Contents

[Introduction 3](#_Toc494118698)

[RESTful Web Services 3](#_Toc494118699)

[What is REST? 3](#_Toc494118700)

[HTTP Methods 3](#_Toc494118701)

[RESTful Web Services 3](#_Toc494118702)

[Environment Setup 3](#_Toc494118703)

[First Application 4](#_Toc494118704)

[Resources 5](#_Toc494118705)

[What is a Resource? 5](#_Toc494118706)

[Representation of Resources 5](#_Toc494118707)

[Good Resources Representation 5](#_Toc494118708)

[Message 5](#_Toc494118709)

[HTTP Request 6](#_Toc494118710)

[HTTP Response 6](#_Toc494118711)

[Example 7](#_Toc494118712)

[Addressing 7](#_Toc494118713)

[Constructing a Standard URI 7](#_Toc494118714)

[Method 8](#_Toc494118715)

[Statelessness 8](#_Toc494118716)

[Advantages of Statelessness (ข้อดี) 8](#_Toc494118717)

[Disadvantages of Statelessness (ข้อเสีย) 8](#_Toc494118718)

[Caching 8](#_Toc494118719)

[Cache-Control Header 9](#_Toc494118720)

[Best Practices 9](#_Toc494118721)

[Security 10](#_Toc494118722)

[HTTP Code 10](#_Toc494118723)

[Java (JAX-RS) 11](#_Toc494118724)

[Specifications 11](#_Toc494118725)

[Note 12](#_Toc494118726)

[Summary 12](#_Toc494118727)

[Reference 13](#_Toc494118728)

# Introduction

## RESTful Web Services

RESTful web services เป็น web services สถาปัตยกรรม REST ซึ่งในสถาปัตยกรรม REST ทุกอย่างเป็นทรัพยากร

## What is REST?

REST ย่อมาจาก Representational State Transfer. REST เป็นสถาปัตยกรรมที่ใช้มาครฐาน Web และใช้ protocol HTTP สำหรับการสื่อสารข้อมูล การเข้าถึงทรัพยากรด้วย Interface ร่วมกัน โดยใช้มาตรฐาน HTTP REST ซึ่งแต่ละแหล่งข้อมูลจะมีการระบุโดย URI หรือ Global ID REST ใช้ในการแทนค่าต่างๆ เพื่อแสดงทรัพยากร เช่น Text, JSON และ XML โดย JSON เป็นรูปแบบที่นิยมใช้มากที่สุดใน web services

### HTTP Methods

HTTP ที่ใช้บ่อยที่สุดในสถานปัตยกรรม REST คือ

1. GET ให้เข้าถึงทรัพยากรแบบอ่านอย่างเดียว
2. PUT ใช้เพื่อสร้างทรัพยากรใหม่
3. DELETE ใช้เพื่อนำทรัพยากรออก
4. POST ใช้เพื่อแก่ไขทรัพยากรที่มีอยู่ หรือ สร้างทรัพยากรใหม่

## RESTful Web Services

Web services คือชุด protocol ที่เปิดใช้งานสำหรับแลกเปลี่ยนข้อมูลระหว่าง application หรือ system ที่เขียนขึ้น

Web services บนสถาปัตยกรรม REST เรียกว่า RESTful Web Services. Web service ใช้ HTTP method

