

MVC

Model-View-Controller

SC363204

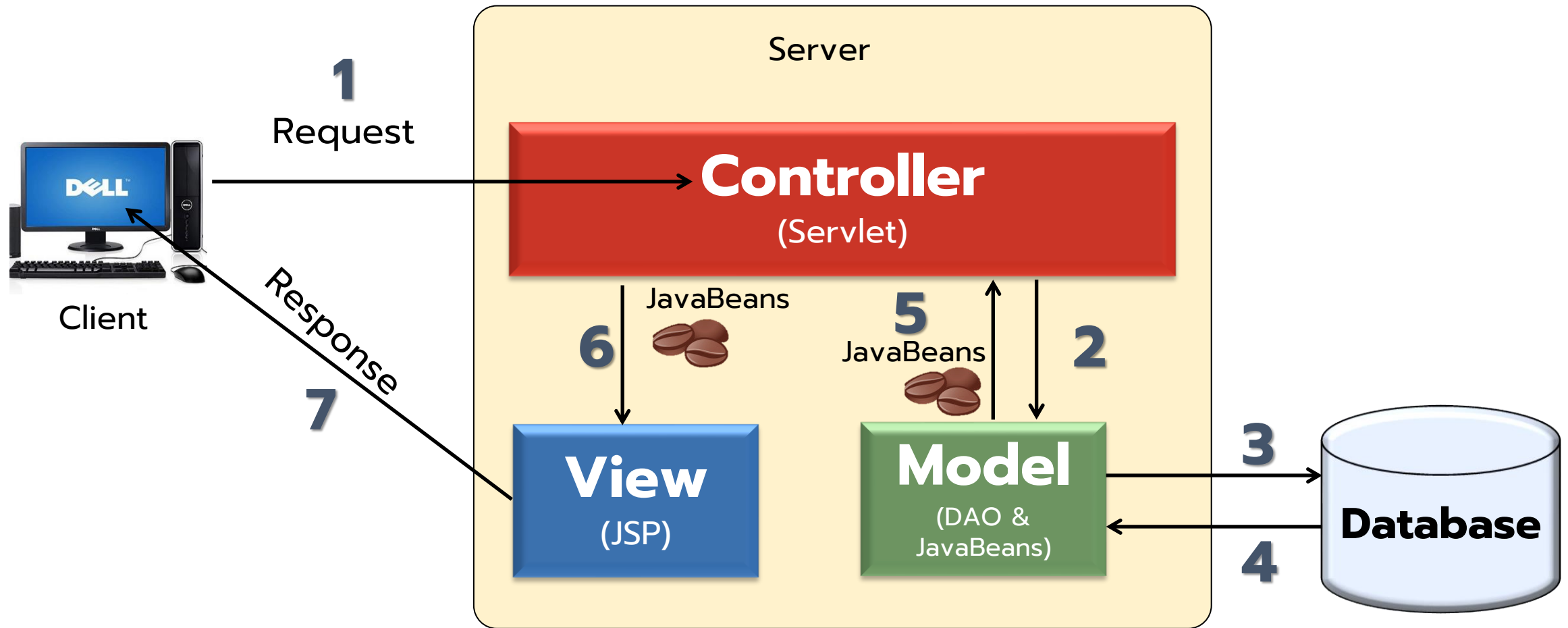
Java Web Application Development
การพัฒนาโปรแกรมประยุกต์บนเว็บด้วยภาษาจาวา



สถาปัตยกรรม MVC

- **MVC** คือ รูปแบบการสร้างแอปพลิเคชันที่แบ่งส่วนการทำงานออกเป็น 3 ส่วน
 - **Controller** ตัวกลางในการรับ-ส่งข้อมูลระหว่างส่วน View และส่วน Model
 - **Model** ประมวลผล Business Logic เช่น การประมวลผลกับฐานข้อมูล ปรับปรุงแก้ไข เพิ่ม หรือดึงข้อมูล ผลลัพธ์จาก Model มักจะเป็นข้อมูลที่จะส่งไปให้ส่วน View แสดงผล
 - **View** คือ ชุดคำสั่งสำหรับ Render หน้า Interface กับผู้ใช้ โดยใช้ข้อมูลจาก Model
- เป้าหมายของ MVC
 - การแบ่งส่วนการทำงานของระบบอย่างชัดเจน ช่วยต่อการทำงานเป็นทีม
 - ช่วยต่อการ maintenance ในอนาคต

สถาปัตยกรรม MVC



MVC

- การส่งต่อ response ไปยัง URL อื่นแบบอัตโนมัติ
 - **แบบ Redirect** คือ การส่ง URL ที่ต้องการ กลับไปยัง browser เพื่อให้ browser ส่ง request กลับมายัง URL ที่กำหนด
 - **แบบ Request Dispatch** คือ การส่ง Object HttpServletRequest และ HttpServletResponse ไปยังอีก URL หนึ่งซึ่งอยู่บน Web Container เดียวกันให้ประมวลผลต่อ
 - วิธีนี้ใช้ส่งภายใน server เดียวกัน ทำให้ผู้ใช้ยังคงเห็น URL เดิมบน Browser

