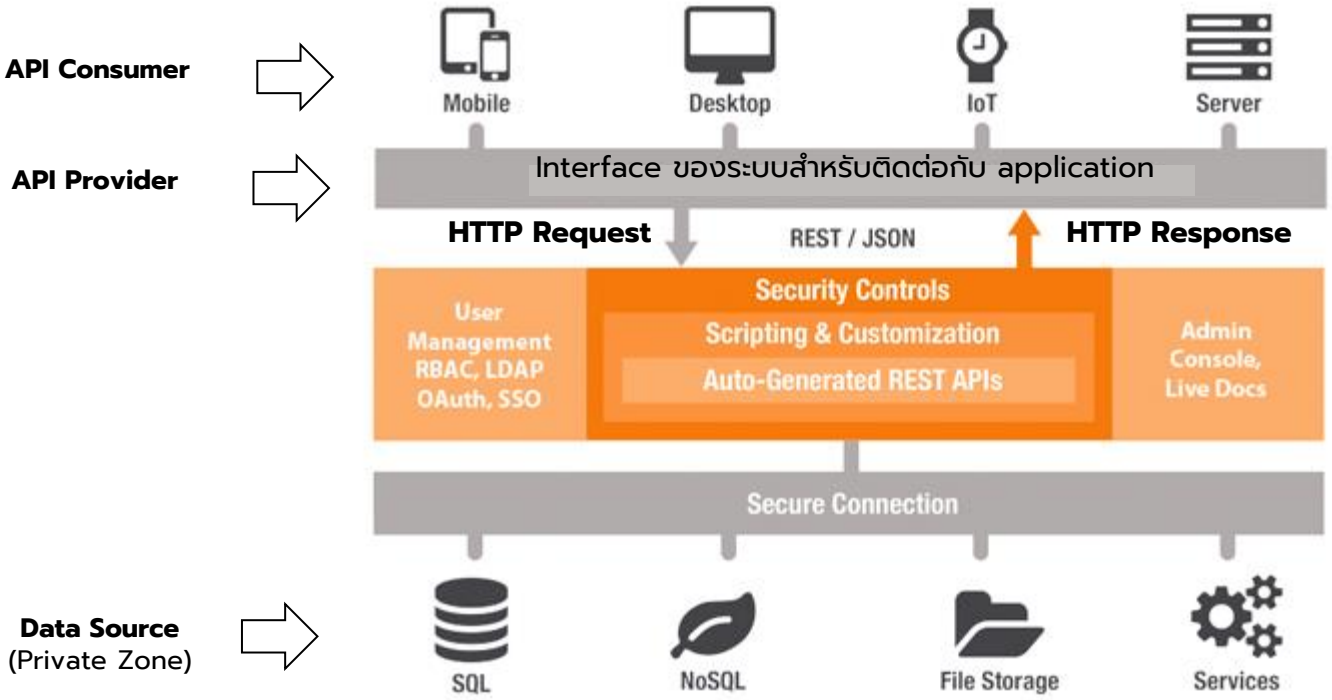
RESTful API

- ซอฟท์แวร์ที่อนุญาตให้แอปพลิเคชันอื่นๆ สามารถเรียกใช้ได้ เพื่อขอข้อมูล หรือช่วยประมวลงานเฉพาะทาง
- ผู้ให้บริการ API
  - Twitter API หรือ Facebook API ให้บริการข้อมูลเพื่อให้แอปพลิเคชันอื่นๆนำไปวิเคราะห์ (Data Analytics) ด้านต่างๆ เช่น การตลาด กระแสความนิยม
  - Google Maps ให้บริการข้อมูลเกี่ยวกับแผนที่
- ผู้ใช้ API
  - Mobile Application
  - Web Application
  - IoT Application