# Environment Setup

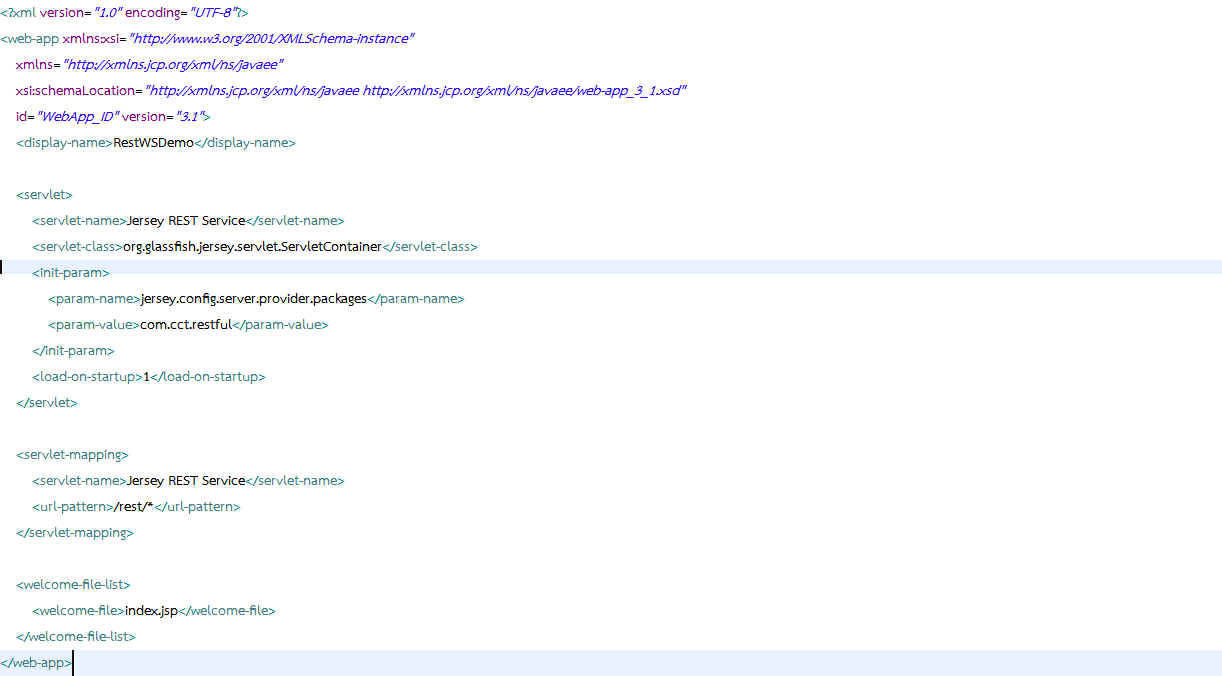
การเตรียม environment สำหรับการพัฒนา โดยจะทำงานร่วมกับ Jersey Framework เพื่อสร้าง RESTful Web Services โดย Jersey Framework ใช้ JAX-RS 2.0 API ซึ่งเป็นข้อกำหนดมาตรฐาน เพื่อสร้าง RESTful Web Services

1. Java 8
2. Tomcat 8
3. Eclipse Neon
4. Jersey Framework Libraries

# First Application

ก่อนการเริ่มต้นเขียน RESTful Web Services กับ Jersey Framework ต้องทำการตรวจสอบการตั้งค่า environment ของ Jersey Framework ให้ถูกต้อง

1. ทำการสร้าง Java Project เป็น Dynamic Web Project
2. เพิ่ม libraries ที่ต้องการ
3. สร้าง web.xml configuration file



1. สร้าง Web Service
2. ทำการสร้าง client สำหรับทดสอบ

# Resources

## What is a Resource?

สถาปัตยกรรม REST ถือว่าทุกๆ content เป็นทรัพยากร แหล่งข้อมูลเหล่านี้ สามารถเป็นไฟล์ข้อความ, หน้า html, รูปภาพ, วิดีโอ หรือ Dynamic Business Data. REST Server จะ provides การเข้าถึงทรัพยากร และ REST Client จะเข้าถึงและเปลี่ยนทรัพยากร แต่ละทรัพยากรจะระบุโดย URI หรือ Global ID

### Representation of Resources

ทรัพยากรใน REST เป็น Object ที่คล้าย OOP หรือเหมือนกับ Entity ในฐานข้อมูล เมื่อทรัพยากรถูกระบุแล้ว การแทนของมันคือการตัดสินใจโดยใช้รูปแบบมาตรฐาน เพื่อให้ server สามารถส่งทรัพยากรในรูปแบบดังกล่าว และเข้าใจในรูปแบบเดียวกัน

