บทที่ 5

MVC

(Model–View–Controller)

ธีระยุทธ ทองเครือ

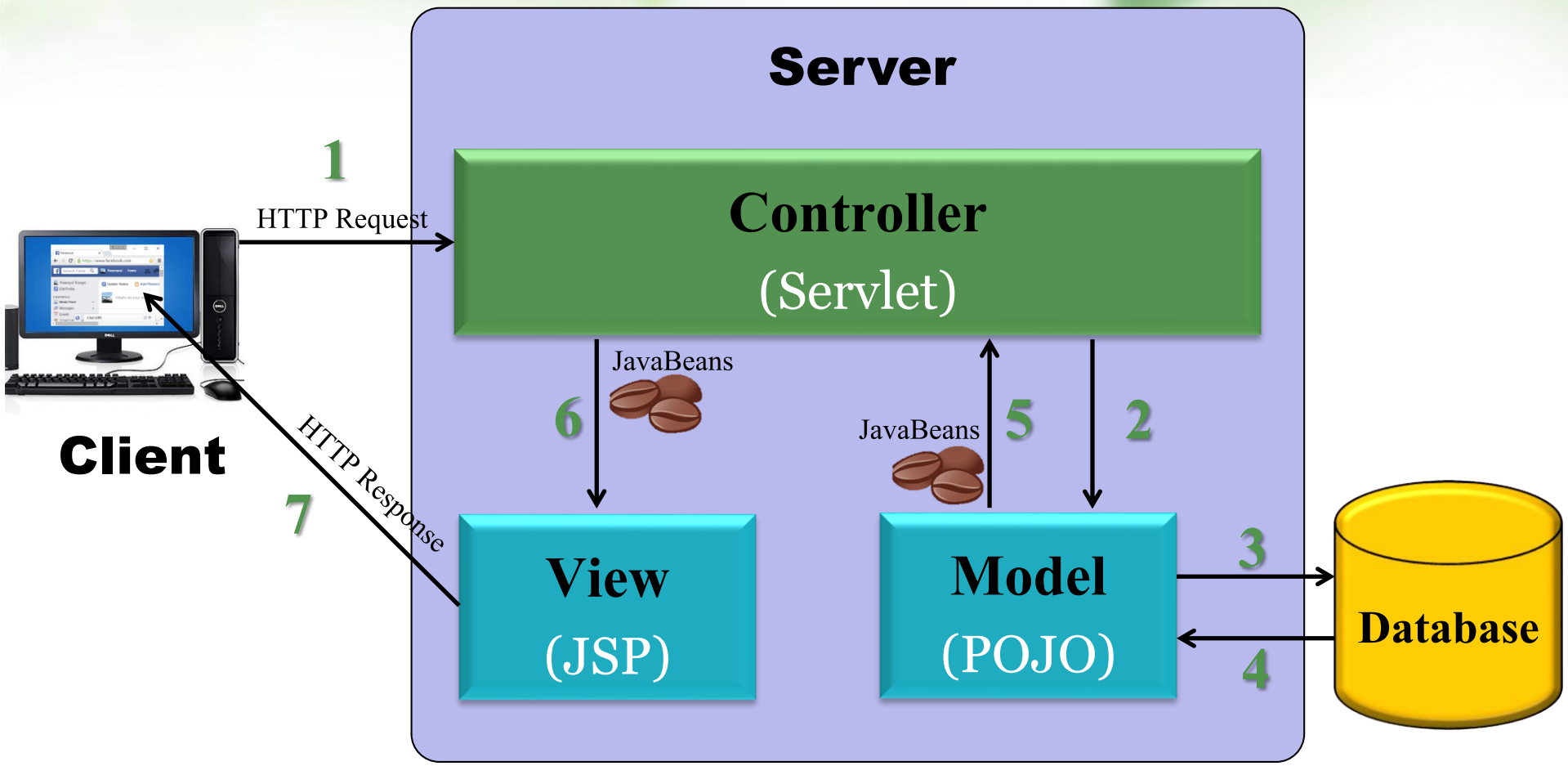
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยขอนแก่น

MVC Architecture

- Model-View-Controller (MVC) คือ การสร้างแอปพลิเคชันโดยแบ่งส่วนการทำงานออกเป็น 3 ส่วน ได้แก่
  - **Controller** ตัวกลางในการรับ-ส่งข้อมูลระหว่างส่วน View และส่วน Model
  - **Model** ทำหน้าประมวลผล Business Logic เช่น การประมวลผลกับฐานข้อมูล ปรับปรุงแก้ไข เพิ่ม หรือดึงข้อมูล ผลลัพธ์จาก Model มักจะเป็นข้อมูลที่จะส่งไปให้ส่วน View แสดงผล
  - **View** คือ ชุดคำสั่งสำหรับ Render ข้อมูลจาก Model เพื่อส่งกลับไปยังผู้ใช้
- เป้าหมายของ MVC
  - การแบ่งส่วนการทำงานของระบบอย่างชัดเจน
  - ง่ายต่อการ maintenance ในอนาคต

MVC Architecture สำหรับ Java Web Apps



* **POJO** ย่อมาจาก Plain Old Java Object ภายในบรรจุเมธอดที่มีการติดต่อกับฐานข้อมูลของระบบ หรือเป็นคำสั่งติดต่อกับระบบภายนอก

JavaBeans

- การแยกส่วนการทำงานแบบ MVC จำเป็นจำต้องมีการลำเลียงข้อมูลจากส่วนหนึ่งไปยังอีกส่วนหนึ่งเสมอ ซึ่งเป็นข้อมูลที่มีหลากหลายชนิด เข้าไว้ด้วยกัน เช่น ข้อมูลลูกค้า มีชื่อ ที่อยู่ เบอร์โทร
- การพัฒนาระบบแบบ MVC จึงต้องสร้างคลาสสำหรับการบรรจุข้อมูล เรียก คลาสนั้นว่า JavaBeans
- คุณสมบัติของคลาส JavaBeans
  - กำหนดการเข้าถึง attribute (หรือตัวแปร) เป็น private
  - มี method สำหรับกำหนดค่าให้กับ attribute ของ Object ชื่อ setXxx( )
  - มี method สำหรับอ่านค่า attribute ของ Object ชื่อ getXxx( )
  - จะมี constructor หรือไม่มีก็ได้ ถ้ามี constructor นั้นจะต้องไม่มีพารามิเตอร์

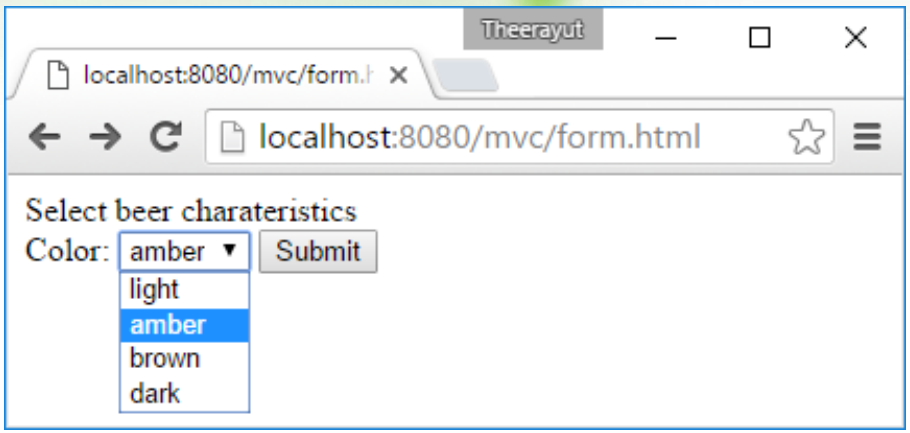
*หมายเหตุ JavaBeans ที่ใช้ข้อมูลใน Session จะต้อง **`implements Serializable`** เสมอ