RESTful API

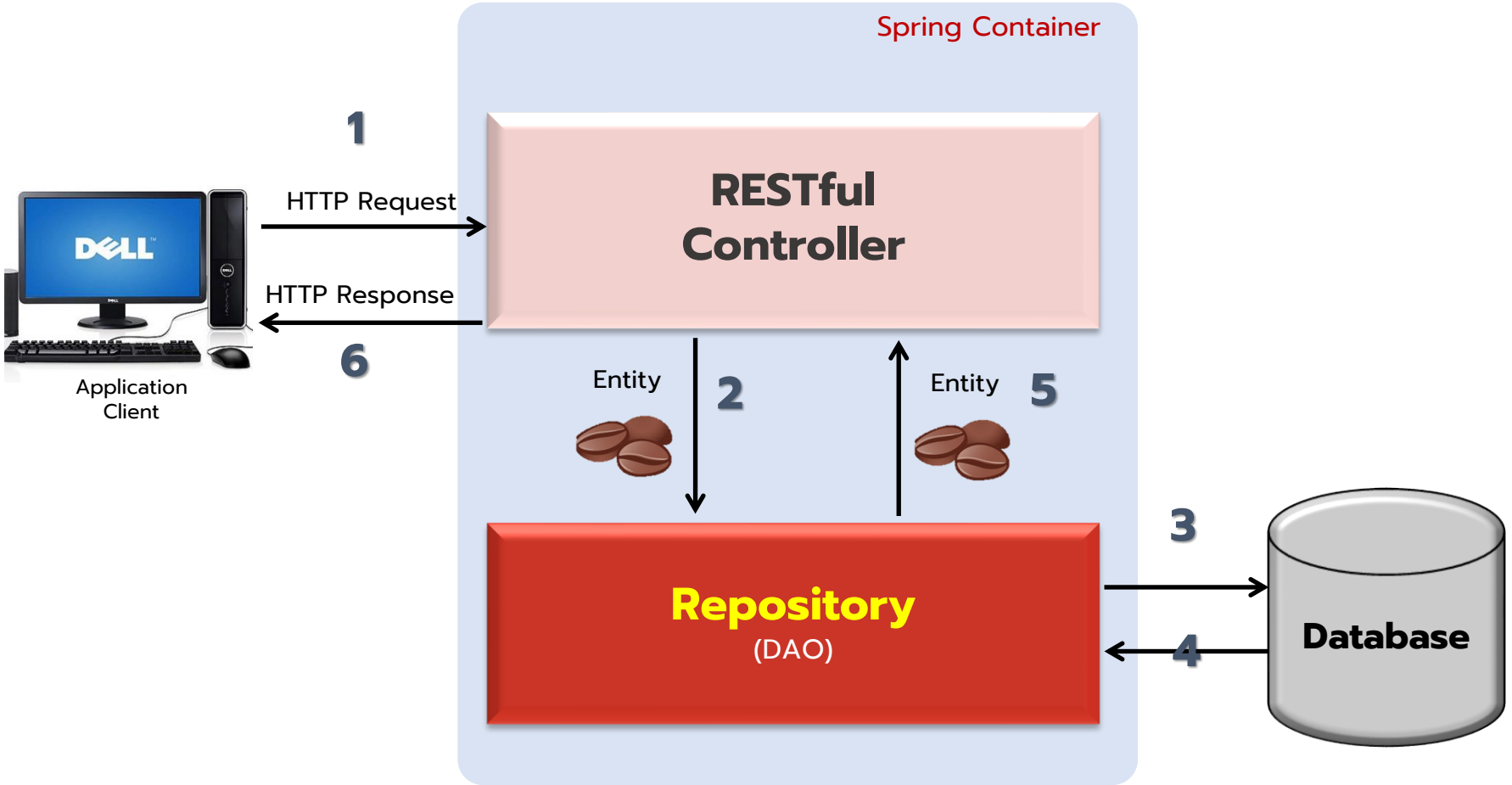
- มองข้อมูลทุกอย่างเป็นแหล่งข้อมูล (Resource) ที่อ้างอิงได้ด้วย URI
- รับ HTTP Request และส่ง HTTP Response บนเครือข่าย internet
- การทำงานมีความคล้ายคลึงกับฟังก์ชัน หรือเมธอด คือ สามารถรับ input และ return ผลลัพธ์กลับได้
- HTTP Response มีรูปแบบที่สามารถนำไปประมวลผลต่อได้ เช่น XML, JSON, Plain Text

สถาปัตยกรรม



ฟังก์ชันพื้นฐาน เช่น CRUD สร้างครั้งเดียว ใช้งานกับ App ที่อยู่บนอุปกรณ์ที่หลากหลาย

การสร้าง RESTful API ด้วย Spring



URL สำหรับเรียก RESTful API

- URL ที่ใช้ส่ง request ไปยัง RESTful API (API Endpoint) มีหลากหลายรูปแบบ แต่มักจะเป็นคำนาม และมีความหมายเป็นข้อมูลที่ต้องการ
  - /customers
    - ขอข้อมูลลูกค้าทั้งหมด
  - /customers/{id} เช่น /customers/5
    - ขอข้อมูลลูกค้าที่มีรหัสลูกค้าเป็น 5
  - /customers?gender=male&province=khonkaen
    - ระบุเงื่อนไขด้วย parameter
  - /orders/{id}/lineitems เช่น /orders/5/lineitems
    - ขอข้อมูลรายการย่อยของใบสั่งซื้อตามรหัสที่ระบุ

