

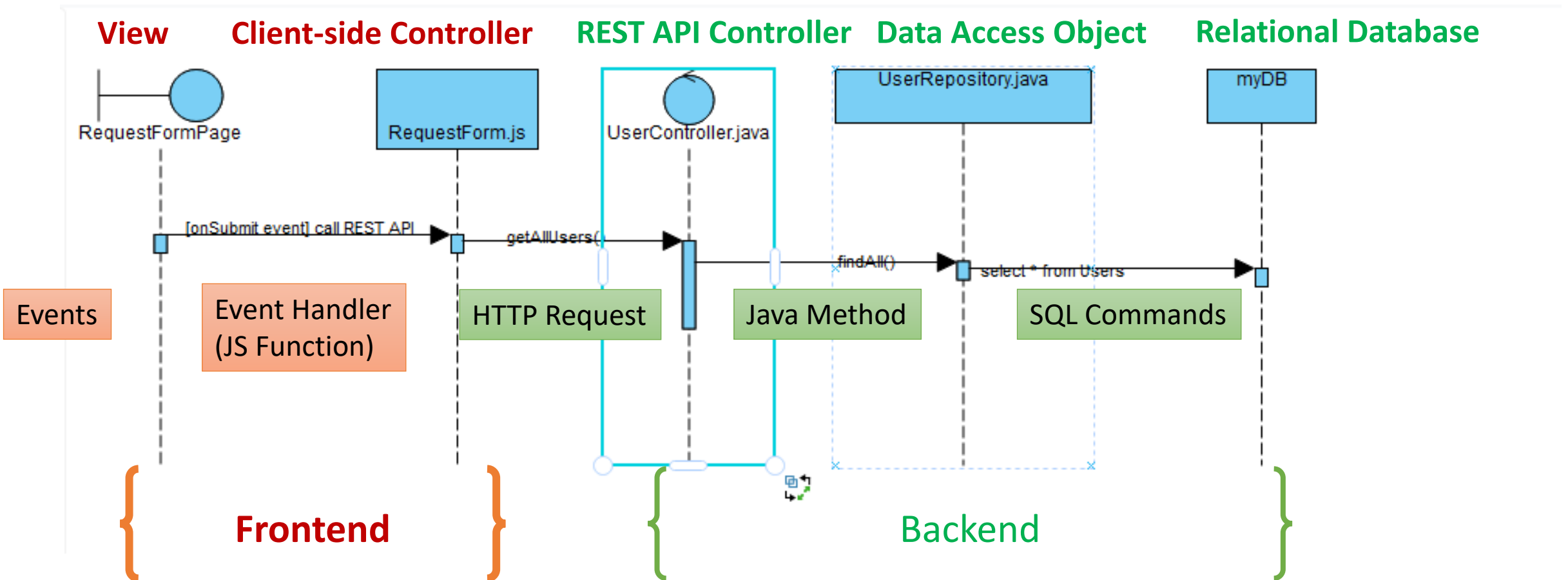
# Workshop สร้าง CRUD API สำหรับจัดการ Data

นำเสนอโดย ผศ.ดร.ทรงศักดิ์ ร่องวิริยะพานิช


# Software Components ที่ประกอบขึ้นเป็น Web Application ตามรูปแบบสถาปัตยกรรม MVC (Model-View-Controller)

- **View** ทำหน้าที่
  - รับคำสั่ง รับข้อมูลอินพุตจาก User เพื่อให้ระบบทำ Action บางอย่างตอบกลับ
  - แสดงผลลัพธ์ที่ได้จากระบบให้ User
- **Controller**
  - Client-side Controller ทำหน้าที่ Validate Input เช็คความถูกต้อง ครบถ้วนของข้อมูลอินพุต ก่อนส่งไป Backend (REST API Controller ที่รับผิดชอบ) เพื่อประมวลผล ทำ Action ตามที่ User ร้องขอ และ เลือก View ที่แสดงผลลัพธ์ที่ได้หลังการประมวลผลตอบกลับจาก Backend เพื่อให้ User ทำงานในขั้นตอนถัดไป เรียกว่า Event Handlers ก็ได้
  - REST API Controller (Server-side Controller) ทำหน้าที่รับคำร้อง Request จาก Client-side Controller เพื่อ process logic ประมวลผล แล้วส่งผลลัพธ์คืนเป็น Model Object ที่ encapsulate Data ส่งคืนให้ Client-side Controller นำไปแสดงผลในหน้า View ให้ User
- **Model** มาจากการวิเคราะห์หา application data ที่ต้องใช้ในระบบ หรือ ที่ต้องจัดเก็บลง DB และ business rules. (as known as domain objects, entities)
  - DTO (Data Transfer Object) เป็น Object ที่เก็บข้อมูลย่อยให้อยู่เป็นก้อน จัดเป็นโครงสร้าง (Encapsulation ให้เป็น Object) เพื่อสะดวกในการส่งต่อระหว่าง Components เช่น Transfer Data จาก Backend ไป Frontend
  - DAO (Data Access Object) หรือ Data Repository เป็น Object ที่ช่วยให้สามารถเข้าถึง Database เพื่อจัดการข้อมูลได้สะดวก ง่าย (Create Retrieve Update Delete Data)
- **Database**
  - Relational Database ประกอบด้วย Data Tables ที่จัดเก็บ 1) ข้อมูลพื้นฐานหลัก (Master Data) และ 2) ข้อมูลรายการธุรกรรม (Transaction Data)
  - No SQL Database เช่น MongoDB

# ตัวอย่าง Scenario แสดง Interaction ระหว่าง Frontend, Client-side Controller, REST API Controller, Data Access Object และ Database



# สิ่งที่ต้องจัดเตรียมก่อนเข้า workshop

- URL สำหรับ TU API: <https://restapi.tu.ac.th/api/v1/auth/Ad/verify>
- Application key:  
TU5e4eb2cbe5154ae0e2c8e0114774e4cc9badea2ae91819780fddfcc35244f3f7c299bd99c0fc9071898547cd02ca7e6b
- สร้าง Database ชื่อ myDB บนเครื่อง Local โดยใช้โปรแกรม Microsoft SQL Server Management Studio (SSMS) 
- กำหนด DB Configuration ในไฟล์ application.properties ภายใน Spring Boot Project

```
1 # Microsoft SQL Server configuration
2 spring.cloud.config.enabled=false
3 spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
4 spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=myDB;encrypt=true;trustServerCertificate=true;
5 spring.datasource.username=sa
6 spring.datasource.password=rongviri126
7
8 # Enable SQL logging for debugging (optional)
9 spring.jpa.show-sql=true
10 spring.jpa.properties.hibernate.format_sql = true
11
12 ## Hibernate Properties
13 # The SQL dialect makes Hibernate generate better SQL for the chosen database
14 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.SQLServer2012Dialect
15
16 # Hibernate ddl auto (create, create-drop, validate, update)
17 spring.jpa.hibernate.ddl-auto = update
```

# Channel ที่สร้าง Application key สำหรับเรียก TU REST API

ตั้งค่า

โปรดรักษาข้อมูลแขนแขนของท่าน เนื่องจากมีข้อมูลสำคัญ

\*Channel ID

8dfccb4e3f6708611b78156d8104ab003b9b0531

\*Token

TU5e4eb2cbe5154ae0e2c8e0114774e4cc9badea2ae91819780fddfcc35244f3f7c299bd99c0fc9071898547cd02ca7e6b

\*ชื่อระบบ

RequestForm-CS264

ไม่เกิน 20 ตัวอักษร

\*รายละเอียด

RequestForm-CS264

## Domain objects ที่เกี่ยวข้องกับ User story (Entity Classes) และ Data ที่เราต้องการ Save ลง DB (Persistence Class)

- Login Account
- Student
- Employee
- User ที่ประกอบด้วย **Attributes** ต่อไปนี้
  - id: Long พร้อมเงื่อนไข “เป็นคีย์ (Primary Key) ที่ถูกสร้างให้ไม่ซ้ำกันของ User”
  - firstName: String พร้อมเงื่อนไข “ค่าห้ามว่าง”
  - lastName: String พร้อมเงื่อนไข “ค่าห้ามว่าง”
  - email: String พร้อมเงื่อนไข “email ของ User ห้ามซ้ำ”

# วิธีการแปลงจาก Persistence Class เป็น Data Table

- จาก Design ที่กำหนด Domain Object → นำไปสู่การสร้างเป็น JPA Entity Class  
→ Data Table ใน Relational Database