JavaBeans

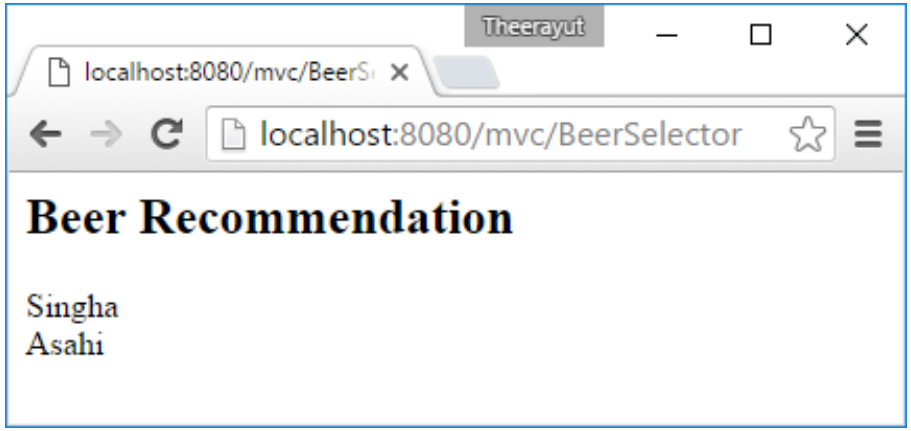
```
public class Triangle {
    private float base;
    private float height;

    public float getBase() {
        return base;
    }
    public void setBase(float base) {
        this.base = base;
    }
    public float getHeight() {
        return height;
    }
    public void setHeight(float height) {
        this.height = height;
    }
}
```

Triangle
- base: int - height: int
+ getBase( ): float + setBase(float): void + getHeight( ): + setHeight(float): void

ระบบแนะนำเครื่องดื่ม

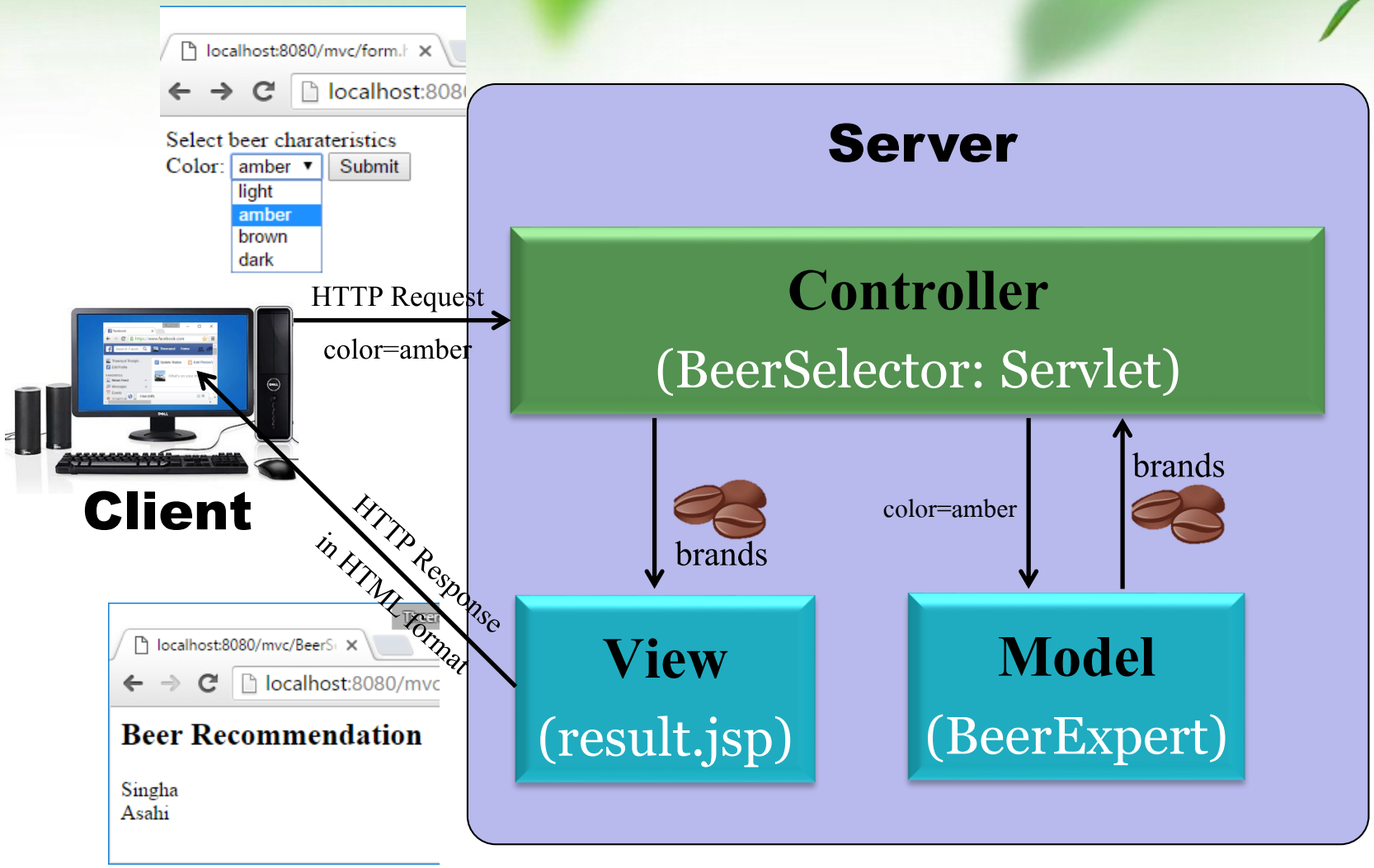


ฟอร์ม HTML ที่จะส่งไปยัง server



ผลลัพธ์ที่สร้างจาก JSP

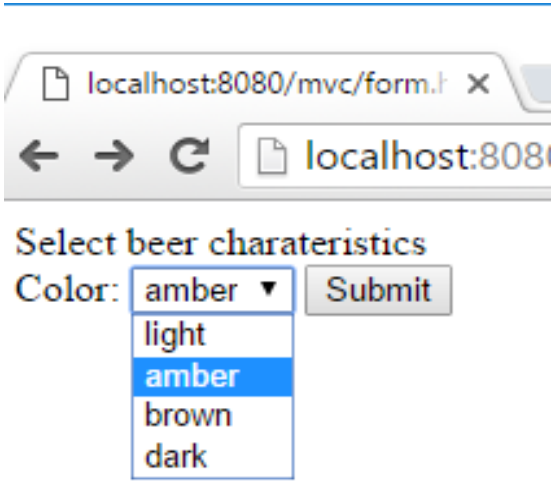
ระบบแนะนำเครื่องดื่ม



ฟอร์มสำหรับส่งข้อมูล (request)

ชื่อ Servlet ที่รับ request

```
<html>
<body>
  <form action="BeerSelector" method="post">
     Select beer charateristics<br>
     Color:
     <select name="color">
        <option value="light">light</option>
        <option value="amber">amber</option>
        <option value="brown">brown</option>
        <option value="dark">dark</option>
     </select>
     <input type="submit">
   </form>
</body>
</html>
```



ส่วน Controller (Servlet)

```
@WebServlet("/BeerSelector")
public class BeerSelector extends HttpServlet {

 void doPost(HttpServletRequest request, HttpServletResponse response){
   String color = request.getParameter("color");

   // ส่วน Model
   BeerExpert beerExpert = new BeerExpert();
   ArrayList<String> brands = beerExpert.getBrands(color);

   request.setAttribute("brands", brands);

   // ส่งไปยังส่วน View
   RequestDispatcher view = request.getRequestDispatcher("result.jsp");
   view.forward(request, response);
 }

}
```

ดึงข้อมูลจาก request object

สร้าง BeerExpert object
และส่งสีที่ผู้ใช้เลือก เพื่อขอข้อมูล

นำผลลัพธ์เก็บไว้ใน request object

ส่ง request และ response ไป render การแสดงผลในส่วน View (JSP)
เป็นการส่งแบบแบบ Request Dispatch
(การส่งต่อ request ไม่สามารถใช้วิธี Send Redirect ได้)