### Good Resources Representation

REST ไม่ได้กำหนดข้อจำกัด เกี่ยวกับ format ของทรัพยากร Client สามารถขอแบบ JSON ในขณะที่ Client อื่นขอแบบ XML ของทรัพยากรเดียวกัน ไปยัง Server และ เป็นหน้าที่ของ REST ทีจะตอบกลับ Client ใน format ที่เข้าใจ

สิ่งสำคัญที่ต้องพิจารณาขณะออกแบบรูปแบบของทรัพยากร ใน RESTful Web Services

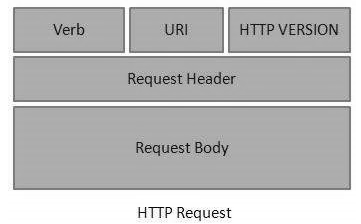
* ความเข้าใจ ทั้ง server และ client ควรเข้าใจและใช้รูปแบบของทรัพยากร
* ความครบถ้วน format ควรสามารถแสดงทรัพยากรได้อย่างสมบูรณ์
* การเชื่อมโยง สามารถเชื่อมโยงไปยังแหล่งข้อมูลอื่น และรูปแบบก็รองรับ

ปัจจุบัน web services ส่วนใหญ่จะแสดงทรัพยาโดยใช้รูปแบบ XML หรือ JSON ซึ่งมี Libraries หรือ Tools ที่ช่วยในการ parse และ ปรับเปลี่ยนข้อมูล XML และ JSON ได้

# Message

RESTful Web Services ใช้โปรโตคอล HTTP เป็นสื่อการติดต่อสื่อสารระหว่าง client กับ server โดย client จะส่งข้อความในรูปแบบคำขอ HTTP และ server จะตอบในรูปแบบ HTTP เทคนิคนี้เรียกว่า Messaging ข้อความเหล่านี้ประกอบด้วยข้อมูลข้อความและข้อมูลเมตาเช่นข้อมูลเกี่ยวกับข้อความ ให้เราดูข้อความขอ HTTP และ HTTP Response สำหรับ HTTP 1.1

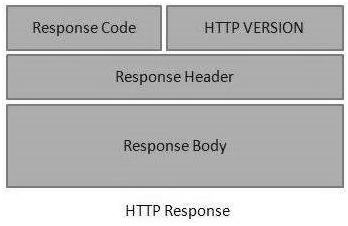
## HTTP Request



HTTP Request มี 5 ส่วนที่สำคัญ

* Verb แสดง HTTP Methods เช่น GET, POST, DELETE, PUT ฯลฯ
* URI Uniform Resource Identifier (URI) เพื่อระบุทรัพยากรบน server
* HTTP Version ระบุ HTTP Version เช่น HTTP v1.1
* Request Header ประกอบด้วย meta data สำหรับข้อความ HTTP Request เหมือน key - value เช่น client (หรือ browser) ประเภท, รูปแบบ ที่รองรับ client, รูปแบบของ message body, การตั้งค่า cache ฯลฯ
* Request Body เนื้อหาของข้อความหรือการเป็นตัวแทนทรัพยากร

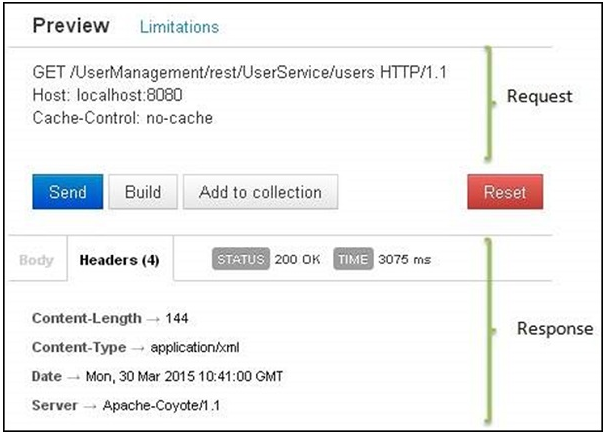
## HTTP Response



HTTP Response มี 4 ส่วนที่สำคัญ

* Status / Response Code ระบุ server status สำหรับทรัพยากรที่ request เช่น 404 คือทรัพยากรที่ไม่พบและ 200 คือการตอบสนองก็โอเค
* HTTP Version ระบุ HTTP Version เช่น HTTP v1.1
* Response Header ประกอบด้วย meta data สำหรับ HTTP Response เป็น key – value เช่นcontent length, content type, response date, server type เป็นต้น
* Response Body เนื้อหาข้อความตอบกลับหรือการเป็นตัวแทนทรัพยากร