การส่งต่อแบบ Redirect

```
protected void doGet(HttpServletRequest request,  
                        HttpServletResponse response) {  
  
    /**  
     * ประมวลผลตาม request ที่ส่งเข้ามา  
     **/  
  
    // เมื่อทำงานเสร็จให้ redirect ไปยังหน้า slide.html อัตโนมัติ  
    response.sendRedirect("foo/slide.html");  
  
}
```

ขั้นตอนการส่งต่อแบบ Redirect

① client พิมพ์ URL หรือ submit form มายัง Servlet



② browser ส่ง request ไปยัง web server

request
...



③ servlet ทำงานเสร็จ และตัดสินใจว่าจะไปที่ URL ใดต่อ

CodeReturn

⑥ เมื่อ browser รู้ว่าเป็นสถานะ 301 จะอ่าน URL ในส่วน Location



⑤

ส่งสถานะ 301 กลับ และส่ง URL ที่ระบุในเมธอด sendRedirect() แบบไปกับ header response

HTTP/1.1 301 OK
Location: http://www.example.com/
Date: 2018, 19
00:00:00 GMT
Server: Apache/2.4.18
Content-Type: text/html; charset=UTF-8



④

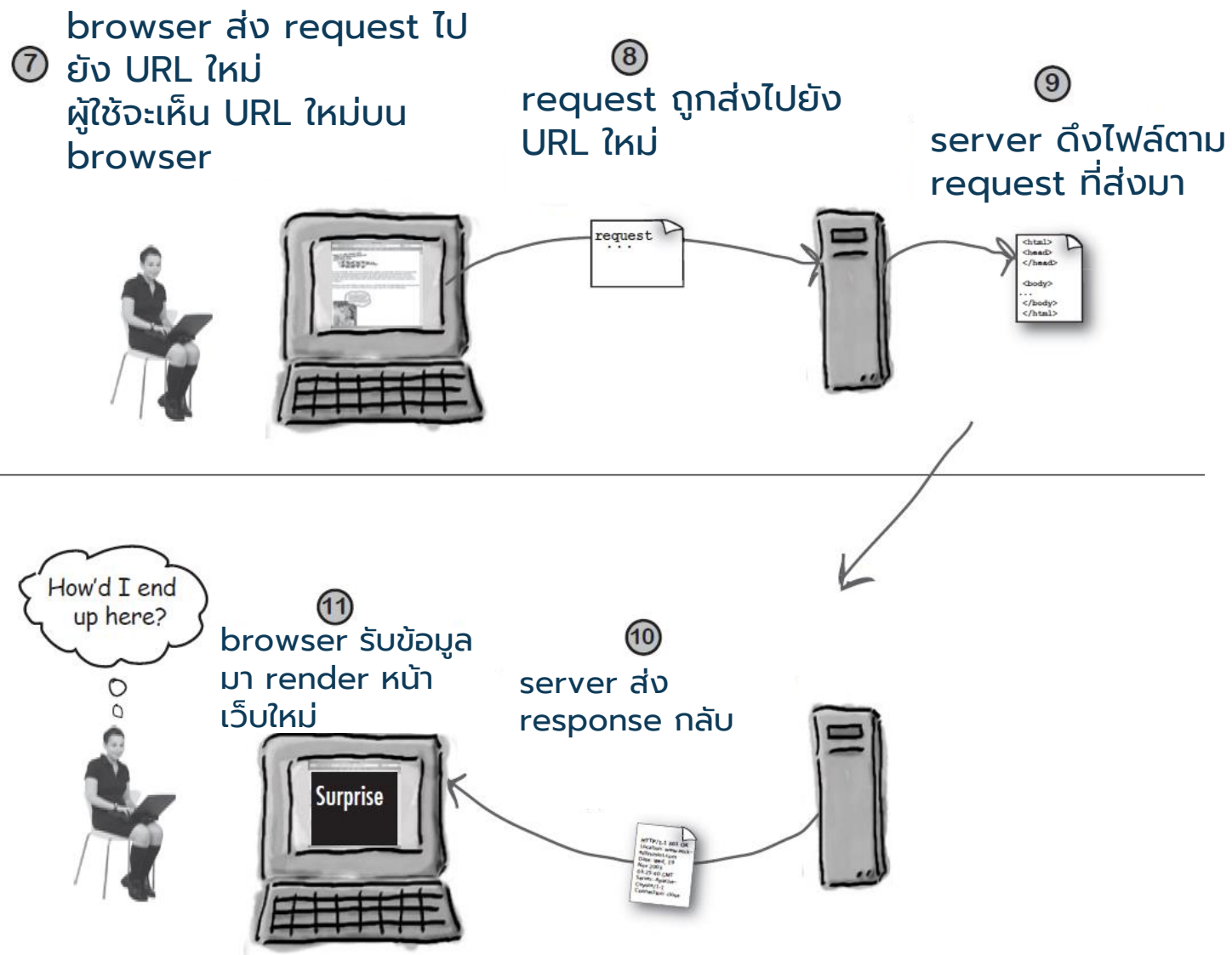
servlet เรียกเมธอด sendRedirect()