ส่วน Model

```
import java.util.ArrayList;

public class BeerExpert {

    public ArrayList<String> getBrands(String color) {

        ArrayList<String> brands = new ArrayList<String>();

        if (color.equals("amber")) {
            brands.add("Singha");
            brands.add("Asahi");
        } else {
            brands.add("Carlsberg");
            brands.add("Heineken");
            brands.add("Tiger");
        }
        return brands;
    }
}
```

สร้าง object สำหรับเก็บผลลัพธ์

ตรวจสอบเงื่อนไขเพื่อเก็บข้อมูลลงใน object

เพิ่มข้อมูล String ลงใน ArrayList object

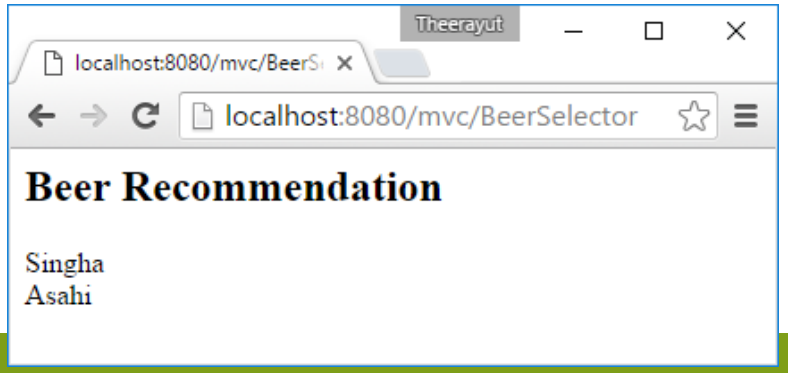
ส่ง ArrayList object กลับ

ส่วน View (result.jsp)

```
<%@page import="java.util.ArrayList"%>
<html>
<body>

<h2>Beer Recommendation</h2>

<%
  ArrayList<?> beerList = (ArrayList<?>)request.getAttribute("brands");

   for(int i=0; i<beerList.size(); i++) {
      out.println(beerList.get(i) + "<br>");
   }
%>

</body>
</html>
```

ดึงผลลัพธ์จาก request object

วนลูปนำ String จาก ArrayList มาสร้างเป็นผลลัพธ์ HTML



ภาพรวมโค้ด

form.html

```
<html>
<body>
  <form action="BeerSelector" method="post">
     Select beer characteristics<br>
     Color:
     <select name="color">
        <option value="light">light</option>
        <option value="amber">amber</option>
        <option value="brown">brown</option>
        <option value="dark">dark</option>
     </select>
     <input type="submit">
   </form>
</body></html>
```

result.jsp

V

```
<%@page import="java.util.ArrayList"%>
<html>
<body>
<h2>Beer Recommendation</h2>
<%
  ArrayList<?> beerList = (ArrayList<?>)request.getAttribute("brands");

  for(int i=0; i<beerList.size(); i++) {
     out.println(beerList.get(i) + "<br>");
  }
%>
</body>
</html>
```

BeerSelector.java

C

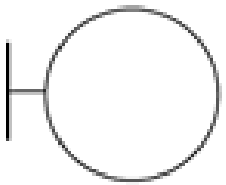
```
@WebServlet("/BeerSelector")
public class BeerSelector extends HttpServlet {
  private static final long serialVersionUID = 1L;

  protected void doPost(HttpServletRequest request, HttpServletResponse
    String color = request.getParameter("color");

    // ส่วนModel
    BeerExpert beerExpert = new BeerExpert();
    ArrayList<String> brands = beerExpert.getBrands(color);

    request.setAttribute("brands", brands);

    // ส่งไปยังส่วนView
    RequestDispatcher view = request.getRequestDispatcher("result.jsp");
    view.forward(request, response);
  }
}
```

BeerExpert.java

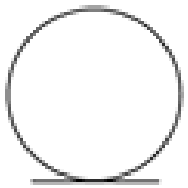
M

```
public class BeerExpert {

    public ArrayList<String> getBrands(String color) {

        ArrayList<String> brands = new ArrayList<String>();

        if (color.equals("amber")) {
            brands.add("Singha");
            brands.add("Asahi");
        } else {
            brands.add("Carlsberg");
            brands.add("Heineken");
            brands.add("Tiger");
        }
        return brands;
    }
}
```

MVC กับสัญลักษณ์ใน UML



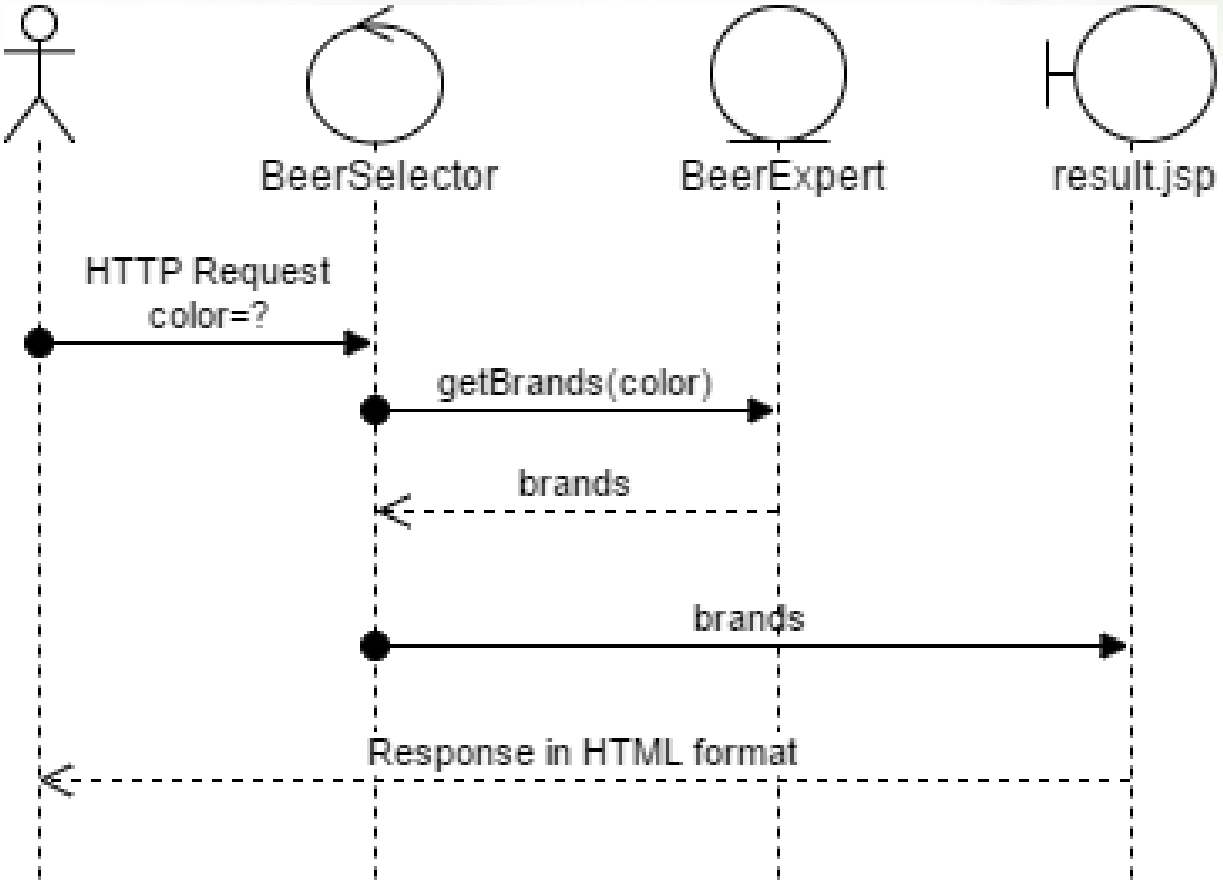
(view)



(model)

(controller)