User
- id : Long - firstName : String - lastName : String - email : String
+ getId() : Long + getFirstName() : String + getLastName() : String + getEmail() : String + setId(Long id) : void + setFirstName(String firstName) : void + setLastName(String lastName) : void + setEmail(String email) : void

```
6 @Data
7 @Entity
8 @Table(name = "users")
9 public class User {
10
11     @Id
12     @GeneratedValue(strategy = GenerationType.IDENTITY)
13     private Long id;
14
15     @Column(name = "first_name", nullable = false)
16     private String firstName;
17
18     @Column(name = "last_name", nullable = false)
19     private String lastName;
20
21     @Column(name = "email", nullable = false, unique = true)
22     private String email;
23 }
24
```

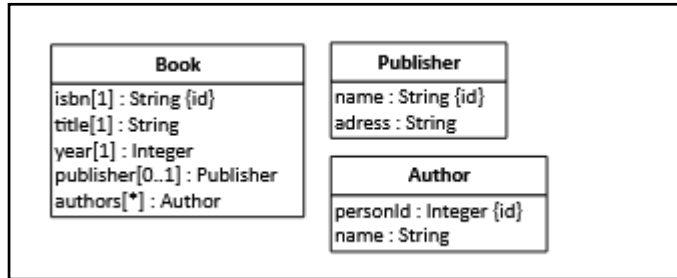
id	email	first_name	last_name

# แบบฝึกหัด ให้นักศึกษา design entities แสดงเป็น class diagram สำหรับโจทย์ต่อไปนี้

- กำหนดให้นักศึกษาร่าง **Domain Objects** สำหรับจัดเก็บข้อมูลต่อไปนี้
- หนังสือ (**Book**) ที่ประกอบด้วย ชื่อหนังสือ หมายเลข **ISBN** หนังสือหนึ่งเล่มสามารถมี **authors** ผู้แต่งหนังสือมากกว่า **1** คน แต่ต้องมีอย่างน้อย **1** คน และ หนังสือมีสำนักพิมพ์ **Publisher 1** แห่ง โดยในการเพิ่มข้อมูลหนังสืออาจจะยังไม่จำเป็นต้องระบุสำนักพิมพ์ก็ได้
- รายละเอียดของสำนักพิมพ์ (**Publisher**) **1** แห่ง ประกอบด้วยชื่อสำนักพิมพ์ และ ที่อยู่
- รายละเอียดของผู้แต่งหนังสือ (**Author**) ประกอบด้วย ชื่อผู้แต่ง รหัส **ID** ของผู้แต่งหนังสือ

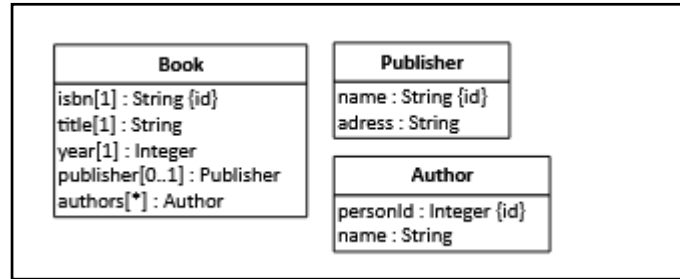


# เฉลย Class diagram สำหรับโจทย์หน้า 7 และ ให้กำหนด Relationship ระหว่าง Entities



- ให้นักศึกษาทำความเข้าใจความสัมพันธ์ระหว่างคลาส 4 แบบดังนี้
  - One To One
  - One To Many
  - Many To One
  - Many To Many
- ตัวอย่าง
  - One-to-One, one record in a table is associated with one and only one record in another table. ได้แก่ Person กับ Passport
  - One-to-Many พิจารณา 1 รายการข้อมูล A สัมพันธ์กับหลายข้อมูล B เช่น Customer กับ Orders
  - Many-to-Many พิจารณาข้อมูล A 1 รายการมีความสัมพันธ์กับหลายข้อมูล B และ ข้อมูล B 1 รายการมีความสัมพันธ์กับหลายข้อมูล A ได้แก่ a student and a teacher. Students are taught by many Teachers, and each Teacher has a class of many Students.
  - Many-to-One พิจารณาเหมือนฟังก์ชันที่ map input หลายค่ากับ output เดียวกัน ได้แก่ Addresses and Zip Code, Products and Category, Students and Grade

# เฉลย Class diagram สำหรับโจทย์หน้า 7 และ ให้กำหนด Relationship ระหว่าง Entities



- ให้นักศึกษาลากเส้นความสัมพันธ์ระหว่าง 3 classes โดยระบุความสัมพันธ์ว่าเป็นแบบ

- One To One
- One To Many
- Many To One
- Many To Many

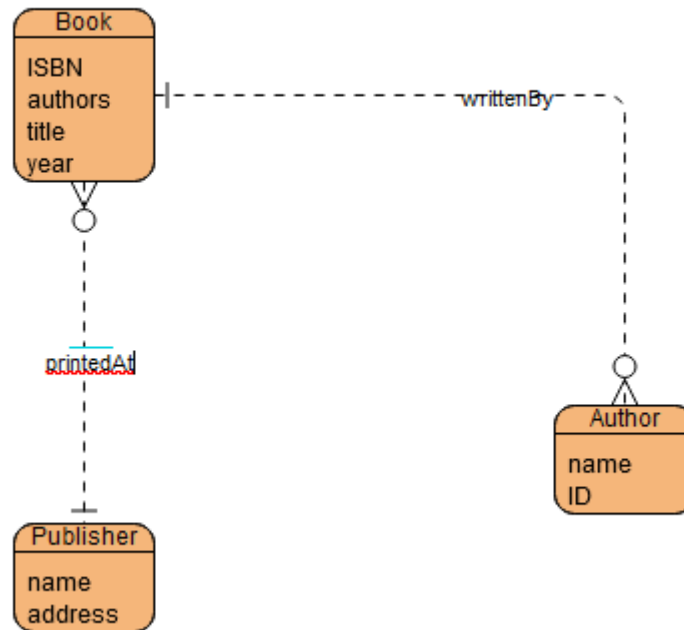


Table 13.1. Sample data for Publisher

Name	Address
Bantam Books	New York, USA
Basic Books	New York, USA

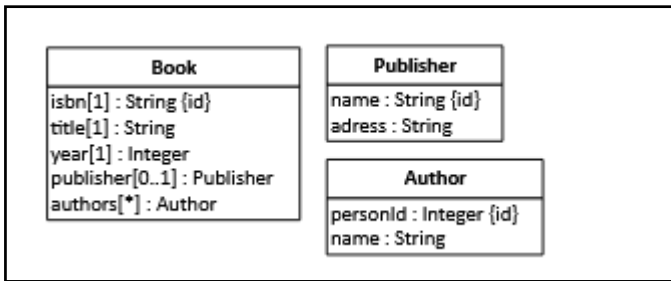
Table 13.2. Sample data for Book

ISBN	Title	Year	Authors	Publisher
0553345842	The Mind's I	1982	1, 2	Bantam Books
1463794762	The Critique of Pure Reason	2011	3	
1928565379	The Critique of Practical Reason	2009	3	
0465030793	I Am A Strange Loop	2000	2	Basic Books

Table 13.3. Sample data for Author

Author ID	Name
1	Daniel Dennett
2	Douglas Hofstadter
3	Immanuel Kant

# เฉลย Relationship ระหว่าง Entities



- ให้นักศึกษาลากเส้นความสัมพันธ์ระหว่าง 3 classes โดยระบุความสัมพันธ์ว่าเป็นแบบ
  - One To One
  - One To Many ได้แก่ 1 Publisher สัมพันธ์กับ Many Books, 1 Author สัมพันธ์กับ Many Books
  - Many To One ได้แก่ Many Books สัมพันธ์กับ 1 Publisher ได้แก่ Many Books กับ 1 Publisher
  - Many To Many ได้แก่ Many Books สัมพันธ์กับ Many Authors

Table 13.1. Sample data for Publisher

Name	Address
Bantam Books	New York, USA
Basic Books	New York, USA

Table 13.2. Sample data for Book

ISBN	Title	Year	Authors	Publisher
0553345842	The Mind's I	1982	1, 2	Bantam Books
1463794762	The Critique of Pure Reason	2011	3	
1928565379	The Critique of Practical Reason	2009	3	
0465030793	I Am A Strange Loop	2000	2	Basic Books