## Example



# Addressing

การระบุ address หมายถึงการค้นหาทรัพยากร หรือ แหล่งข้อมูลจำนวนมาก ที่อยู่บน server และทรัพยากร ในสถาปัตยกรรม REST คือจะถูกระบุโดย URI (Uniform Resource Identifier) มีรูปแบบต่อไปนี้



วัตถุประสงค์ของ URI คือการค้นหาทรัพยากรบน server ที่ติดตั้ง web services สิ่งสำคัญของคำขอคือ VERB ซึ่งระบุสิ่งที่จะดำเนินการกับทรัพยากร เช่น จากตัวอย่าง First Application URI คือ <http://localhost:8080/RestWSDemo/rest/UserService/users> และ VERB เป็น GET

## Constructing a Standard URI

สิ่งสำคัญที่ต้องพิจารณาในการออกแบบ URI

* ใช้คำนามพหูพจน์ ใช้คำนามพหูพจน์ในการกำหนดแหล่งข้อมูล
* หลีกเลี่ยงการใช้ช่องว่างใช้เครื่องมาย \_ หรือ – เมื่อชื่อทรัพยากรยาว
* ใช้ตัวอักษรพิมพ์เล็ก แม้ URI ไม่คำนึงตัวพิมพ์เล็ก แต่เป็นวิธีที่ดีในการเก็บ URL ไว้เฉพาะอักษรตัวพิมพ์เล็ก

เท่านั้น

* Maintain ย้อนหลัง web services เป็นบริการสาธารณะ URI เมื่อเป็นสาธารณะควรมีไว้ใช้งานเสมอ ใน

กรณีที่เปลี่ยน URI ใช้ HTTP status code 300

* ใช้ HTTP Verb เช่น GET, PUT,และ DELETE เพื่อจัดการกับทรัพยากร ไม่ควรใช้ชื่อ URI

# Method

RESTful web service ใช้ประโยชน์จาก HTTP Verb ในการระบุการดำเนินการกับทรัพยากร ประเด็นสำคัญที่ต้องคำนึง

* GET read only และปลอดภัย
* PUT and DELETE ผลลัพธ์เหมือนกันไม่ว่าเรียกกี่ครั้ง
* PUT and POST ผลลัพธ์เกือบเหมือนกัน

# Statelessness

ตามสถาปัตยกรรม REST, RESTful Web Services ไม่ควรเก็บ client state ไว้บน server ข้อจำกัดนี้เรียกว่า Statelessness เป็นหน้าที่ของ client ที่ต้องส่ง context ไปยัง server และ server จะ store ไว้เพื่อประมวลผล

## Advantages of Statelessness (ข้อดี)

1. Web services สามารถประมวลผล request แต่ละ method อิสระ
2. Web services ไม่ต้อง maintain client ก่อนหน้า ช่วยลดความยุ่งยากซับซ้อน
3. ข้อดี HTTP Protocol, RESTful Web Services ทำงานกับ HTTP Protocol ได้อย่างราบรื่น

## Disadvantages of Statelessness (ข้อเสีย)

Web services ต้องได้รับข้อมูลเพิ่มในแต่ละ request

# Caching

Caching หมายถึงการจัดเก็บ Response ของ Server ใน Client เพื่อให้ Client ไม่จำเป็นต้อง server request สำหรับทรัพยากรเดียวกันซ้ำแล้วซ้ำอีก Response ของ Server ควรมีข้อมูลเกี่ยวกับวิธีการทำ Caching เพื่อให้ client เก็บข้อมูล Response สำหรับช่วงเวลาหรือไม่ควรเก็บ response ของ server ไว้