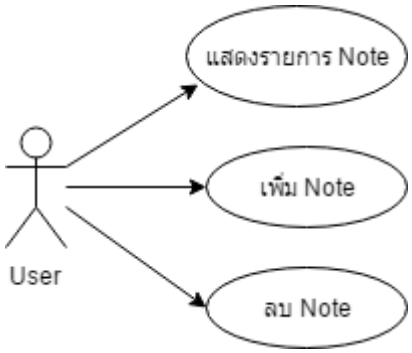
Sequence Diagram ของการแนะนำเครื่องดื่ม



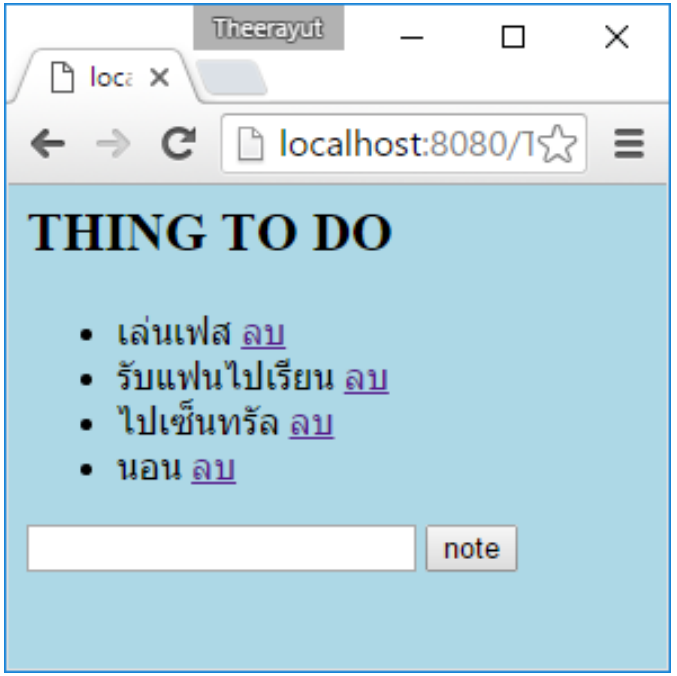
ToDo App

- เว็บแอปพลิเคชัน ToDo ใช้สำหรับบันทึกงานที่ต้องทำในแต่ละวัน ซึ่งออกแบบซอฟท์แวร์ตามสถาปัตยกรรม MVC การทำงานประกอบด้วย แสดงรายการงานที่ต้องทำ และสามารถเลือกลบรายการที่ทำเสร็จแล้วได้

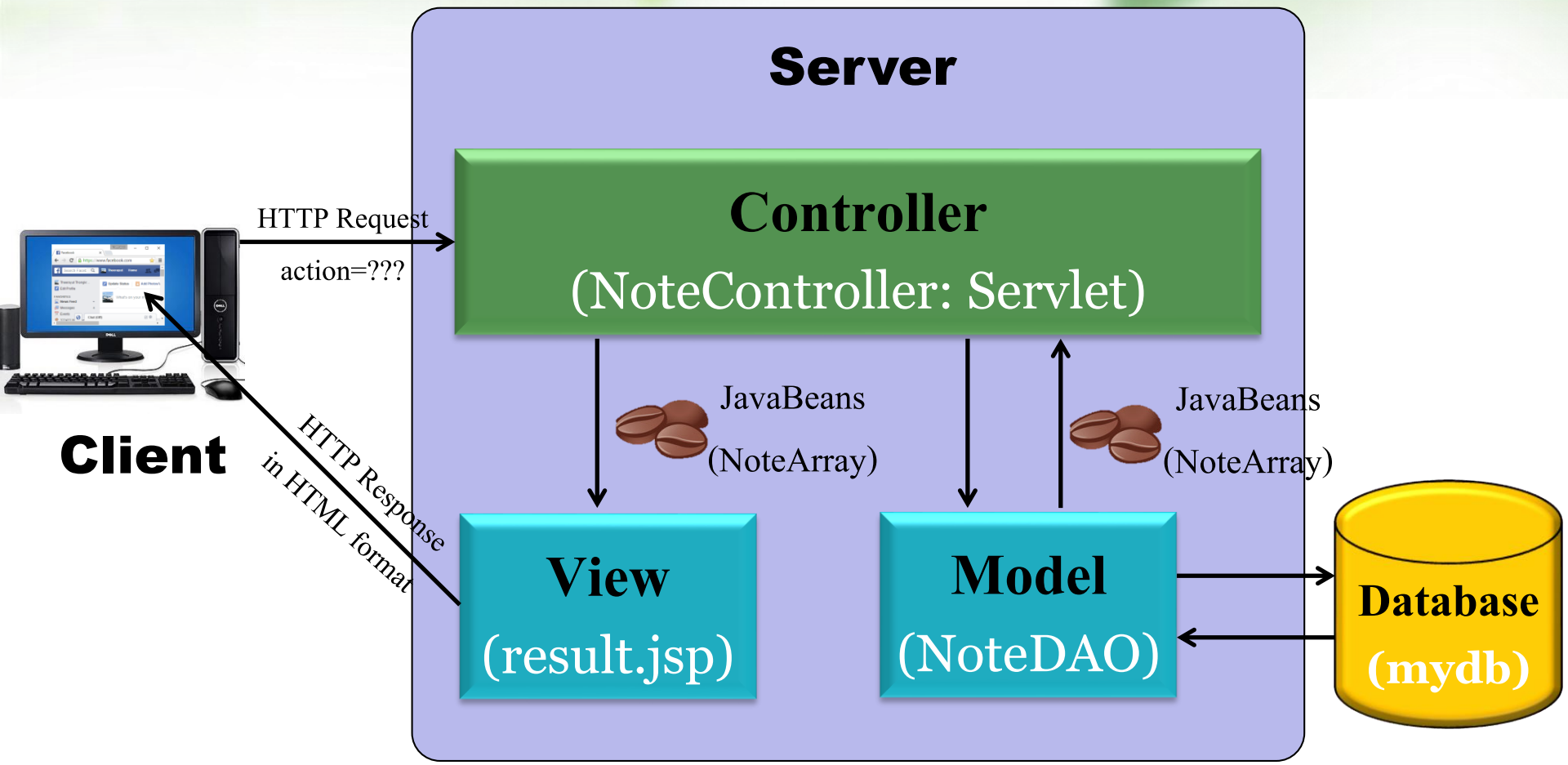
Usecase Diagram



ตัวอย่างหน้าจอ



ToDo App


Server
Controller
(NoteController: Servlet)
HTTP Request
action=???
Client
HTTP Response
in HTML format
JavaBeans
(NoteArray)
JavaBeans
(NoteArray)
View
(result.jsp)
Model
(NoteDAO)
Database
(mydb)

ขั้นตอนการสร้างเว็บแอปแบบ MVC

1. สร้าง Model

2. สร้าง Controller

3. สร้าง View

ส่วน Model

- สร้างฐานข้อมูลใหม่ชื่อ mydb และสร้างตาราง note ตามโครงสร้างที่กำหนด
- สร้าง package model
- สร้างคลาสโมเดล NoteDAO ซึ่งประกอบด้วยฟังก์ชันติดต่อกับฐานข้อมูล
  - สร้าง Constructor ของคลาสที่มีการติดต่อกับฐานข้อมูล
  - สร้างฟังก์ชันติดต่อฐานข้อมูลที่แอปพลิเคชันต้องการ โดยอาจเริ่มต้นที่ฟังก์ชัน SELECT ข้อมูลก่อน
  - ทดสอบการทำงานของฟังก์ชัน อาจเขียนฟังก์ชัน main( ) เพื่อทดสอบบน Console ก่อน
- สร้างคลาส JavaBeans เพื่อใช้บรรจุข้อมูลชื่อ Note กำหนดชนิดและชื่อ attribute ให้ตรงกับตาราง note

โครงสร้างตาราง Note

ชื่อฟิลด์	คำอธิบาย	ชนิดข้อมูล	หมายเหตุ
nid	รหัสโน้ตงาน	INT	คีย์หลัก แบบเพิ่มค่าอัตโนมัติ
task	ชื่องานที่ต้องทำ	VARCHAR(100)	

ส่วน Controller

- สร้าง Servlet ใหม่ชื่อ NoteController

- ที่ method doGet( ) รับ parameter ที่ชื่อ action มาด้วย เพื่อให้มีคลาส Controller ควบคุมเพียงคลาสเดียว