CodeReturn

response



ขั้นตอนการส่งต่อแบบ Redirect



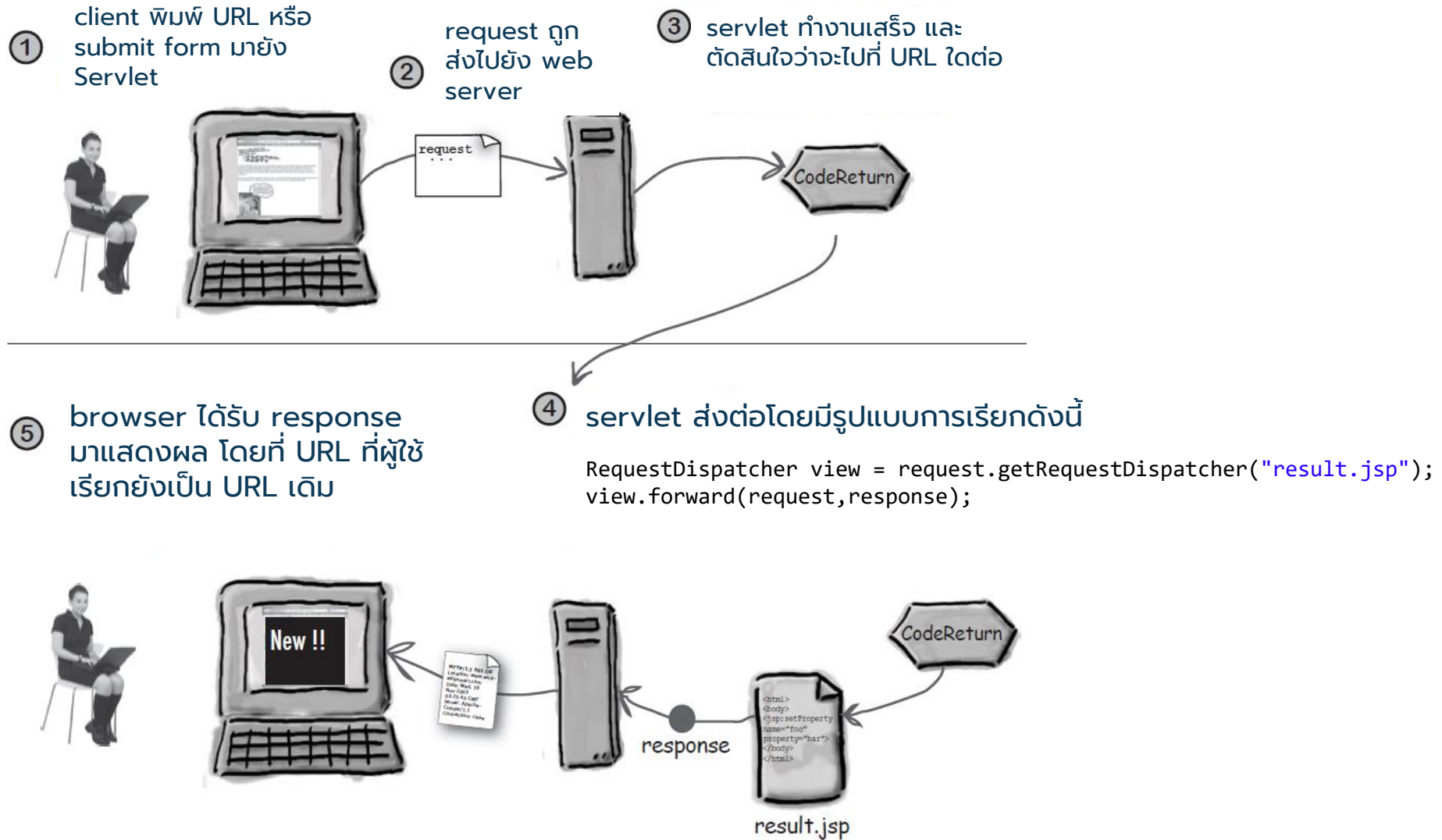
```
protected void doGet(HttpServletRequest request,
                      HttpServletResponse response) {

    /**
     * ประมวลผลตาม request ที่ส่งเข้ามา
     */

    // เมื่อทำงานเสร็จให้ไปยัง servlet หรือ JSP ตาม path ที่ระบุอัตโนมัติ
    RequestDispatcher view = request.getRequestDispatcher("/nextServlet");
    view.forward(request, response);

}
```