Table 13.3. Sample data for Author

Author ID	Name
1	Daniel Dennett
2	Douglas Hofstadter
3	Immanuel Kant

# วิธีการแปลงจาก Persistence Class เป็น Data Table

- ตัวอย่างเช่น **User** ที่ประกอบด้วย **Attributes** ต่อไปนี้
  - id: Long พร้อมเงื่อนไข “เป็นคีย์ (Primary Key) ที่ถูกสร้างให้ไม่ซ้ำกันของ User”
  - firstName: String พร้อมเงื่อนไข “ค่าห้ามว่าง”
  - lastName: String พร้อมเงื่อนไข “ค่าห้ามว่าง”
  - email: String พร้อมเงื่อนไข “email ของ User ห้ามซ้ำ”
- เขียนเป็น Java Class
- ใส่ Annotation **@Entity** ให้เป็น Persistence Class
- ใช้ **Hibernate Framework** แปลงเป็น Data Table

```
6 @Data
7 @Entity
8 @Table(name = "users")
9 public class User {
10
11     @Id
12     @GeneratedValue(strategy = GenerationType.IDENTITY)
13     private Long id;
14
15     @Column(name = "first_name", nullable = false)
16     private String firstName;
17
18     @Column(name = "last_name", nullable = false)
19     private String lastName;
20
21     @Column(name = "email", nullable = false, unique = true)
22     private String email;
23 }
24
```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'dbo.users' table is expanded, showing its columns: 'id' (PK, bigint, not null), 'email' (varchar(255), not null), 'first\_name' (varchar(255), not null), and 'last\_name' (varchar(255), not null). On the right, the 'Results' tab is active, displaying a table with the same column names: 'id', 'email', 'first\_name', and 'last\_name'.

id	email	first_name	last_name
----	-------	------------	-----------

# Hibernate จะ generate คำสั่ง SQL เพื่อสร้าง Data Table และ constraints ของ Table ให้ตาม JPA Entity Class

Hibernate:

```
create table users (  
    id bigint identity not null,  
    email varchar(255) not null,  
    first_name varchar(255) not null,  
    last_name varchar(255) not null,  
    primary key (id)  
)
```

```
6 @Data  
7 @Entity  
8 @Table(name = "users")  
9 public class User {  
10  
11     @Id  
12     @GeneratedValue(strategy = GenerationType.IDENTITY)  
13     private Long id;  
14  
15     @Column(name = "first_name", nullable = false)  
16     private String firstName;  
17  
18     @Column(name = "last_name", nullable = false)  
19     private String lastName;  
20  
21     @Column(name = "email", nullable = false, unique = true)  
22     private String email;  
23 }  
24
```

Hibernate:

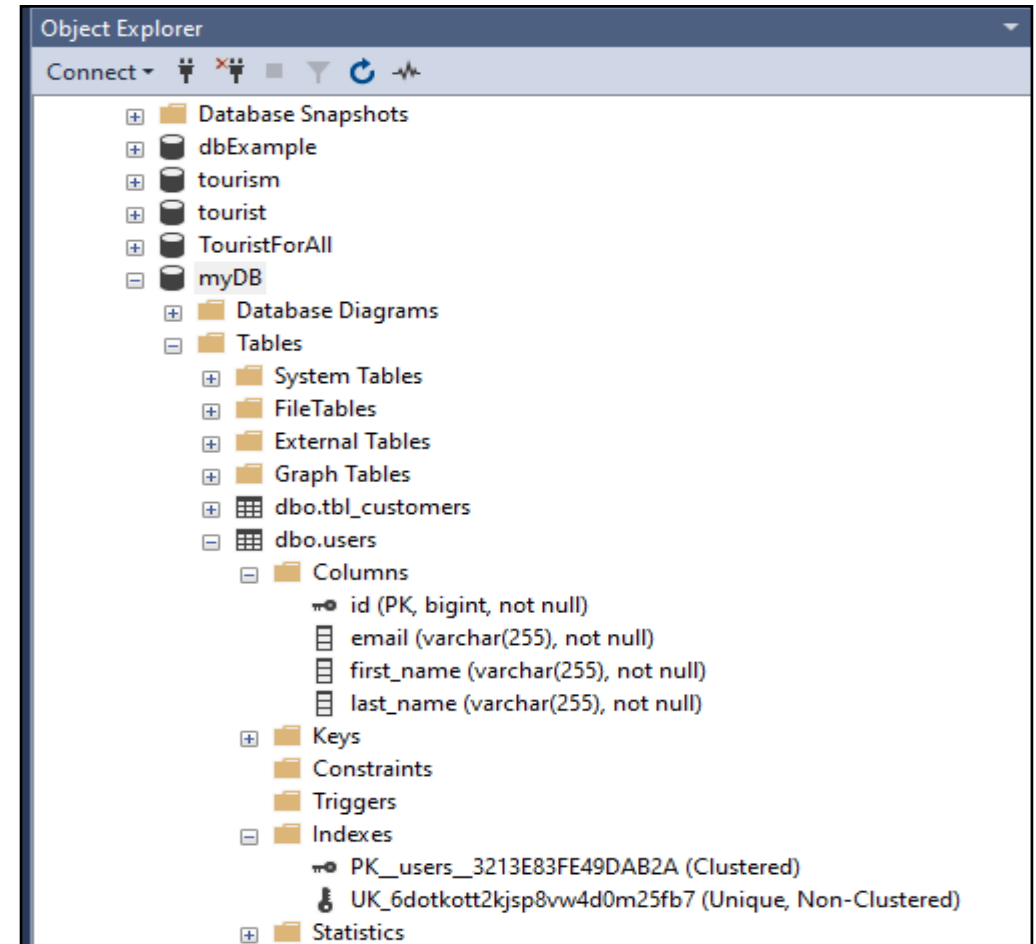
```
alter table users  
drop constraint UK_6dotkott2kjsp8vw4d0m25fb7
```

Hibernate:

```
alter table users  
add constraint UK_6dotkott2kjsp8vw4d0m25fb7 unique (email)
```

Hibernate: create sequence tbl\_customers\_seq start with 1 increment by 50

วิธีการเช็ค ว่า MS SQL Server ได้สร้าง Data Table, Constraints และ Sequencer ตามที่ Hibernate กำหนดหรือไม่?



# Hibernate จะ generate คำสั่ง SQL เพื่อสร้าง Sequencer สำหรับใช้ใน MS SQL Database

```
demo - DemoApplication [Spring Boot App] D:\spring-tool-suite-4-4.14.0.RELEASE-e4.23.0-win32.win32.x86_64.self-extracting\c
)
Hibernate:
    create table users (
        id bigint identity not null,
        email varchar(255) not null,
        first_name varchar(255) not null,
        last_name varchar(255) not null,
        primary key (id)
    )
Hibernate:
    alter table users
        drop constraint UK_6dotkott2kjsp8vw4d0m25fb7
Hibernate:
    alter table users
        add constraint UK_6dotkott2kjsp8vw4d0m25fb7 unique (email)
Hibernate: create sequence tbl_customers_seq start with 1 increment by 50
```

```
SQLQuery1.sql - lo...host.myDB (sa (56))* - X
SELECT name
FROM sys.sequences;

SELECT current_value
FROM sys.sequences
WHERE name = 'tbl_customers_seq';

SELECT NEXT VALUE FOR tbl_customers_seq;
```

100 %

Results Messages

	(No column name)
1	1

# คำสั่ง SQL ในการ insert data เข้าตาราง

## คำสั่ง SQL ในการ ค้นหาข้อมูลจากตาราง (select)

```
USE [myDB]
GO

INSERT INTO [dbo].[users]
    ([email]
    ,[first_name]
    ,[last_name])
VALUES
    ('rongviri@yahoo.com'
    ,'songsakdi'
    ,'rongviriyapanish')
GO
```

```
SELECT TOP (1000) [id]
    ,[email]
    ,[first_name]
    ,[last_name]
FROM [myDB].[dbo].[users]
```

```
/****** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [id]
    ,[email]
    ,[first_name]
    ,[last_name]
FROM [myDB].[dbo].[users]
```

100 %

Results Messages

	id	email	first_name	last_name
1	1	rongviri@yahoo.com	songsakdi	rongviriyapanish

# แบบฝึกหัดให้นักศึกษาสร้าง REST Controller, Repository for User และ คลาส User Entity ที่ Map กับ ตาราง User

```
1 package com.example.demo;
2
3
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.web.bind.annotation.*;
6
7 import java.util.List;
8
9 @RestController
10 @RequestMapping("/api/users")
11 public class UserController {
12
13     @Autowired
14     private UserRepository userRepository;
15
16     @GetMapping
17     public List<User> getAllUsers() {
18         return userRepository.findAll();
19     }
20
21     @PostMapping
22     public User createUser(@RequestBody User user) {
23         return userRepository.save(user);
24     }
25 }
26
```

```
1 package com.example.demo;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 @Repository
7 public interface UserRepository extends JpaRepository<User, Long> {
8 }
9
```