- สร้าง Object NoteDAO และเรียกเมธอดเพื่อขอข้อมูลเก็บลง request object

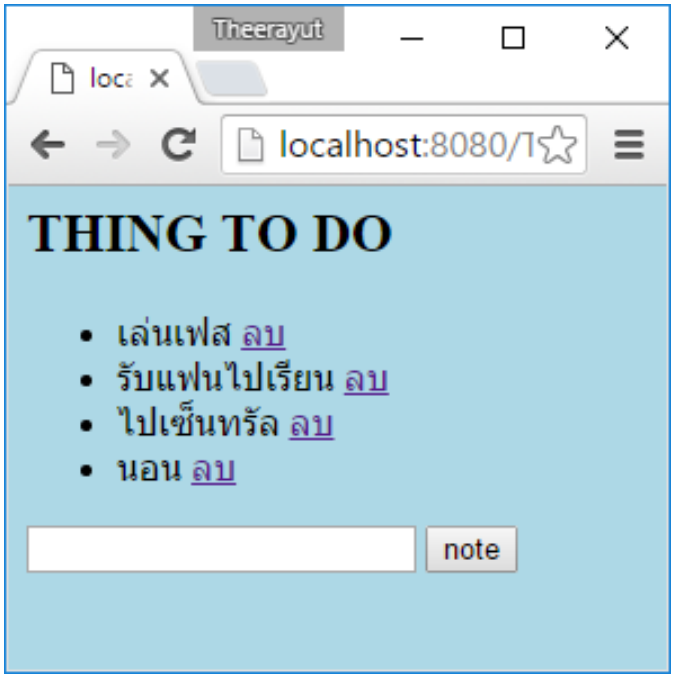
- forward ข้อมูลไปยังส่วน View

ส่วน View

- สร้างไฟล์ result.jsp
  - ใช้คำสั่งดึงข้อมูลจาก request ด้วยเมธอด getAttribute( )
  - แสดงข้อมูลจาก Object ที่ได้
  - แทรก link สำหรับลบ

- สร้าง <form> กำหนด action ให้ส่งไปยัง Controller

- กำหนดช่อง input และปุ่มส่งข้อมูล



โปรดระวัง !!!

- อย่าลืมไฟล์ JDBC Driver (.jar) เพื่อใช้ติดต่อฐานข้อมูล

- ชื่อฐานข้อมูลเปลี่ยนไปแล้ว dbURL ก็ต้องเปลี่ยนด้วย

Usecase แสดงรายการ note

C

NoteController.java

```java
package controller;

import java.io.IOException;

@WebServlet("/NoteController")
public class NoteController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        NoteDAO notedao = new NoteDAO();
        ArrayList<Note> notes = notedao.getAllTask();

        request.setAttribute("notes", notes);

        RequestDispatcher view = request.getRequestDispatcher("result.jsp");
        view.forward(request, response);
    }
}
```

V

result.jsp

```jsp
<%@page import="java.util.ArrayList, model.Note"%>
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf8">
</head>
<body>
<%
    ArrayList<Note> nlist = (ArrayList<Note>)request.getAttribute("notes");
    for (int i=0; i<nlist.size(); i++) {
        Note note = nlist.get(i);
        out.println(note.getTask() + "<br>");
    }
%>
</body>
</html>
```

M

NoteDAO.java

```java
public class NoteDAO {
    public ArrayList<Note> getAllTask() {
        ArrayList<Note> notes = new ArrayList<Note>();

        try {
            Class.forName("com.mysql.jdbc.Driver");
            String dbURL = "jdbc:mysql://localhost/mydb?characterEncoding=utf-8";
            Connection con = DriverManager.getConnection(dbURL, "root", "");
            Statement statement = con.createStatement();
            ResultSet resultSet = statement.executeQuery("select * from note");
            while (resultSet.next()) {
                int nid = resultSet.getInt("nid");
                String task = resultSet.getString("task");
                System.out.println(nid + "," + task);
                Note note = new Note();
                note.setNid(nid);
                note.setTask(task);
                notes.add(note);
            }
        } catch (ClassNotFoundException e) {
            System.err.println("Error loading driver: " + e);
        } catch (SQLException e) {
            System.err.println("Error database connection: " + e);
        }

        return notes;
```

JavaBean

Note.java

```java
package model;

public class Note {
    private int nid;
    private String task;

    public int getNid() {
        return nid;
    }
    public void setNid(int nid) {
        this.nid = nid;
    }
    public String getTask() {
        return task;
    }
    public void setTask(String task) {
        this.task = task;
    }
}
```

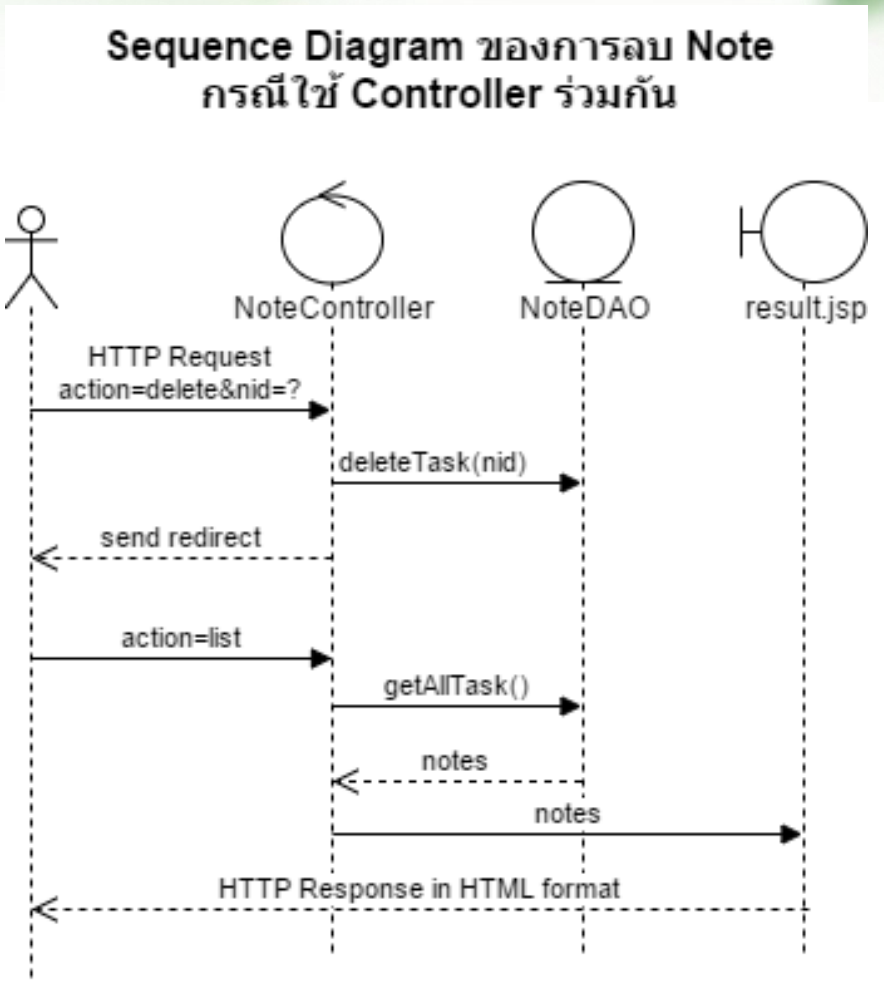
http://localhost:8080/todo/NoteController