ขั้นตอนการส่งต่อแบบ Request Dispatch



การฝากข้อมูลไปกับ request object

Servlet/JSP สามารถส่งต่อ object ไปยังไฟล์ Servlet/JSP อื่นๆ ได้ ขั้นตอนดังนี้

- กำหนดชื่อ และข้อมูลที่ต้องการส่ง

```
request.setAttribute("ชื่อ attribute", ตัวแปรหรือค่าส่งต่อ); เช่น  
request.setAttribute("fullname", "somsak");
```

- ใช้คำสั่งส่งต่อ

```
request.getRequestDispatcher("ชื่อ Servlet/JSP").forward(request,response);
```

- ดึงข้อมูลที่ส่งต่อมาใช้

```
request.getAttribute("ชื่อ attribute"); เช่น  
  
String fullname = (String)request.getAttribute("fullname");
```

Example

- Servlet

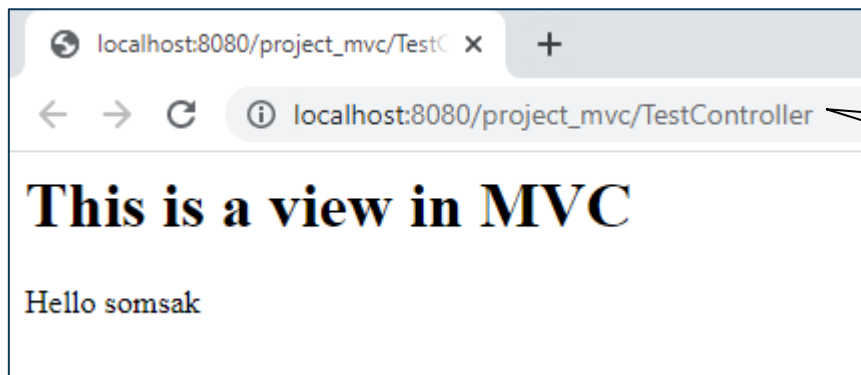
```
//@WebServlet("/TestController")
public class TestController extends HttpServlet {
    protected void doGet(...) {
        request.setAttribute("fullname", "somsak");
        request.getRequestDispatcher("next-page.jsp").forward(request,response);
    }
}
```

- JSP (next-page.jsp)