สามารถดู list ของ methods ที่สามารถเรียกใช้จาก Repository จาก

Ref. <https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html>

```
1 # Microsoft SQL Server configuration
2 spring.cloud.config.enabled=false
3 spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
4 spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=myDB;encrypt=true;trustServerCertificate=true;
5 spring.datasource.username=sa
6 spring.datasource.password=rongviri126
7
8 # Enable SQL logging for debugging (optional)
9 spring.jpa.show-sql=true
10 spring.jpa.properties.hibernate.format_sql = true
11
12 ## Hibernate Properties
13 # The SQL dialect makes Hibernate generate better SQL for the chosen database
14 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.SQLServer2012Dialect
15
16 # Hibernate ddl auto (create, create-drop, validate, update)
17 spring.jpa.hibernate.ddl-auto = update
```



# ทดสอบ insert User ใหม่โดยใช้ POSTMAN เรียก Post Method 'createUser(@RequestBody User user)'

Overview | **POST Post data** | GET Get data | + | No environment

REST API basics: CRUD, test & variable / **Post data** | Save | Share

POST | http://localhost:8080/api/users | กำหนด Headers | Send

Params | Authorization | **Headers (10)** | Body | Scripts | Tests | Settings | Cookies

Headers 9 hidden

	Key	Value	Description		Bulk Edit	Presets
<input checked="" type="checkbox"/>	Content-Type	application/json				
	Key	Value	Description			

Body | Cookies | Headers (5) | Test Results (1/1) | 200 OK • 494 ms • 239 B • Save Response

Pretty | Raw | Preview | Visualize | JSON |

```
1 {
2   "id": 2,
3   "firstName": "John",
4   "lastName": "Doe",
5   "email": "john.doe@example.com"
6 }
```

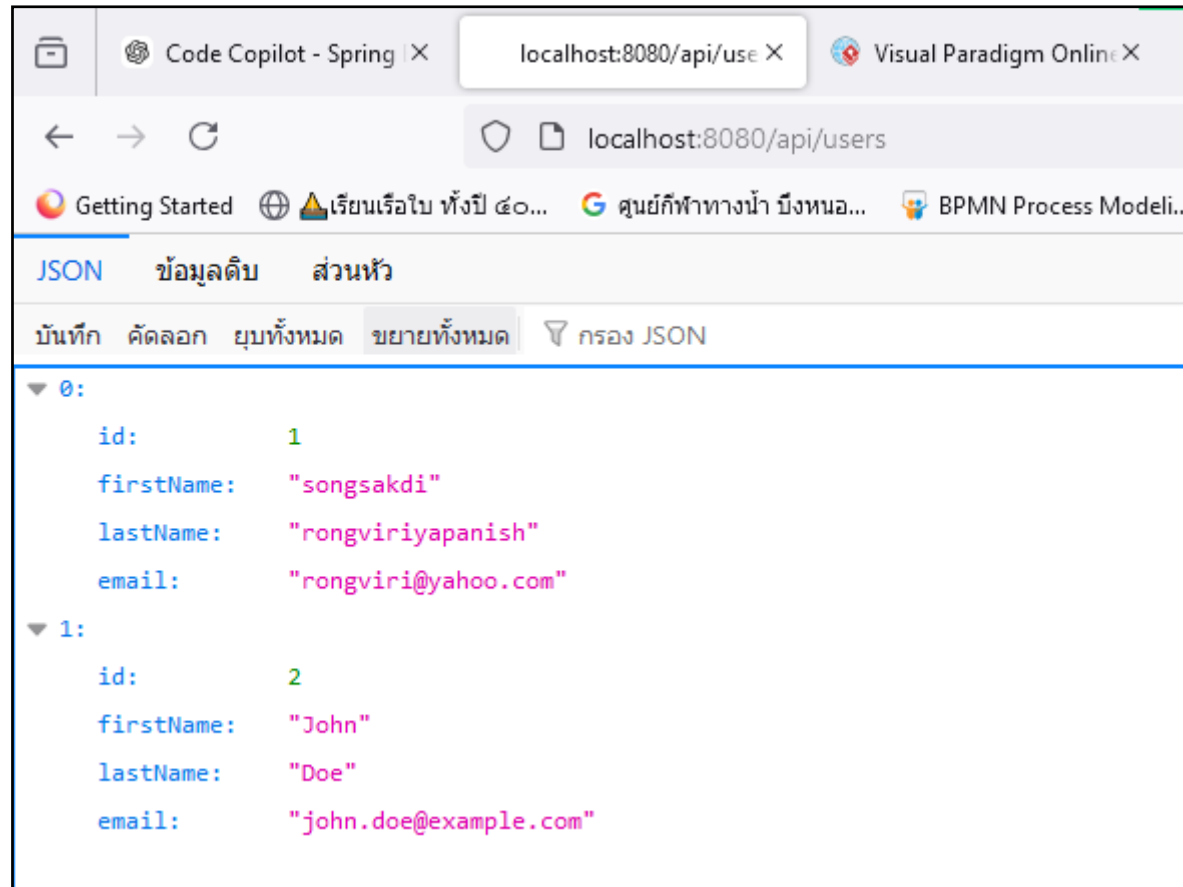
กำหนด Body

Params | Authorization | Headers (10) | **Body** | Scripts | Tests | Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL | JSON

```
1
2 "firstName": "John",
3 "lastName": "Doe",
4 "email": "john.doe@example.com"
5
```

# ทดสอบเรียกใช้ GET Method เพื่อดูรายการข้อมูล Users โดยใช้ Browser



# แบบฝึกหัด การรันคำสั่ง SQL เพื่อจัดเตรียม Master Data

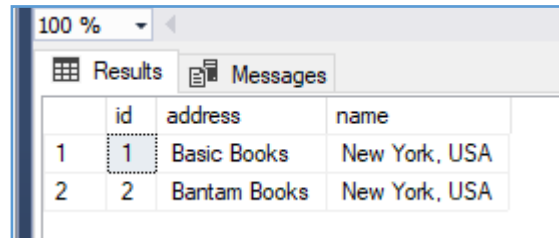
```
USE [myDB]
GO

INSERT INTO [dbo].[publishers]
    ([address]
    ,[name])
VALUES
    ('Basic Books'
    ,'New York, USA')
GO

INSERT INTO [dbo].[publishers]
    ([address]
    ,[name])
VALUES
    ('Bantam Books'
    ,'New York, USA')
GO
```

Table 13.1. Sample data for Publisher

Name	Address
Bantam Books	New York, USA
Basic Books	New York, USA



The screenshot shows a SQL Server Results window with a zoom level of 100%. It displays two rows of data in a table with columns 'id', 'address', and 'name'. The first row has id 1, address 'Basic Books', and name 'New York, USA'. The second row has id 2, address 'Bantam Books', and name 'New York, USA'.

	id	address	name
1	1	Basic Books	New York, USA
2	2	Bantam Books	New York, USA

- ให้นักศึกษารันคำสั่ง sql “insert into” เพื่อสร้างข้อมูล Publisher ตามรูป ลงในตาราง ‘publishers’

# แบบฝึกหัด การรันคำสั่ง SQL เพื่อจัดเตรียม Master Data

```
USE [myDB]
GO

INSERT INTO [dbo].[authors]
    ([name])
VALUES
    ('Immanuel Kant')
GO

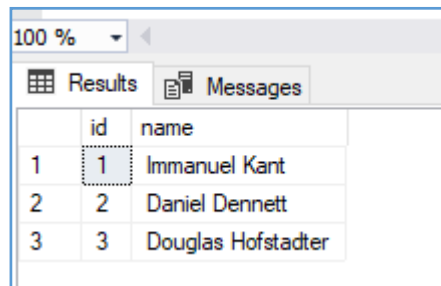
INSERT INTO [dbo].[authors]
    ([name])
VALUES
    ('Daniel Dennett')
GO

INSERT INTO [dbo].[authors]
    ([name])
VALUES
    ('Douglas Hofstadter')
GO
```

Table 13.3. Sample data for Author

Author ID	Name
1	Daniel Dennett
2	Douglas Hofstadter
3	Immanuel Kant

- ให้นักศึกษารันคำสั่ง sql “insert into” เพื่อสร้างข้อมูลผู้แต่ง (Authors) ตามรูป ลงในตาราง ‘authors’



	id	name
1	1	Immanuel Kant
2	2	Daniel Dennett
3	3	Douglas Hofstadter

# แบบฝึกหัด การรันคำสั่ง SQL เพื่อจัดเตรียม Master Data

```
USE [myDB]
GO
INSERT INTO [dbo].[books]
    ([isbn]
    ,[title]
    ,[year]
    ,[publisher_id])
VALUES
    ('0553345842'
    ,'The Mind's I'
    ,1982
    ,2)
GO
INSERT INTO [dbo].[books]
    ([isbn]
    ,[title]
    ,[year])
VALUES
    ('1463794762'
    ,'The Critique of Pure Reason'
    ,2011)
GO
INSERT INTO [dbo].[books]
    ([isbn]
    ,[title]
    ,[year])
VALUES
    ('1928565379'
    ,'The Critique of Practical Reason'
    ,2009)
GO
INSERT INTO [dbo].[books]
    ([isbn]
    ,[title]
    ,[year]
    ,[publisher_id])
VALUES
    ('0465030793'
    ,'I Am A Strange Loop'
    ,2000, 1)
GO
```