Header ที่ Response ของ Server เพื่อกำหนดค่า caching ของ client

* Date วันที่และเวลาของทรัพยากรเมื่อถูกสร้าง
* Last Modified วันที่และเวลาของทรัพยากรเมื่อมีการปรับเปลี่ยนครั้งล่าสุด
* Cache – Control primary header เพื่อควบคุม caching
* Expires วันที่และเวลาหมดอายุของ caching
* Age ระยะเวลา (วินาที) เมื่อดึงข้อมูลทรัพยากรจาก server

## Cache-Control Header

* Public resource สามารถ cache ด้วย component
* Private resource สามารถ cache ได้ด้วย client และ server เท่านั้น ไม่มีตัวกลางใด

สามารถ cache ทรัพยากร ได้

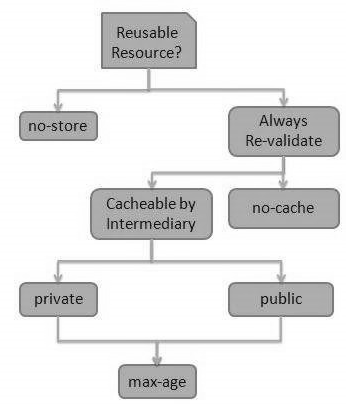
* no-cache / no-store resource ไม่สามารถ cache ได้
* max-age อายุสูงสุดที่ cache สามารถใช้งานได้ (วินาที) หลังจากนั้น client ต้อง request อีก

ครั้ง

* must-revalidate server ตรวจสอบทรัพยากรซ้ำ เมื่อ max-age ผ่านไปแล้ว

## Best Practices

* เก็บ content เช่น image, css, java script ได้โดยมีวันหมดอายุ 2 – 3 วัน
* ไม่กำหนด expiry date สูงเกินไป
* Content cache ไม่กี่ชั่วโมง



# Security

RESTful Web Services ทำงานร่วมกับ HTTP URL Path การป้องกัน RESTful Web Service จึงเป็นสิ่งสำคัญมาก ในลักษณะเดียวกับ website ที่มีการรักษาความปลอดภัย

แนวทางปฏิบัติที่ดีที่สุด ในขณะที่ออกแบบ RESTful Web Service

* Validation

ตรวจสอบความถูกต้องทั้งหมดบน server ปกป้อง server จากการโจมตีแบบ SQL หรือ NoSQL injection

* Session Based Authentication

ใช้การรับรองความถูกต้องของ session เพื่อรับรองความถูกต้องของ user เมื่อใดก็ตามที่มีการ request ไปยังweb services

* No Sensitive Data in the URL

อย่าใช้ username, password หรือ session token ใน URL ค่าเหล่านี้ควรถูกส่งผ่านไปยัง Web Service ผ่านทางวิธี POST

* Restriction on Method Execution

อนุญาตให้มีการใช้ method ต่างๆอย่าง จำกัด เช่น GET, POST และ DELETE วิธีการ GET ไม่ควรลบข้อมูล

* Validate Malformed XML / JSON

ตรวจสอบ input ที่มี format ส่งไปยัง method ของ web service

* Throw generic Error Messages

Method ของ web service ควรใช้ข้อความของ HTTP Error เช่น 403 เพื่อแสดงสิทธิ์การเข้าถึง ฯลฯ

## HTTP Code

* 200 OK
* 201 CREATED

ทรัพยากรถูกสร้างขึ้นโดยใช้ POST หรือ PUT request แล้วส่งกลับ link ที่สร้าง resource ใหม่ โดยใช้ location header

* 204 NO CONTENT

เมื่อ response body เป็นค่าว่าง เช่น DELETE request

* 304 NOT MODIFIED

ใช้เพื่อลดการใช้ bandwidth ในกรณีที่มีเงื่อนไขของ GET request โดย response body จะเป็นค่าว่าง, header จะมีวันที่ location ฯลฯ