http://localhost:8080/todo/NoteController

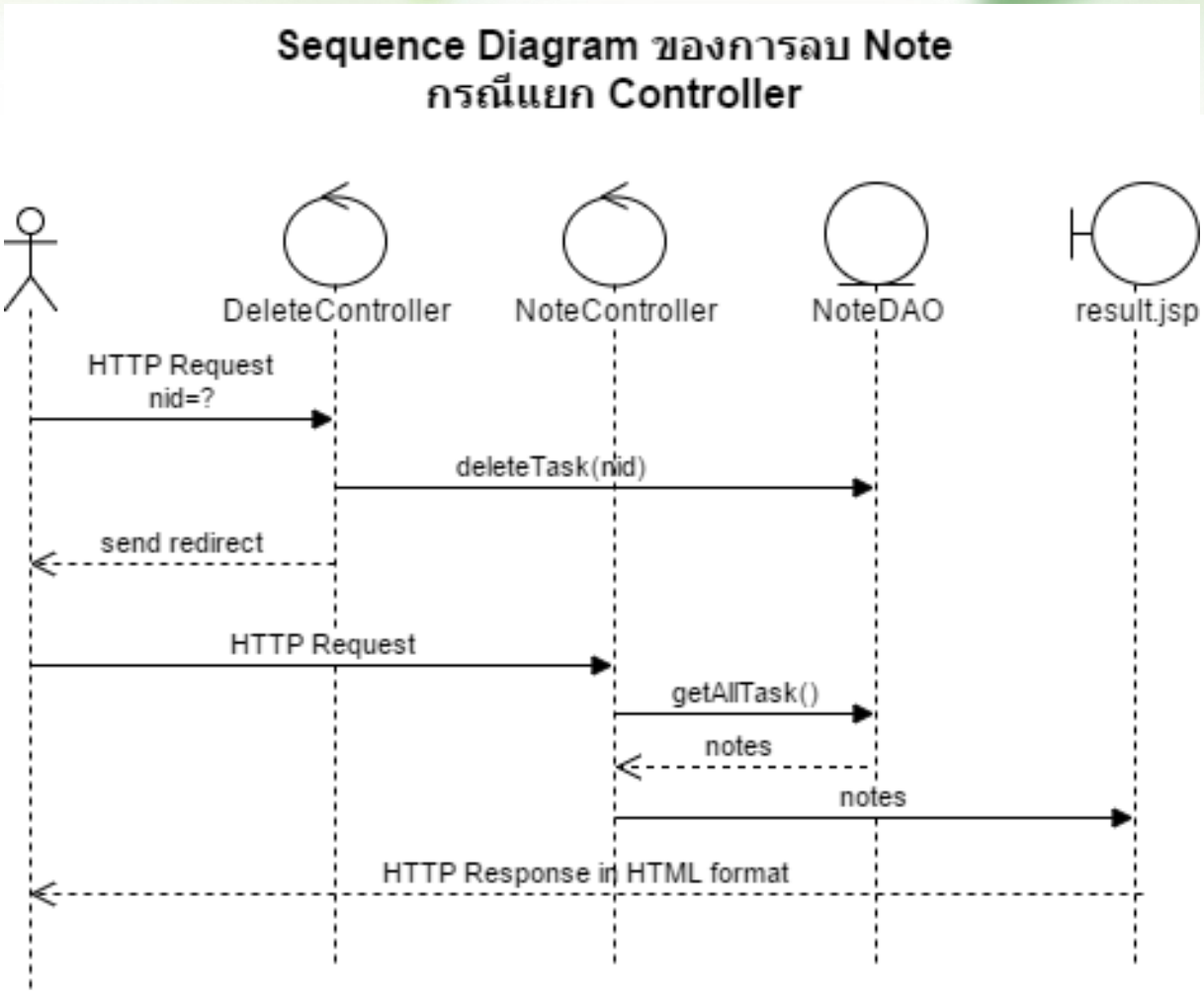
ทดสอบ 1
ทดสอบ 2

Output

Sequence Diagram การลบ Note แบบที่ 1



Sequence Diagram การลบ Note แบบที่ 2



การสร้าง Controller เดียวรับทุก request

- Controller ที่สร้างขึ้นสำหรับรับข้อมูลจากผู้ใช้อาจเขียนเป็นคลาส Servlet เดียว ที่สามารถรับข้อมูลเพื่อมา insert/update/delete ได้ เป็นวิธีการที่ช่วยลดความยุ่งยากในการสร้างหลาย Controller
- หลักการคือ กำหนดให้มี parameter ที่กำหนดวิธีการประมวลผล ซึ่งอาจกำหนดเป็น

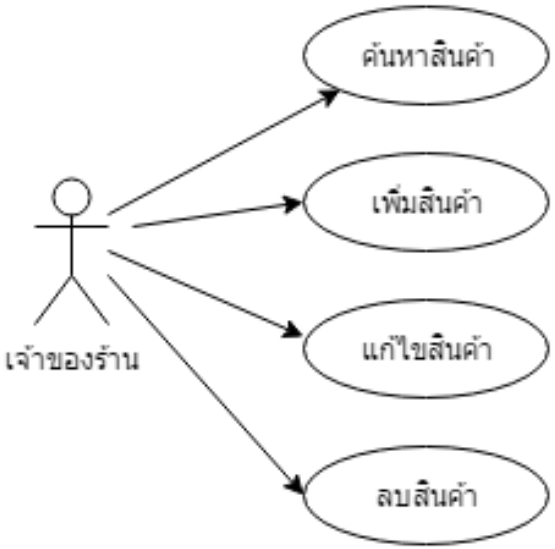
```
<input type="hidden" name="action" value="add">
```

- เมื่อ Servlet รับค่า action จะเชคข้อความก่อนเป็นอันดับแรก แล้วนำไปเข้าเงื่อนไข

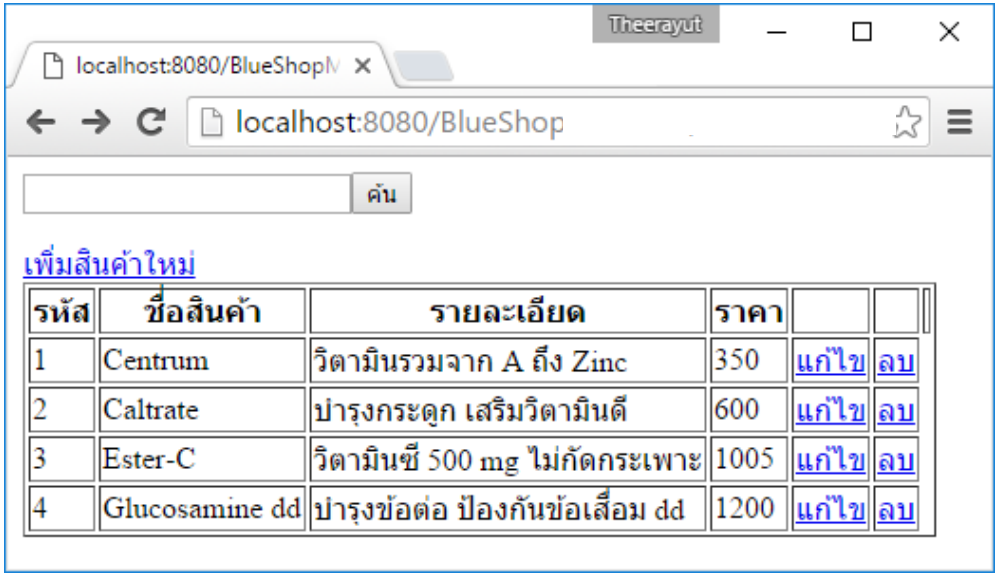
```
protected void service(...) {
    String action = request.getParameter("action");
    ...
    if (action.equals("add")) {
        ...
    } else if (action.equals("edit")) {
        ...
    } else if (action.equals("remove")) {
        ...
    }
}
```