Table 13.2. Sample data for Book

ISBN	Title	Year	Authors	Publisher
0553345842	The Mind's I	1982	1, 2	Bantam Books
1463794762	The Critique of Pure Reason	2011	3	
1928565379	The Critique of Practical Reason	2009	3	
0465030793	I Am A Strange Loop	2000	2	Basic Books

	id	isbn	title	year	publisher_id
1	1	0553345842	The Mind's I	1982	2
2	5	1463794762	The Critique of Pure Reason	2011	NULL
3	7	1928565379	The Critique of Practical Reason	2009	NULL
4	9	0465030793	I Am A Strange Loop	2000	1

- ให้นักศึกษารันคำสั่ง  
sql “insert  
into” เพื่อสร้าง  
ข้อมูลหนังสือ  
(Books) ตามรูป  
ลงในตาราง  
‘books’

# แบบฝึกหัด การรันคำสั่ง SQL เพื่อจัดเตรียม Master Data

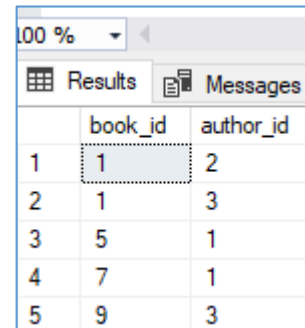
```
USE [myDB]
GO
INSERT INTO [dbo].[book_authors]
    ([book_id]
    ,[author_id])
VALUES
    (1,2)
GO
INSERT INTO [dbo].[book_authors]
    ([book_id]
    ,[author_id])
VALUES
    (1,3)
GO
INSERT INTO [dbo].[book_authors]
    ([book_id]
    ,[author_id])
VALUES
    (5,1)
GO
INSERT INTO [dbo].[book_authors]
    ([book_id]
    ,[author_id])
VALUES
    (7,1)
GO
INSERT INTO [dbo].[book_authors]
    ([book_id]
    ,[author_id])
VALUES
    (9,3)
GO
```

Table 13.2. Sample data for Book

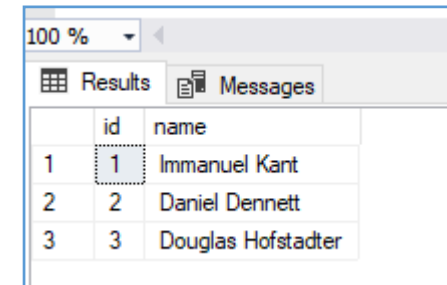
ISBN	Title	Year	Authors	Publisher
0553345842	The Mind's I	1982	1, 2	Bantam Books
1463794762	The Critique of Pure Reason	2011	3	
1928565379	The Critique of Practical Reason	2009	3	
0465030793	I Am A Strange Loop	2000	2	Basic Books

Table 13.3. Sample data for Author

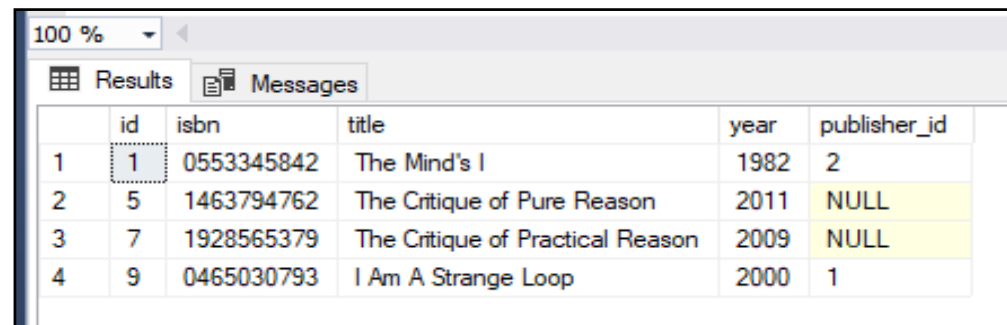
Author ID	Name
1	Daniel Dennett
2	Douglas Hofstadter
3	Immanuel Kant



	book_id	author_id
1	1	2
2	1	3
3	5	1
4	7	1
5	9	3



	id	name
1	1	Immanuel Kant
2	2	Daniel Dennett
3	3	Douglas Hofstadter



	id	isbn	title	year	publisher_id
1	1	0553345842	The Mind's I	1982	2
2	5	1463794762	The Critique of Pure Reason	2011	NULL
3	7	1928565379	The Critique of Practical Reason	2009	NULL
4	9	0465030793	I Am A Strange Loop	2000	1

- ให้นักศึกษารันคำสั่ง  
sql “insert into”  
เพื่อสร้างข้อมูลผู้แต่ง  
หนังสือ ของหนังสือต่างๆ  
(Book-Authors)  
ตามรูป ลงในตาราง  
‘book\_authors’

### Hibernate:

```
create table authors (  
    id bigint identity not null,  
    name varchar(255) not null,  
    primary key (id)  
)
```

### Hibernate:

```
create table book_authors (  
    book_id bigint not null,  
    author_id bigint not null  
)
```

### Hibernate:

```
create table books (  
    id bigint identity not null,  
    isbn varchar(255) not null,  
    title varchar(255) not null,  
    year int not null,  
    publisher_id bigint,  
    primary key (id)  
)
```

### Hibernate:

```
create table publishers (  
    id bigint identity not null,  
    address varchar(255) not null,  
    name varchar(255) not null,  
    primary key (id)  
)
```

### Hibernate:

```
alter table books  
drop constraint UK_kibbepcitr0a3cpk3rfr7nihn
```

### Hibernate:

```
alter table books  
add constraint UK_kibbepcitr0a3cpk3rfr7nihn  
unique (isbn)
```

### Hibernate:

```
alter table book_authors  
add constraint FKo86065vktj3hy1m7syr9cn7va  
foreign key (author_id)  
references authors
```

# SQL Commands ที่ Hibernate generated

ให้อัตโนมัติ

### Hibernate:

```
alter table book_authors  
add constraint FKbhqtkv2cndf10uhtknaqbyo0a  
foreign key (book_id)  
references books
```

### Hibernate:

```
alter table books  
add constraint FKayy5edfrqnegqj3882nce6qo8  
foreign key (publisher_id)  
references publishers
```

# โค้ด Java สำหรับ JPA Entity 'Book'

```
1 package com.example.demo;
2
3 import jakarta.persistence.*;
4 import lombok.Data;
5
6 import java.util.List;
7
8 @Data
9 @Entity
10 @Table(name = "books")
11 public class Book {
12
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Long id;
16
17     @Column(nullable = false)
18     private String title;
19
20     @Column(nullable = false, unique = true)
21     private String ISBN;
22
23     @Column(nullable = false)
24     private Integer year;
25
26     // Many-to-One relationship with Publisher
27     @ManyToOne
28     @JoinColumn(name = "publisher_id", nullable = false)
29     private Publisher publisher;
30
31     // Many-to-Many relationship with Author
32     @ManyToMany
33     @JoinTable(
34         name = "book_authors",
35         joinColumns = @JoinColumn(name = "book_id"),
36         inverseJoinColumns = @JoinColumn(name = "author_id")
37     )
38     private List<Author> authors;
39 }
```



# โค้ด Java สำหรับ JPA Entity 'Publisher'

```
1 package com.example.demo;
2
3
4 import jakarta.persistence.*;
5 import lombok.Data;
6
7 @Data
8 @Entity
9 @Table(name = "publishers")
10 public class Publisher {
11
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Long id;
15
16     @Column(nullable = false)
17     private String name;
18
19     @Column(nullable = false)
20     private String address;
21 }
22
```

# โค้ด Java สำหรับ JPA Entity 'Author'

```
1 package com.example.demo;
2
3 import jakarta.persistence.*;
4 import lombok.Data;
5
6 //import java.util.List;
7
8 @Data
9 @Entity
10 @Table(name = "authors")
11 public class Author {
12
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Long id;
16
17     @Column(nullable = false)
18     private String name;
19
20     // Many-to-Many relationship with Book
21     // @ManyToMany(mappedBy = "authors")
22     // private List<Book> books;
23 }
```