* 400 BAD REQUEST

ข้อมูลไม่ถูกต้อง เช่น validate error, ไม่พบข้อมูล

* 401 UNAUTHORIZED

เมื่อ authen token ไม่ถูกต้อง

* 403 FORBIDDEN

ผู้ใช้ไม่สามารถเข้าใช้งาน method

* 404 NOT FOUND

Method ไม่พร้อมใช้งาน

* 409 CONFLICT

เกิด conflict ขณะ run method เช่น เพิ่มรายการที่ซ้ำกัน

* 500 INTERNAL SERVER ERROR

Server throw exception ในขณะ execute method

# Java (JAX-RS)

JAX-RS เป็นมาตรฐาน Java API สำหรับ RESTful Web Services. JAX-RS เป็น Java พื้นฐานที่รองรับการสร้าง RESTful Web Services. JAX-RS version 2.0 จะใช้ annotations เพื่อลดความซับซ้อนในการพัฒนา web services

## Specifications

@Path Path ของ Class หรือ Method ทรัพยากร

@GET HTTP GET Request ใช้ดึง ทรัพยากร

@PUT HTTP PUT Request ใช้สร้าง ทรัพยากร

@POST HTTP POST Request ใช้สร้าง/แก้ไข ทรัพยากร

@DELETE HTTP DELETE Request ใช้ลบทรัพยากร

@HEAD HTTP HEAD Request ใช้เพื่อดูสถานะความพร้อมใช้งานของ method

@Produces HTTP Response ที่สร้างขึ้นโดย Web Services เช่น APPLICATION/XML, TEXT/HTML,

APPLICATION/JSON เป็นต้น

@Consumes รูปแบบ HTTP Request เช่น application/x-www-formurlencoded เพื่อยอมรับข้อมูลใน

ระหว่าง HTTP body กับ POST Request

@PathParam Bind parameter passed ไปยัง method ด้วย value ใน path

@QueryParam Bind parameter passed ไปยัง method ด้วย query parameter ใน path

@MatrixParam Bind parameter passed ไปยัง method ด้วย HTTP matrix parameter ใน path

@HeaderParam Bind parameter passed ไปยัง method ด้วย HTTP header

@CookieParam Bind parameter passed ไปยัง method ด้วย Cookie

@FormParam Bind parameter passed ไปยัง method ด้วย Form value

@DefaultValue กำหนดค่า default ให้กับ parameter passed ไปยัง method

@Context Context ของทรัพยากร เช่น HTTPRequest เป็น context

## Note

ใช้ Jersey implement กับ JAX-RS 2.0 โดย Oracle, ใน Restful Web Services

# Summary

1. ใน 1 class สามารถมี method ที่เหมือนกันได้มากกว่า 1 method แต่ต้องทำการกำหนด @Path ดังภาพ



1. ใน 1 project หากมี web service มากกว่า 1 service แล้วต้องการแยก package การทำงานออกจากกัน ให้ทำการใช้เครื่องหมาย comma คั่นระหว่าง package ดังภาพ



# Reference

<https://www.tutorialspoint.com/restful/restful_discussion.htm>

<https://docs.jboss.org/resteasy/docs/3.0.9.Final/userguide/html_single/>

<https://javabrains.io/courses/javaee_jaxrs/>

<https://stackoverflow.com/questions/31657641/tomcat-restful-web-service-deployment>

<https://www.tutorialspoint.com/restful/restful_first_application.htm>

<https://www.tutorialspoint.com/restful/restful_environment.htm>

<http://crunchify.com/create-very-simple-jersey-rest-service-and-send-json-data-from-java-client/>

<https://www.pegaxchange.com/2016/08/11/creating-a-rest-web-service-with-eclipse-neon-tomcat-9-jax-rs-jersey-2-24-and-jackson/>

<https://afsinka.wordpress.com/2015/12/27/restful-web-service-example-with-jersey-2-and-tomcat-8/>

<http://www.vogella.com/tutorials/REST/article.html>

<https://www.tutorialspoint.com/restful/restful_quick_guide.htm>

<https://technologyconversations.com/2014/08/12/rest-api-with-json/>

<http://azagorneanu.blogspot.com/2011/09/configure-maven-to-generate-classes.html>

<https://dennis-xlc.gitbooks.io/restful-java-with-jax-rs-2-0-2rd-edition/en/index.html>