ระบบจัดการข้อมูลสินค้า BlueShop

Use case Diagram

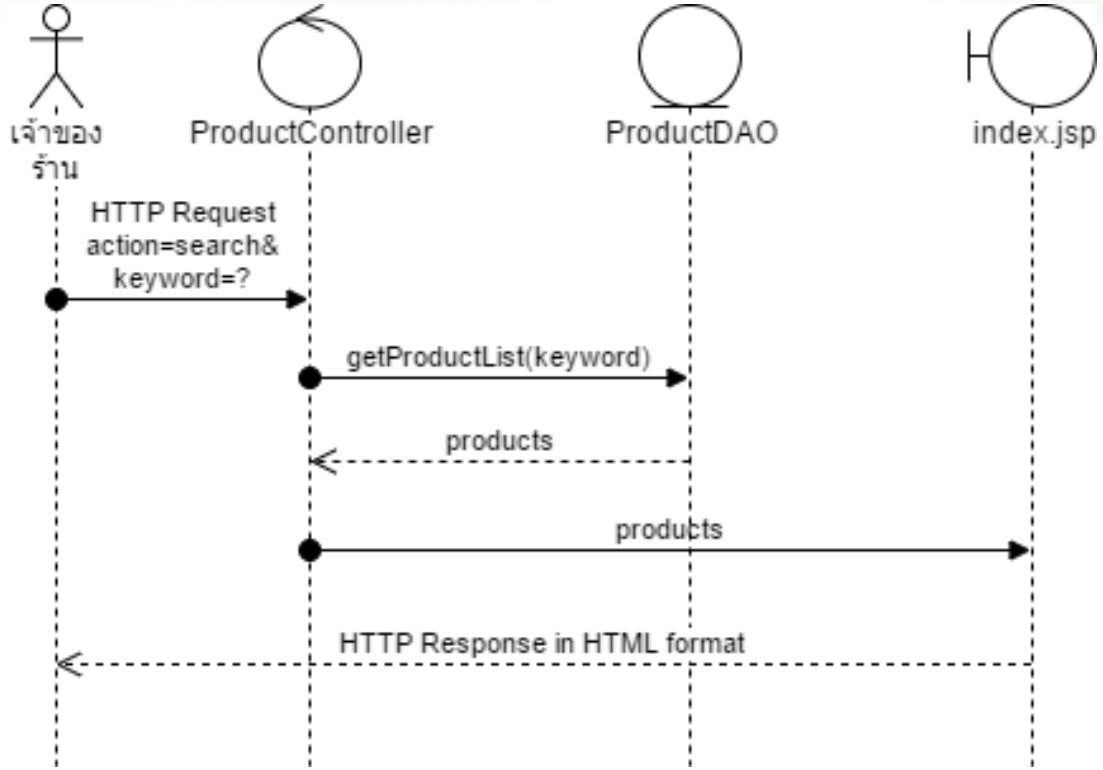


ตัวอย่างหน้าจอของระบบ



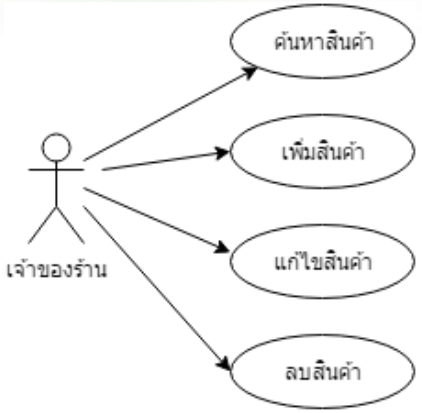
รหัส	ชื่อสินค้า	รายละเอียด	ราคา		
1	Centrum	วิตามินรวมจาก A ถึง Zinc	350	แก้ไข	ลบ
2	Caltrate	บำรุงกระดูก เสริมวิตามินดี	600	แก้ไข	ลบ
3	Ester-C	วิตามินซี 500 mg ไม่กัดกระเพาะ	1005	แก้ไข	ลบ
4	Glucosamine dd	บำรุงข้อต่อ ป้องกันข้อเสื่อม dd	1200	แก้ไข	ลบ

Sequence Diagram ค้นหาสินค้า



Assignment#5

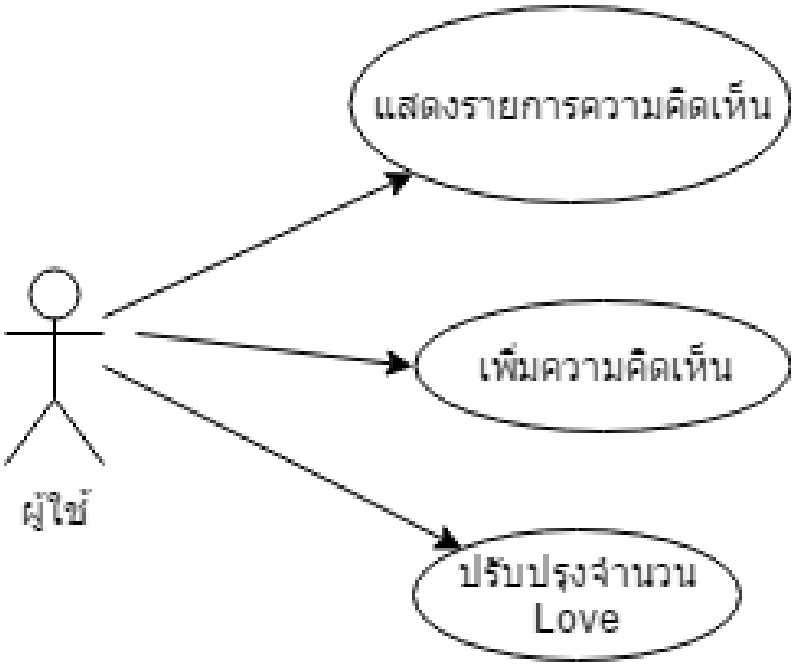
- **พัฒนาระบบจัดการข้อมูลสินค้า BlueShop ตามสถาปัตยกรรมม MVC โดยพัฒนาการทำงาน 3 ส่วน**
  - ค้นหาสินค้า
  - แก้ไขสินค้า
  - ลบสินค้า



- **วาด Sequence Diagram 2 ภาพ ได้แก่ แก้ไขสินค้า และลบสินค้า**
  - วาดด้วยเว็บ draw.io
  - Export เป็นภาพ 2 ภาพเก็บลงใน Eclipse Project โฟลเดอร์ WebContent
- **หมายเหตุ**
  - หากมีเวลาอยากให้ทำเพิ่มสินค้า ฝึกประสบการณ์
  - ส่วน Controller อาจแยกหรือรวมไว้เป็นคลาสเดียวก็ได้
- กำหนดส่งวันพุธที่ 8 มีนาคม 2560 ก่อนเวลา 23.00 น.