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<html>
<head><meta charset="utf-8"></head>
<body>
<% String fullname = (String)request.getAttribute("fullname"); %>
<%= fullname %>
</body></html>
```

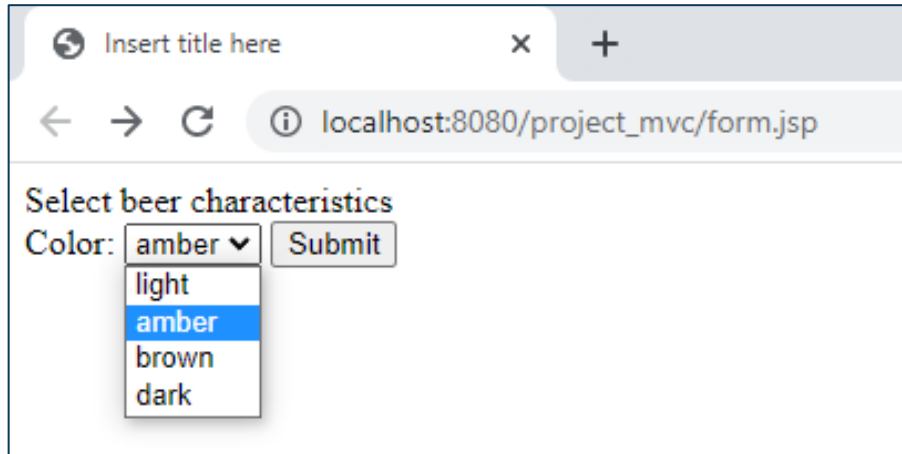
Example

- ผลลัพธ์



สังเกตว่า URL จะไม่
เปลี่ยนไปยัง next-page.jsp

ระบบแนะนำเครื่องดื่ม



Insert title here x +

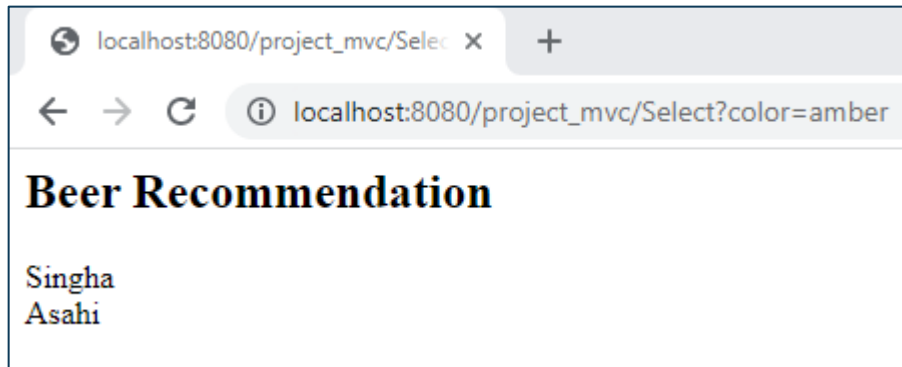
localhost:8080/project_mvc/form.jsp

Select beer characteristics

Color: amber ▼ Submit

- light
- amber
- brown
- dark

ฟอร์ม HTML ที่จะส่งไปยัง Controller