HTTP Method สำหรับ RESTful API

- RESTful API อาจใช้ URL เดียวกัน แต่กำหนด HTTP Request Method ที่แตกต่างกัน เพื่อจำแนกการทำงาน

Method	Database Operation	การทำงาน
GET	SELECT	ขอข้อมูลแบบรายการ หรือขอข้อมูล 1 รายการ
POST	INSERT	เพิ่มข้อมูล (ไม่ต้องรู้ ID)
PUT	UPDATE	แก้ไขข้อมูล (ต้องรู้ ID)
DELETE	DELETE	ลบข้อมูล (ต้องรู้ ID)

ตัวอย่าง

- http://www.javawebcpkku.com/customers/5
  - GET – ดึงข้อมูลลูกค้าหมายเลข 5
  - PUT - แก้ไขข้อมูลลูกค้าหมายเลข 5
  - DELETE - ลบข้อมูลลูกค้าหมายเลข 5

- http://www. javawebcpkku.com/customers
  - GET – ดึงข้อมูลลูกค้าทั้งหมด
  - POST – เพิ่มข้อมูลลูกค้า

ตัวอย่าง

ชื่อบริการ	Method	URL สำหรับส่ง request	ข้อมูลที่ส่งไป	ข้อมูลที่ส่งกลับ
ให้รายการข้อมูลสินค้า	GET	http://localhost:8080/products	-	รายการสินค้าทั้งหมด
เพิ่มข้อมูลสินค้า	POST	http://localhost:8080/products	ข้อมูลสินค้า รูปแบบ JSON	ข้อมูลสินค้าที่เพิ่มใหม่ พร้อมรหัส
ให้ข้อมูลสินค้าตามรหัส	GET	http://localhost:8080/products/{id}	แนบรหัสสินค้ามากับ URL	ข้อมูลสินค้า 1 ชิ้น
แก้ไขข้อมูลสินค้า	PUT	http://localhost:8080/products/{id}	ข้อมูลสินค้า รูปแบบ JSON	ข้อมูลสินค้าที่แก้ไข
ลบข้อมูลสินค้า	DELETE	http://localhost:8080/products/{id}	แนบรหัสสินค้ามากับ URL	-

การใช้ Spring MVC สร้าง RESTful API

- สร้างคลาส @RestController
- กำหนด URL Mapping ให้สอดคล้องกับการทำงาน
- กำหนด response โดย return object ที่ต้องการ ซึ่งจะถูกแปลงเป็น JSON อัตโนมัติ
- ทดสอบการทำงานด้วย Postman

RESTful API สำหรับดึงข้อมูล

กำหนด path เริ่มต้น โดยอาจใส่ version ให้ api ในรูปแบบ path

```
@RestController
@RequestMapping("/api/v1")
public class CustomerRestController {
	@Autowired
	private CustomerRepository repo;

	@GetMapping("/customers")
	public List<Customer> getAllCustomer() {
		return repo.findAll();
	}

	@GetMapping("/customer/{id}")
	public Customer getCustomer(@PathVariable("id") Integer id) {
		return repo.findById(id);
	}

	. . .

}
```

RESTful API สำหรับเพิ่ม/แก้ไข/ลบ

```java
@PostMapping("/customer")
public Customer addCustomer(@RequestBody Customer customer) {
        return repo.save(customer);
}

@PutMapping("/customer")
public Customer editCustomer(@RequestBody Customer customer) {
        // ค่าที่ส่งมาเป็นเพียง JavaBean จึงต้องดึง Entity Object ออกมาก่อน
        Customer editCustomer = repo.findById(customer.getId());
        // ย้ายค่าจาก JavaBean ลง Entity Object
        editCustomer.setFirstName(customer.getFirstName());
        editCustomer.setLastName(customer.getLastName());
        return repo.save(editCustomer);
}

@DeleteMapping("/customer/{id}")
public void deleteCustomer(@PathVariable("id") Integer id) {
        repo.delete(id);
}
```