# ตัวอย่างการ Implement REST API ที่เรียกใช้ Repository / DTO ในการค้นหาหนังสือจาก Database

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.Optional;

@RestController
@RequestMapping("/api/books")
public class BookController {
    @Autowired
    private BookRepository bookRepository;
    // GET method to retrieve book by title
    @GetMapping("/getBook")
    public ResponseEntity<?> getBookByTitle(@RequestParam(name="title") String title) {
        // Find the book by title
        Optional<Book> optionalBook = bookRepository.findByTitle(title);
        // If the book is found, return it with HTTP 200 (OK) status
        if (optionalBook.isPresent()) {
            Book book = optionalBook.get();
            return ResponseEntity.ok(book);
        }

        // If the book is not found, return HTTP 404 (Not Found)
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("The book with the given title not found");
    }
}
```

# ตัวอย่างการ Implement REST API ที่เรียกใช้ Repository / DTO

## ในการค้นหาหนังสือจาก Database

- Implement REST API เพื่อสืบค้นหนังสือด้วยชื่อหนังสือ โดยเรียกผ่าน URL ต่อไปนี้  
<http://localhost:8080/api/books/getBook?title=The Critique of Pure Reason>
- หากเจอหนังสือให้ return รายละเอียดของหนังสือที่พบ เป็น json ดังรูป

```
{
  "id": 5,
  "title": "The Critique of Pure Reason",
  "year": 2011,
  "publisher": null,
  "authors": [
    {
      "id": 1,
      "name": "Immanuel Kant"
    }
  ],
  "isbn": "1463794762"
}
```

- หากไม่พบให้ return status 404 พร้อม Message "The book with the given title not found"

สถานะ	วิธีการ	โดเมน	ไฟล์	ตัวเริ่มต้น	ชนิด	ถ่ายโอนแล้ว	ขนาด	ส่วนหัว	ดูก็	คำขอ	การตอบสนอง	การจับเวลา
404	GET	localhost:8080	getBook?title=A Test Book	document	html	200 B	39 B	HTML				
	GET	localhost:8080	favicon.ico	FaviconLoader.sy...	json	133 B (ถูกแย่งชิง)	133 B				The book with the given title not found	

# ตัวอย่างการ Implement REST API ที่เรียกใช้ Repository / DTO ในการบันทึกข้อมูลหนังสือใหม่ลง Database

- Implement REST API

เพื่อบันทึกข้อมูลหนังสือใหม่ลงใน DB โดยเรียกผ่าน URL ต่อไปนี้  
**http://localhost:8080/api/books/add**

- หากหนังสือเป็นหนังสือใหม่ที่ยังไม่มีใน DB ให้นำไปบันทึกเป็นข้อมูลใหม่ในตาราง **books** พร้อม return HTTP Status Code เป็น **201** พร้อมรายละเอียดหนังสือใหม่ที่บันทึก

```
// POST method to create a new book with authors
@PostMapping("/add")
public ResponseEntity<?> addNewBook(@RequestBody BookRequest bookRequest) {
    // Check if a book with the same title and ISBN exists
    Optional<Book> existingBook = bookRepository.findByTitle(bookRequest.getTitle());

    if (existingBook.isPresent()) {
        // Return 409 Conflict if the book already exists
        return ResponseEntity.status(HttpStatus.CONFLICT)
            .body("Book with title '" + bookRequest.getTitle() + "' already exists.");
    }

    // Save the Publisher (if it exists already, it can be fetched here)
    Publisher publisher = publisherRepository.findById(bookRequest.getPublisher().getId())
        .orElse(bookRequest.getPublisher()); // Save new publisher if not exists

    // Save the Authors (save only if they don't exist)
    List<Author> authors = authorRepository.saveAll(bookRequest.getAuthors());

    // Create the new book entity
    Book newBook = new Book();
    newBook.setTitle(bookRequest.getTitle());
    newBook.setISBN(bookRequest.getISBN());
    newBook.setYear(bookRequest.getYear());
    newBook.setPublisher(publisher);
    newBook.setAuthors(authors);

    // Save the new book to the database
    bookRepository.save(newBook);

    // Return HTTP 201 Created response with the new book
    return ResponseEntity.status(HttpStatus.CREATED).body(newBook);
}
```

# โค้ด POST Method สำหรับ Save New Book

```
// POST method to create a new book with authors
@PostMapping("/add")
public ResponseEntity<?> addNewBook(@RequestBody BookRequest bookRequest) {
    // Check if a book with the same title and ISBN exists
    Optional<Book> existingBook = bookRepository.findByTitle(bookRequest.getTitle());

    if (existingBook.isPresent()) {
        // Return 409 Conflict if the book already exists
        return ResponseEntity.status(HttpStatus.CONFLICT)
            .body("Book with title '" + bookRequest.getTitle() + "' already exists.");
    }
    // Save the Publisher (if it exists already, it can be fetched here)
    Publisher publisher = publisherRepository.findById(bookRequest.getPublisher().getId()).orElse(bookRequest.getPublisher()); // Save new publisher if not exists

    // Save the Authors (save only if they don't exist)
    List<Author> authors = authorRepository.saveAll(bookRequest.getAuthors());

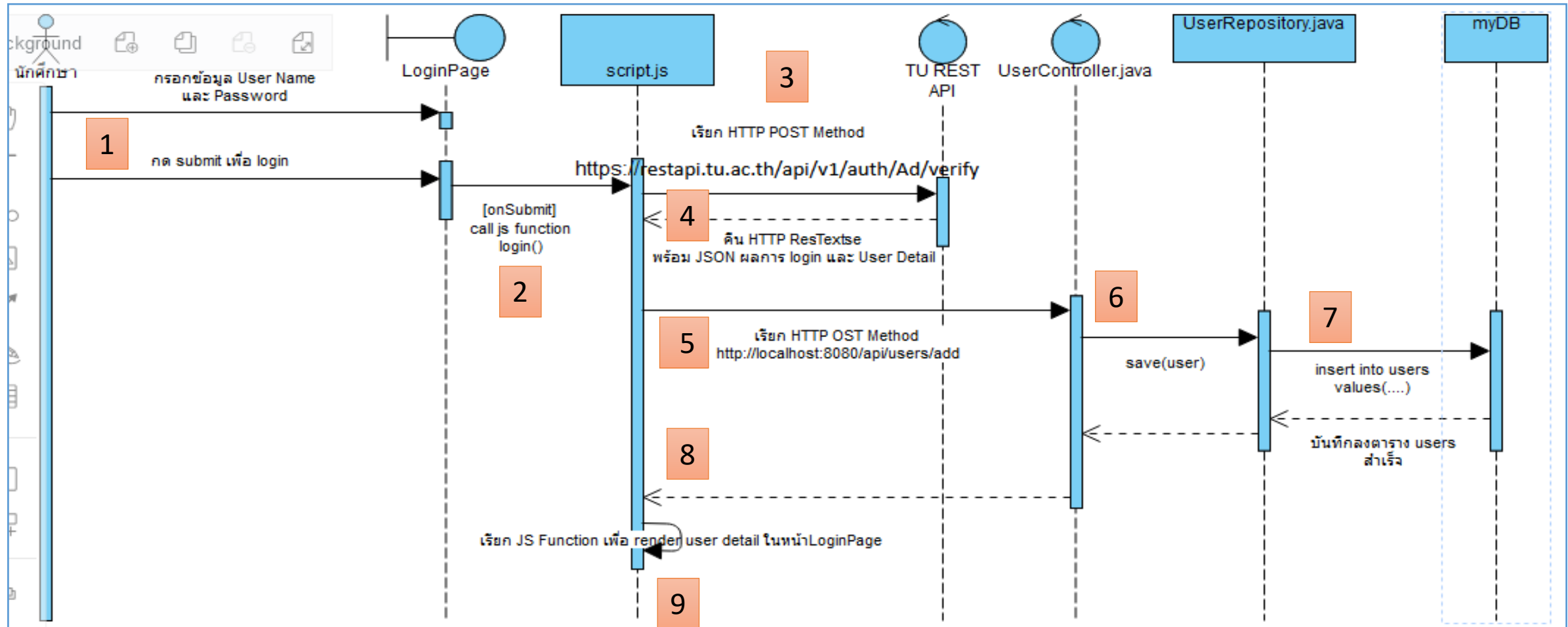
    // Create the new book entity
    Book newBook = new Book();
    newBook.setTitle(bookRequest.getTitle());
    newBook.setISBN(bookRequest.getISBN());
    newBook.setYear(bookRequest.getYear());
    newBook.setPublisher(publisher);
    newBook.setAuthors(authors);