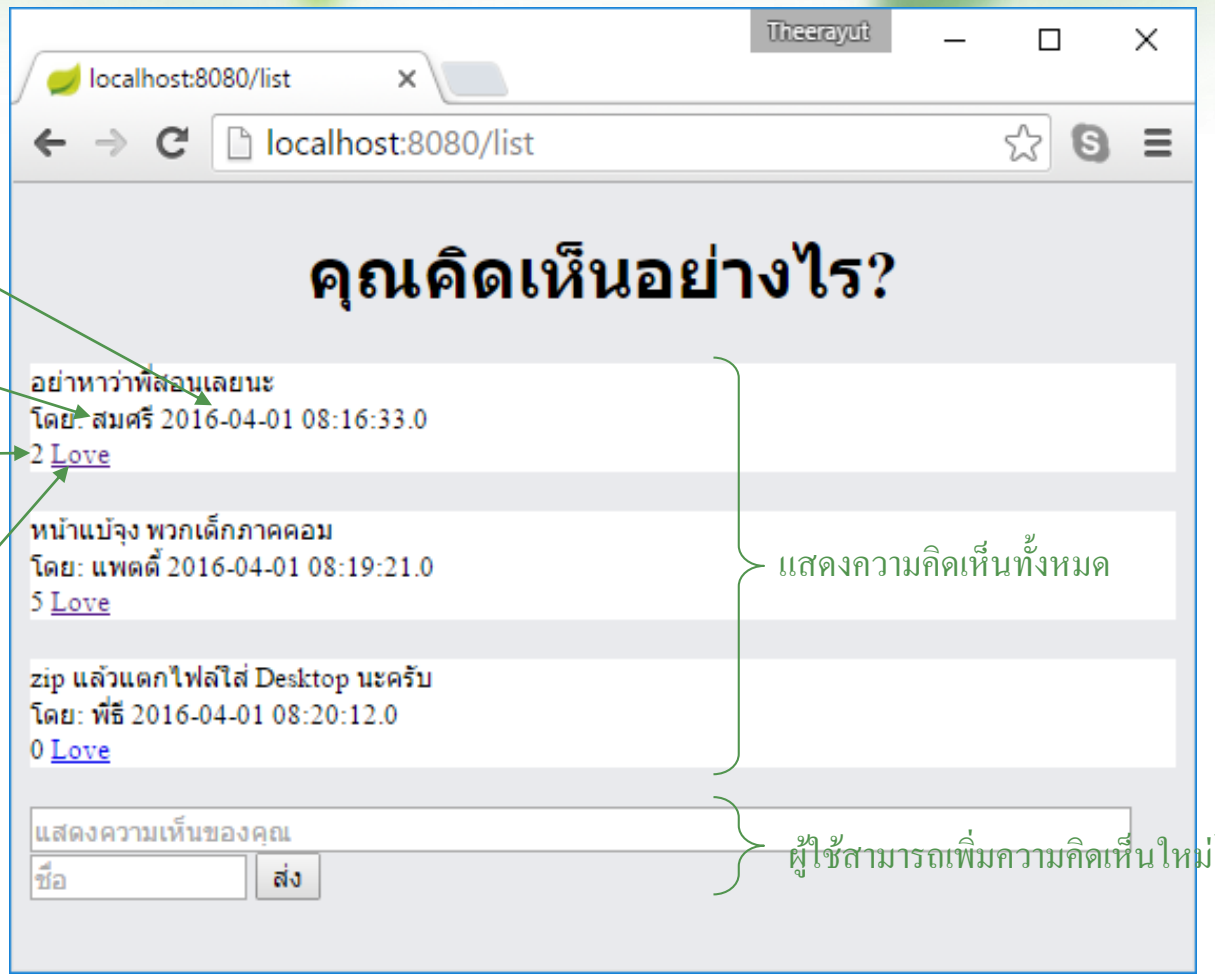
ระบบแสดงความคิดเห็น

จงสร้างระบบแสดงความคิดเห็นจากผู้ใช้ตามสถาปัตยกรรม MVC



Usecase Diagram ของระบบแสดงความคิดเห็น

ส่วน View



วัน เวลาที่แสดงความเห็น

ชื่อผู้แสดงความคิดเห็น

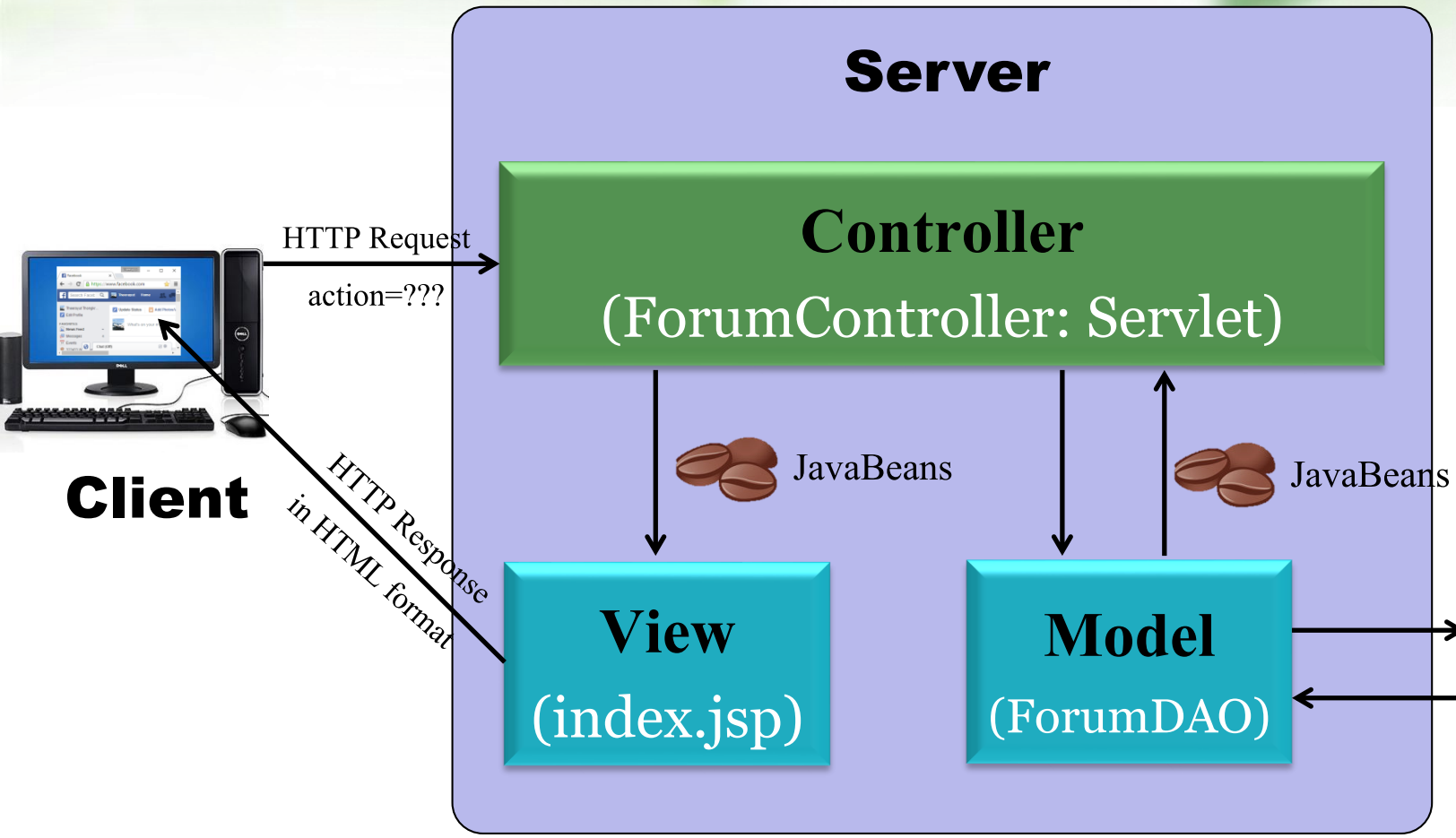
จำนวนผู้ที่กด Love

ผู้ใช้สามารถคลิกที่ลิงก์ Love เพื่อเพิ่มจำนวนความชอบได้

แสดงความคิดเห็นทั้งหมด

ผู้ใช้สามารถเพิ่มความคิดเห็นใหม่ได้ในส่วนท้าย

สถาปัตยกรรมของระบบ



Model และ Controller

- **ส่วน Model** ประกอบด้วยคลาสดังนี้
  - **คลาส Forum** – เป็นคลาส JavaBeans ใช้บรรจุข้อมูลของ 1 แถวจากตาราง Forum
  - **คลาส ForumDAO** – บรรจุเมธอดที่ทำงานกับฐานข้อมูลทั้งหมด

    * การเพิ่มข้อมูลในฟิลด์ชนิด DATETIME สามารถใช้ฟังก์ชัน SQL ช่วยได้ เช่น

    INSERT INTO forum (detail, author, love, post_date) VALUES (?, ?, ?, **now( )**)

    now( ) หมายถึง ให้ดึงวันและเวลาปัจจุบันออกมา

- **ส่วน Controller** ใช้ชื่อคลาส ForumController จะส่งข้อมูลไปทำงานที่ส่วน Model ดังนี้
  - ขอข้อมูลทุกความคิด แล้วส่งผลลัพธ์ไปแสดงผลที่ index.jsp
  - รับข้อมูลจากฟอร์มแสดงความคิดเห็น เพื่อส่งไปเพิ่มข้อมูลที่ Model
  - รับคำร้องขอเพิ่ม Love โดยส่งไปปรับปรุงจำนวน Love ใน Model

โครงสร้างตาราง Forum

ชื่อฟิลด์	คำอธิบาย	ชนิดข้อมูล	หมายเหตุ
fid	รหัสความคิดเห็น	INT	คีย์หลัก แบบเพิ่มค่าอัตโนมัติ
detail	ความคิดเห็น	VARCHAR(100)	
author	ผู้เขียน	VARCHAR(45)	
love	จำนวนผู้ชื่นชอบ	INT	
post_date	วันที่แสดงความคิดเห็น	DATETIME	ใน JavaBean ใช้ชนิด java.sql.Timestamp ในการเก็บข้อมูล