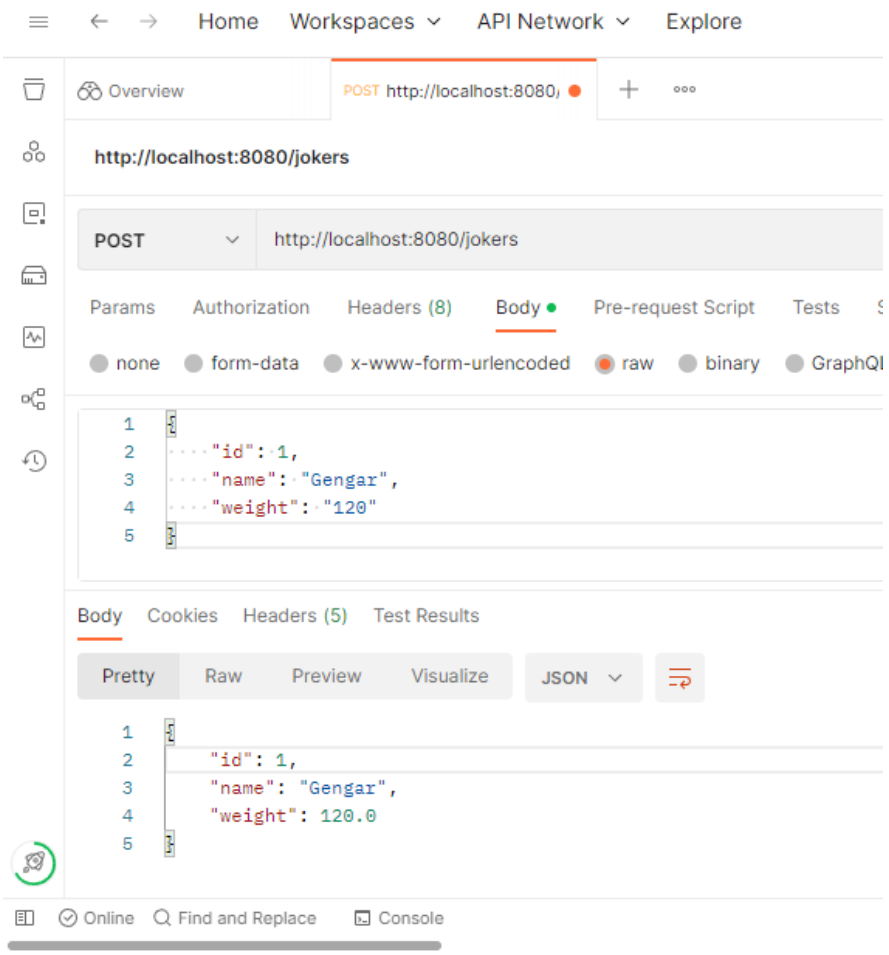
@RequestBody คือ การรับใน รูปแบบ JSON และแปลงเป็น object อัตโนมัติ

การรับ request แบบ JSON

- ใช้ @RequestBody กำกับ

```
@PostMapping("/jokers")
public Joker AddCustomer(@RequestBody Joker joker) {
System.out.println(joker.getName());
System.out.println(joker.getWeight());
return joker;
}
```

```
public class Joker {
        private Integer id;
        private String name;
        private Float weight;
        private Double footSize;
}
```



การสร้าง Error Message

```java
@RestController
@RequestMapping("/api/v2")
public class CustomerRest2 {
	@Autowired
	private CustomerRepository repo;

	@GetMapping("/customer/{id}")
	public ResponseEntity<?> getCustomer(@PathVariable Integer id) {

		Customer customer = repo.findById(id);

		if (customer != null) {
			return new ResponseEntity<>(customer, HttpStatus.OK);

		}  else {
			ErrorDetail errorDetail = new ErrorDetail();
			errorDetail.setStatus(HttpStatus.NOT_FOUND.value());
			errorDetail.setMessage("Customer with id " + id + " not found");
			return new ResponseEntity<>(errorDetail, HttpStatus.NOT_FOUND);
		}
	}
}
```

```java
public class ErrorDetail {
	private int status;
	private String message;

	public int getStatus() {
		return status;
	}

	public void setStatus(int status) {
		this.status = status;
	}

	public String getMessage() {
		return message;
	}

	public void setMessage(String message) {
		this.message = message;
	}
}
```

สร้าง Object ของ ResponseEntity เพื่อใช้เก็บ object ที่จะส่งกลับไป

ตัวอย่าง response หากค้นหาข้อมูลไม่พบ

```json
{
    "status": 404,
    "message": "Customer with id 66 not found"
}
```

การส่งคำร้องไปยัง RESTful API

- RESTful API ถูกเรียกใช้ด้วย URL ดังนั้นผู้ให้บริการจะต้องอธิบายรูปแบบ URL การเรียกใช้แก่นักพัฒนาที่ต้องการมาใช้บริการ