    // Save the new book to the database
    bookRepository.save(newBook);

    // Return HTTP 201 Created response with the new book
    return ResponseEntity.status(HttpStatus.CREATED).body(newBook);
}
```

- ให้ใช้โค้ด **Frontend Node.JS Project** ของนักศึกษาเองที่ทำ **Assignment** รายบุคคลครั้งที่ 1 หรือ ใช้โค้ดของอาจารย์ได้ โดยดาวน์โหลดได้ที่ URL:  
<https://moodle.tu.ac.th/mod/resource/view.php?id=310615>
- แผนภาพแสดงลำดับ **Interaction** ระหว่าง **Components** ต่างๆเพื่อ **Implement User Story** “บันทึกข้อมูลรายละเอียด **User** หลัง **Login**สำเร็จ”

# แผนภาพแสดง Interaction ระหว่าง User และ System และ Interaction ระหว่าง Components ประเภทต่างๆ





# ให้ Design DTO/Entity Class เก็บข้อมูล User ลงใน Database

```
{  
  "status": true,  
  "message": "Success",  
  "type": "student",  
  "username": "5701010101",  
  "tu_status": "สำเร็จการศึกษา",  
  "statusid": "40",  
  "displayname_th": "นักศึกษา ทดสอบ",  
  "displayname_en": "nuksuksa todsob",  
  "email": "email.std@dome.tu.ac.th",  
  "department": "สาขาวิชาการบริหารเทคโนโลยี",  
  "faculty": "วิทยาลัยนวัตกรรม"  
}
```

ให้นำข้อมูลผลลัพธ์ที่ได้จาก TU REST API หลังเช็ค login ไปเก็บลงในตาราง students ในฐานข้อมูล myDB ดังนี้

- username นำไปบันทึกเป็นค่าในคอลัมน์ username เก็บชื่อสำหรับ login
- type นำไปบันทึกเป็นค่าในคอลัมน์ type เก็บประเภทของ user ค่าที่เป็นไปได้จะเป็น student หรือ employee
- displayname\_en นำไปบันทึกเป็นชื่อนามสกุลภาษาอังกฤษ เก็บใส่คอลัมน์ engName
- email นำไปบันทึกเป็นค่าในคอลัมน์ email เก็บอีเมล
- faculty นำไปบันทึกเป็นค่าในคอลัมน์ faculty เก็บชื่อคณะที่นักศึกษาสังกัด

# Develop โค้ดเพื่อทำ Logic ส่วน การประมวลผล ดังนี้

1. ส่วนของฟังก์ชัน js ให้เพิ่มโค้ดเพื่อเช็คเงื่อนไขจากผลลัพธ์ที่ return จาก TU REST API โดยให้เช็ค status ต้องมีค่าเท่ากับ true และ message ที่ return จาก TU REST API ต้องมีค่าเป็น “success” จึงจะ เรียก REST API ที่นักศึกษา develop เพื่อ save ข้อมูลลงในตาราง users ในฐานข้อมูล และ นำชื่อและนามสกุลภาษาไทยของ user พร้อมคณะที่สังกัดไปแสดงที่หน้า Login
2. หากค่า status มีค่าเป็น false ให้นำค่า message ที่ return จาก TU REST API ไปแสดงให้ user เห็นผ่าน console โดยใช้ฟังก์ชัน javascript ชื่อ alert(<string>) โดยให้นำค่า message ที่ return จาก TU REST API ให้ user เห็นดังรูป

## Login

Username:

Password:

Login

Success

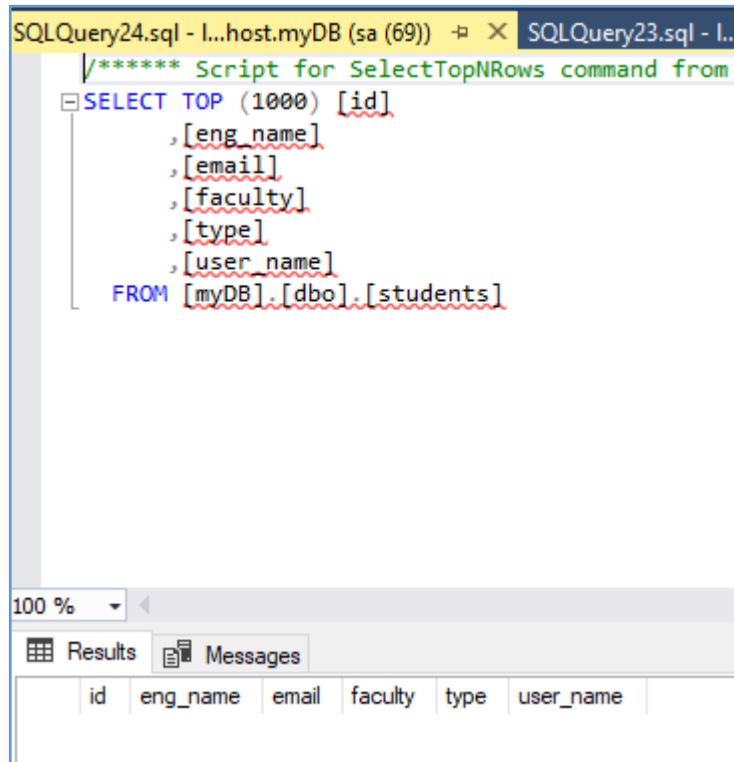
ทรงศักดิ์ รongวิริยะพานิช,คณะวิทยาศาสตร์และเทคโนโลยี

www.w3schools.com

Error: ไม่สามารถ Login ได้สำเร็จ

ตกลง

# ให้นักศึกษาแสดงผลลัพธ์เมื่อสร้าง DTO, Entity Class, Repository และ REST Controller สำเร็จ



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query editor contains the following SQL code:

```
/****** Script for SelectTopNRows command from
```

```
SELECT TOP (1000) [id]
```

```
    ,[eng_name]
```

```
    ,[email]
```

```
    ,[faculty]
```

```
    ,[type]
```

```
    ,[user_name]
```

```
FROM [myDB].[dbo].[students]
```

The results pane at the bottom shows a table with the following columns:

id	eng_name	email	faculty	type	user_name
----	----------	-------	---------	------	-----------

- 1) เมื่อรัน Spring Boot App จะเกิดตาราง students เปล่า  
ที่มีรายละเอียดดังรูป ( 2 คะแนน )

# ให้นักศึกษาแสดงผลลัพธ์เมื่อสร้าง DTO, Entity Class, Repository และ REST Controller สำเร็จ

2) ให้ใช้ POSTMAN เรียก API ทดสอบ  
การเพิ่ม Student ใหม่ลงในตาราง  
students 1 รายใหม่  
สำเร็จได้ดังรูป ( 2 คะแนน )

REST API basics: CRUD, test & variable / New Request

POST http://localhost:8080/api/students/add

Body (raw):

```
1 {"username": "rongviri",
2  "type": "student",
3  "displayname_en": "songsakdi rongviriyanish",
4  "email": "rongviri@yahoo.com",
5  "faculty": "faculty of science and technology"
6 }
```

Response: 201 Created - 68 ms - 337 B

Body (pretty):

```
1 {
2   "id": 1,
3   "username": "rongviri",
4   "type": "student",
5   "displayname_en": "songsakdi rongviriyanish",
6   "email": "rongviri@yahoo.com",
7   "faculty": "faculty of science and technology"
8 }
```

Results						
	id	eng_name	email	faculty	type	user_name
1	1	songsakdi rongviriyanish	rongviri@yahoo.com	faculty of science and technology	student	rongviri

ให้นักศึกษาแสดงผลลัพท์เมื่อทดสอบ Login สำเร็จจะแสดงรายละเอียดของ User พร้อม Status Success พร้อมทั้งมีการ Save User ลงใน ตาราง students หากไม่สำเร็จแสดง Error Message (2 คะแนน)

The screenshot shows a web browser window with the address bar at `localhost:3000`. The page title is "Login Page". The main content area displays a login form with the following elements:

- Login** (Section Header)
- Username:** Input field containing "rongviri"
- Password:** Input field with masked characters "\*\*\*\*\*"
- Login** (Blue button)
- Success** (Text)
- ทรังศักดิ์ ร่องวิริยะพานิช,คณะวิทยาศาสตร์และเทคโนโลยี (Text)

To the right of the login form, an error message is displayed in a white box with a black border:

- www.w3schools.com
- Error: ไม่สามารถ Login ได้สำเร็จ
- ตกลง** (Blue button)

The Windows taskbar at the bottom shows the time as 1:55 AM on 04 พ.ย. 2567.

# ประกาศ Class WebConfig

เพื่อกำหนด CORS ([Cross-Origin Resource Sharing](#))

เพื่อให้ Frontend รันที่ <http://localhost:3000/> เรียก CRUD API (Spring Boot App)

The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer displays the project structure for 'demo [boot] [devtools]'. The 'src/main/java' directory contains the 'com.example.demo' package, which includes various Java files like Author.java, Book.java, Customer.java, etc. The 'WebConfig.java' file is selected in the Package Explorer.

The main editor window shows the code for 'WebConfig.java'. The code defines a Spring Configuration class that implements 'WebMvcConfigurer'. It includes an '@Override' method 'addCorsMappings' that configures CORS for all endpoints ('/\*\*'). The allowed origins are 'http://localhost:3000', 'http://node-server:3000', and 'http://localhost:8080'. The allowed methods are 'GET', 'POST', 'PUT', and 'DELETE'. The allowed headers are set to '\*' to allow all headers.