localhost:8080/project_mvc/Select x +

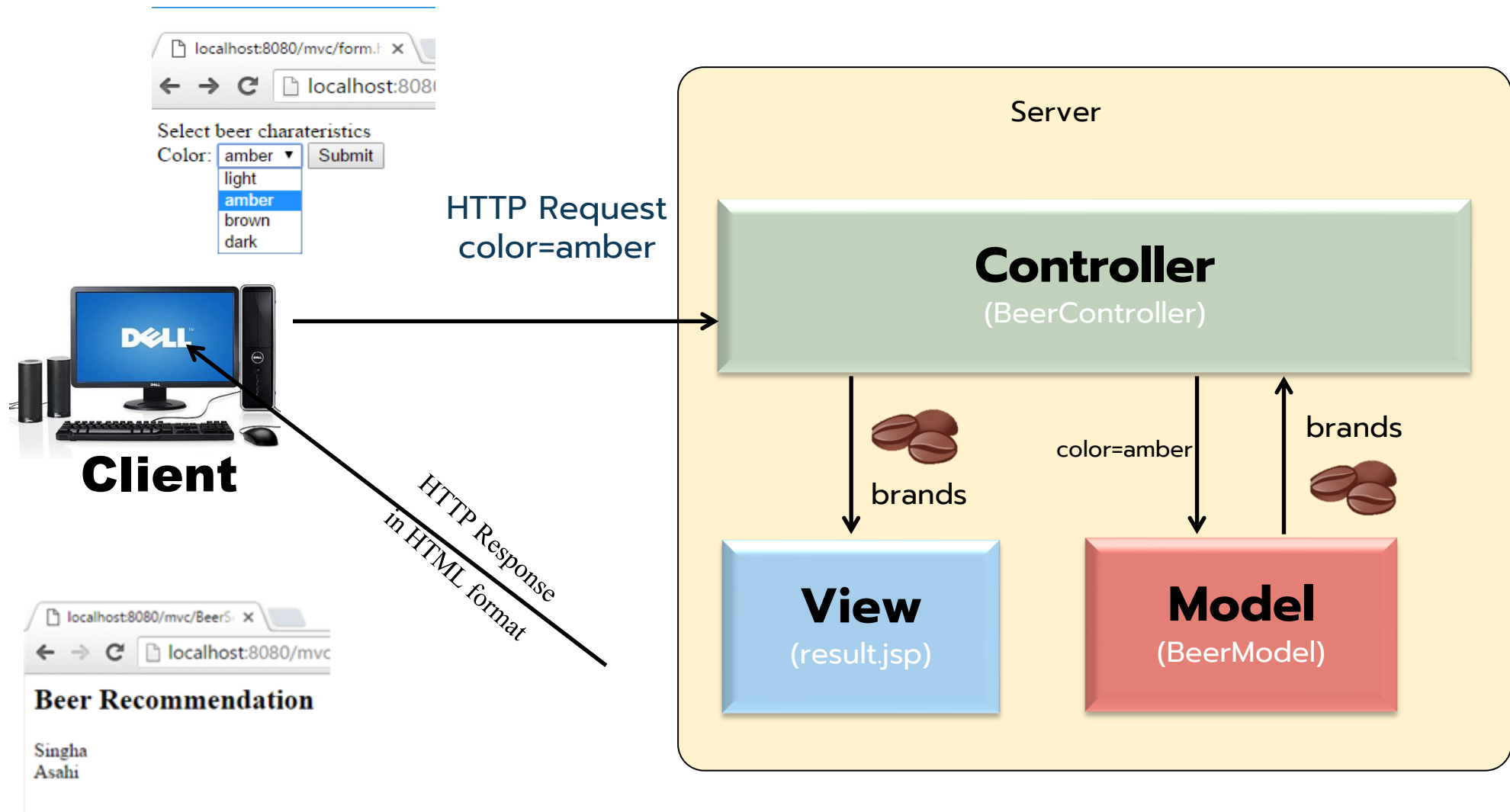
localhost:8080/project_mvc/Select?color=amber

Beer Recommendation

Singha
Asahi

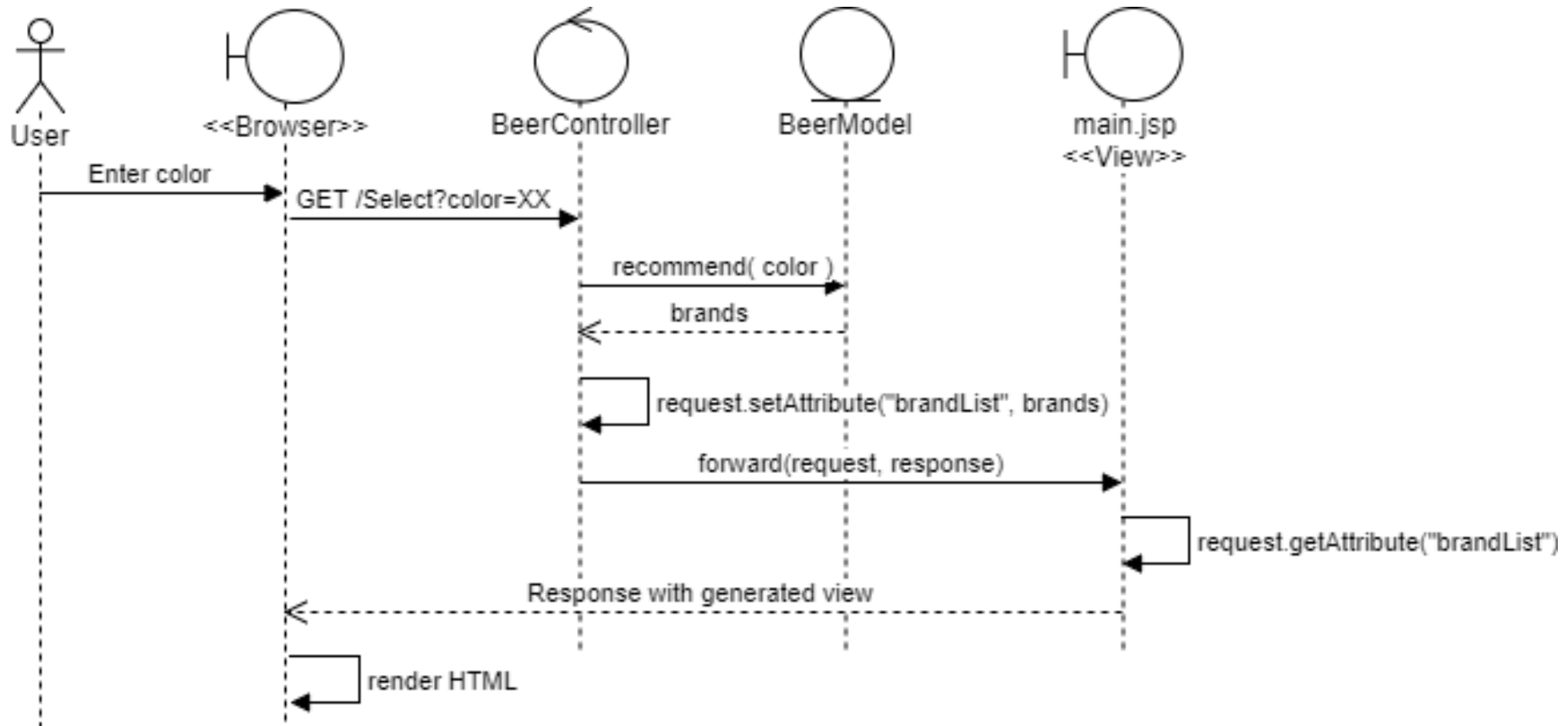
ผลลัพธ์ที่สร้างจาก View

ระบบแนะนำเครื่องดื่ม

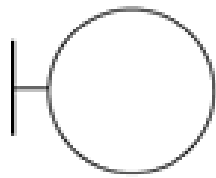


ระบบแนะนำเครื่องดื่ม

- Sequence Diagram ระบบแนะนำเครื่องดื่ม

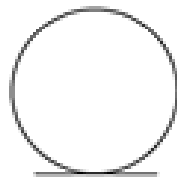


MVC กับสัญลักษณ์ใน UML



Boundary
object

(view)



Entity
object

(model)



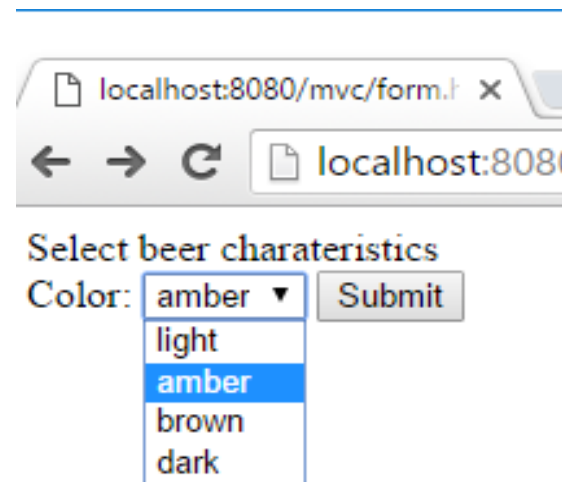
Control
object

(controller)

Form สำหรับส่งข้อมูล

ชื่อ Servlet ที่
รับ request