```
1 package com.example.demo;
2
3 import org.springframework.context.annotation.Configuration;
4
5
6
7 @Configuration
8 public class WebConfig implements WebMvcConfigurer {
9
10     @Override
11     public void addCorsMappings(CorsRegistry registry) {
12         registry.addMapping("/**") // Allow all endpoints
13             .allowedOrigins("http://localhost:3000", "http://node-server:3000", "http://localhost:8080")
14             .allowedMethods("GET", "POST", "PUT", "DELETE") // Allow these HTTP methods
15             .allowedHeaders("*"); // Allow all headers
16     }
17 }
```

The bottom of the screen shows the Console window. It displays the output of the application startup, including the message 'demo - DemoApplication [Spring Boot App] D:\spring-tool-suite-4-4.14.0.RELEASE-e4.23.0-win32.win32.x86\_64.self-extracting\contents\sts-4.14.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre' and the status of the application components, such as 'com.example.demo.DemoApplication : Started' and 'o.s.web.servlet.DispatcherServlet : Complete'.

```

fetch('https://restapi.tu.ac.th/api/v1/auth/Ad/verify', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Application-Key': 'TU5e4eb2cbe5154ae0e2c8e0114774e4cc9badea2ae91819780fddfcc35244f3'
  },
  body: JSON.stringify(jsonObject)
})
.then(response => response.json())
.then(data => {
  document.getElementById('message').innerText = data.message;
  let text = data.displayname_th + "," + data.organization;
  document.getElementById('userDetail').innerText = text;

  // Check if `message` attribute exists and its value is "Success"
  if (data.message === "Success") {
    // Define the JSON data to be sent in the new POST request
    const studentData = {
      "username": "terdsak",
      "type": "employee",
      "displayname_en": "terdsak rongviriyapanish",
      "email": "terdsak@yahoo.com",
      "faculty": "faculty of science and technology"
    };
  }
});

```

คัดลอกโค้ด



// Send a new POST request to the Local server

Using Code Copilot (ChatGPT) เพื่อ generate JS code ส่วนเรียกใช้ API เพื่อ Save User ลง Table

I have this code:

```

fetch('https://restapi.tu.ac.th/api/v1/auth/Ad/verify', {
  method: 'POST', headers: { 'Content-Type': 'application/json',
  'Application-Key':
  'TU5e4eb2cbe5154ae0e2c8e0114774e4cc9badea2ae9181978
  0fddfcc35244f3f7c299bd99c0fc9071898547cd02ca7e6b' },
  body: JSON.stringify(jsonObject) }) .then(response =>
  response.json()) .then(data => {
  document.getElementById('message').innerText =
  data.message; let text = data.displayname_th + "," +
  data.organization;
  document.getElementById('userDetail').innerText = text; })
I want to add more javascript code to check whether json
string return from fetch has an attribute named 'message' with
a value equl to "Success". If it is the case, send a HTTP
request with POST METHOD to a REST API at URL:
"http://localhost:8080/api/students/add". I want to send a
json within the body of that HTTP POST Method request. The
json string is as follows : {"username": "terdsak", "type":
"employee", "displayname_en" : "terdsak rongviriyapanish",
"email": "terdsak@yahoo.com", "faculty": "faculty of science
and technology" }

```



เมื่อกด login และ check login ผ่าน ใน Data Table จะมี ข้อมูล User เพิ่มดังรูป

```
// Send a new POST request to the Local server
fetch('http://localhost:8080/api/students/add', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(studentData)
})
.then(response => response.json())
.then(result => {
  console.log('Student added successfully:', result);
})
.catch(error => {
  console.error('Error adding student:', error);
});

}

});
.catch(error => {
  console.error('Error:', error);
});
```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane displays the database structure for 'myDB', including tables like 'dbo.authors', 'dbo.book\_authors', 'dbo.books', 'dbo.publishers', and 'dbo.students'. The 'dbo.students' table is selected, and its columns are listed: 'id' (PK, bigint, not null), 'eng\_name' (varchar(255), not null), 'email' (varchar(255), not null), 'faculty' (varchar(255), not null), 'type' (varchar(255), not null), and 'user\_name' (varchar(255), not null). On the right, the 'Query Editor' pane shows a SQL query: 'SELECT TOP (1000) [id], [eng\_name], [email], [faculty], [type], [user\_name] FROM [myDB].[dbo].[students]'. Below the query, the 'Results' pane displays the data returned by the query in a table format.

	id	eng_name	email	faculty	type	user_name
1	3	terdsak rongviriyapanish	terdsak@yahoo.com	faculty of science and technology	employee	terdsak
2	4	songsakdi rongviriyapanish	rongviri@yahoo.com	faculty of science and technology	employee	rongviri
3	5	songsakdi rongviriyapanish	rongviri@yahoo.com	faculty of science and technology	employee	rongviri

## Login

Username:

Password:

Login

Success

ทรงศักดิ์ รongviriyapanish, คณะ  
วิทยาศาสตร์และเทคโนโลยี



localhost:3000

90%

startedเรียนเรือใบ ห้างปี ๔๐...ศูนย์กีฬาทางน้ำ บึงหนอง...BPMN Process Modeli...All BPMN 2.0 Level 1 E...Extracting Value from ...ACM Selects: Getting ...

ที่ค้นหาอื่น ๆ

ตัวตรวจสอบคอนโซลตัวบันทึกเครือข่ายตัวแก้ไขสไตล์ประสิทธิภาพหน่วยความจำ

กรอง URL

ปิดใช้งานแคช

ไม่มีการควบคุมปริมาณ

ทั้งหมดHTMLCSSJSXHRแบบอักษรภาพสื่อWSอื่น ๆ

สถานะ	วิธี	โดเมน	ไฟล์	ตัวเริ่มต้น	ชนิด	ถ่ายโอนแ...	ขนาด
200	GET	local...	/	document	html	1.26 kB (...)	941 B
304	GET	local...	styles.css	stylesheet	css	ถูกแคช	730 B
200	GET	local...	script.js	script	js	3.11 kB (...)	2.78 kB
	GET	local...	favicon.ico	FaviconLoader.s...	html	150 B (ถู...	150 B
200	P...	resta...	verify	script.js:17 (fetch)	json	1.28 kB	473 B
200	O...	resta...	verify	fetch	html	797 B	0 B
201	P...	local...	add	script.js:43 (fetch)	json	459 B	168 B

ส่วนหัว

ดูก็ได้ค่าขอการตอบสนองการจับเวลา

กรองส่วนหัว

ปิดกั้นส่งใหม่

POST http://localhost:8080/api/students/add

สถานะ201

รุ่นHTTP/1.1

ถ่ายโอนแล้ว459 B (ขนาด 168 B)

นโยบาย Referrerstrict-origin-when-cross-origin

การแปลงที่อยู่ DNSระบบ

ส่วนหัวการตอบสนอง (291 B)

Access-Control-Allow-Origin: http://localhost:3000

Connection: keep-alive

Content-Type: application/json

Date: Thu, 07 Nov 2024 13:40:12 GMT

Keep-Alive: timeout=60

Transfer-Encoding: chunked

Vary: Origin

Vary: Access-Control-Request-Method

Vary: Access-Control-Request-Headers

ส่วนหัวคำร้องขอ (496 B)

Accept: \*/\*

Accept-Encoding: gzip, deflate, br, zstd

Accept-Language: th,en-US;q=0.7,en;q=0.3

Connection: keep-alive

Content-Length: 161

Login

Username:

rongviri

Password:

Login

Success

ทรงศักดิ์ ร่องวิริยะพานิช,คณษ  
วิทยาาสตร์และเทคโนโลยี