```
<html>
<body>
  <form action="Select">
    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>
```



ส่วน Controller (Servlet)

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

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

        // ส่วน Model
        BeerModel beerModel = new BeerModel();
        ArrayList<String> brands = beerModel.recommend(color);

        request.setAttribute("brandList", brands);

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

ดึงข้อมูลจาก request object

สร้าง BeerExpert object และส่งที่ผู้ใช้เลือก 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 BeerModel {

    public ArrayList<String> recommend(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

ตรวจสอบเงื่อนไขเพื่อ
เก็บข้อมูลลงใน
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>
```

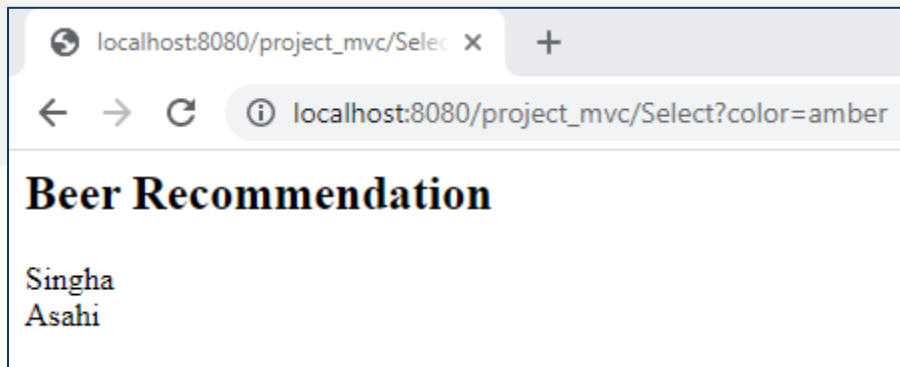
```
<%
    ArrayList<?> brandList = (ArrayList<?>)request.getAttribute("brandList");
```

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

```
</body>
</html>
```

ดึงผลลัพธ์จาก request object

วนลูปนำ String จาก ArrayList มาสร้างเป็นผลลัพธ์ HTML



ภาพรวมโค้ด

form.html

```
1 <html>
2 <body>
3   <form action="/Select">
4     Select beer characteristics<br>
5     Color:
6     <select name="color">
7       <option value="light">light</option>
8       <option value="amber">amber</option>
9       <option value="brown">brown</option>
10      <option value="dark">dark</option>
11    </select>
12    <input type="submit">
13  </form>
14 </body>
15 </html>
16
```

BeerController.java

```
11 @WebServlet("/Select")
12 public class BeerController extends HttpServlet {
13   protected void doGet(HttpServletRequest request, HttpServletResponse response)
14     String color = request.getParameter("color");
15
16     // ส่วน Model
17     BeerModel beerModel = new BeerModel();
18     ArrayList<String> brands = beerModel.recommend(color);
19
20     request.setAttribute("brandList", brands);
21
22     // ส่วนส่ง View
23     request.getRequestDispatcher("result.jsp").forward(request, response);
24   }
25 }
26
```

BeerModel.java

result.jsp

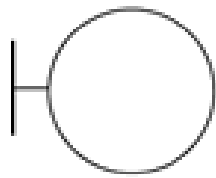
http://localhost:8080/beer/Select?color=amber

Beer Recommendation

Singha
Asahi

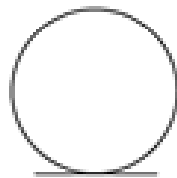
Output

MVC กับสัญลักษณ์ใน UML



Boundary
object

(view)



Entity
object

(model)

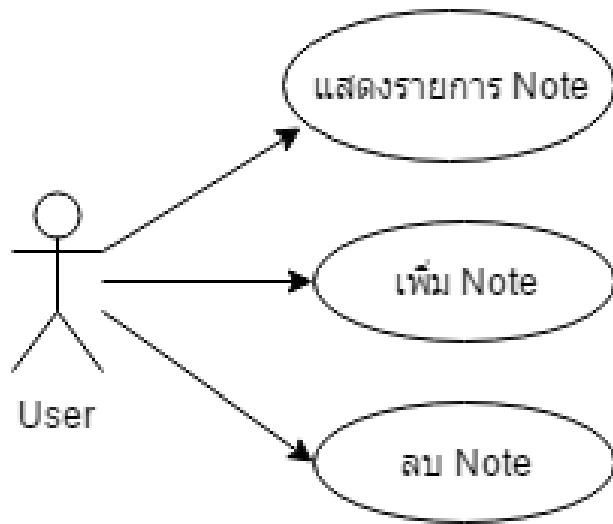


Control
object

(controller)

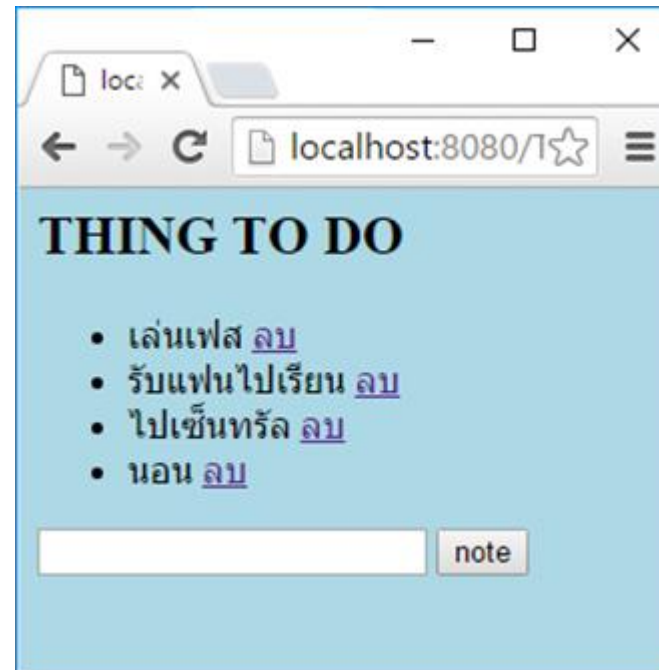
Note App

- Note App คือ เว็บแอปพลิเคชันสำหรับจัดการข้อมูลงานที่ต้องทำในแต่ละวัน การทำงานหลักแสดงดัง usecase diagram



Usecase Diagram

ตัวอย่างหน้าจอ



- สร้าง DAO ที่มีเมธอดจัดการฐานข้อมูล
- สร้าง Controller กำหนดรูปแบบการรับข้อมูล เพื่อส่งต่อไปยัง Model หรือ DAO
- ออกแบบส่วนแสดงผล (View)

การพัฒนาส่วน Model

- สร้างฐานข้อมูลและตาราง note ตามโครงสร้างที่กำหนด

ชื่อฟิลด์	คำอธิบาย	ชนิดข้อมูล	หมายเหตุ
note_id	รหัสโน้ตงาน	INT	คีย์หลัก แบบเพิ่มค่าอัตโนมัติ
note_detail	ชื่องานที่ต้องทำ	VARCHAR(100)	

- สร้างคลาส **NoteDAO** ที่มีเมธอดจัดการฐานข้อมูล
 - อาจเริ่มต้นที่เมธอด SELECT ข้อมูลก่อน
 - ทดสอบการทำงานของเมธอด อาจเขียนฟังก์ชัน main() เพื่อทดสอบบน Console ก่อน
- สร้างคลาส **JavaBeans** เพื่อใช้บรรจุข้อมูล กำหนดชนิดและชื่อ attribute ให้ตรงกับโครงสร้างของตาราง

ตัวอย่างคลาส JavaBeans

```
public class Note {  
    private int noteId;  
    private String noteDetail;  
  
    public int getNoteId() {  
        return noteId;  
    }  
  
    public void setNoteId(int noteId) {  
        this.noteId = noteId;  
    }  
  
    public String getNoteDetail() {  
        return noteDetail;  
    }  
  
    public void setNoteDetail(String noteDetail) {  
        this.noteDetail = noteDetail;  
    }  
}
```

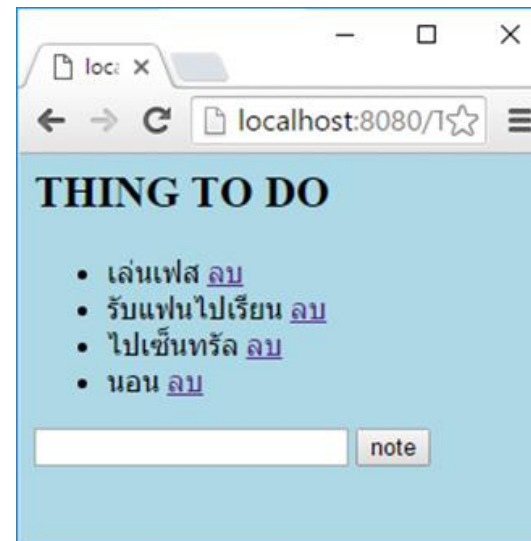
Note
- noteId: int - noteDetail: String
+ getNoteId(): int + setNoteId(int): void + getNoteDetail(): String + setNoteDetail(String): void

การพัฒนาส่วน Controller

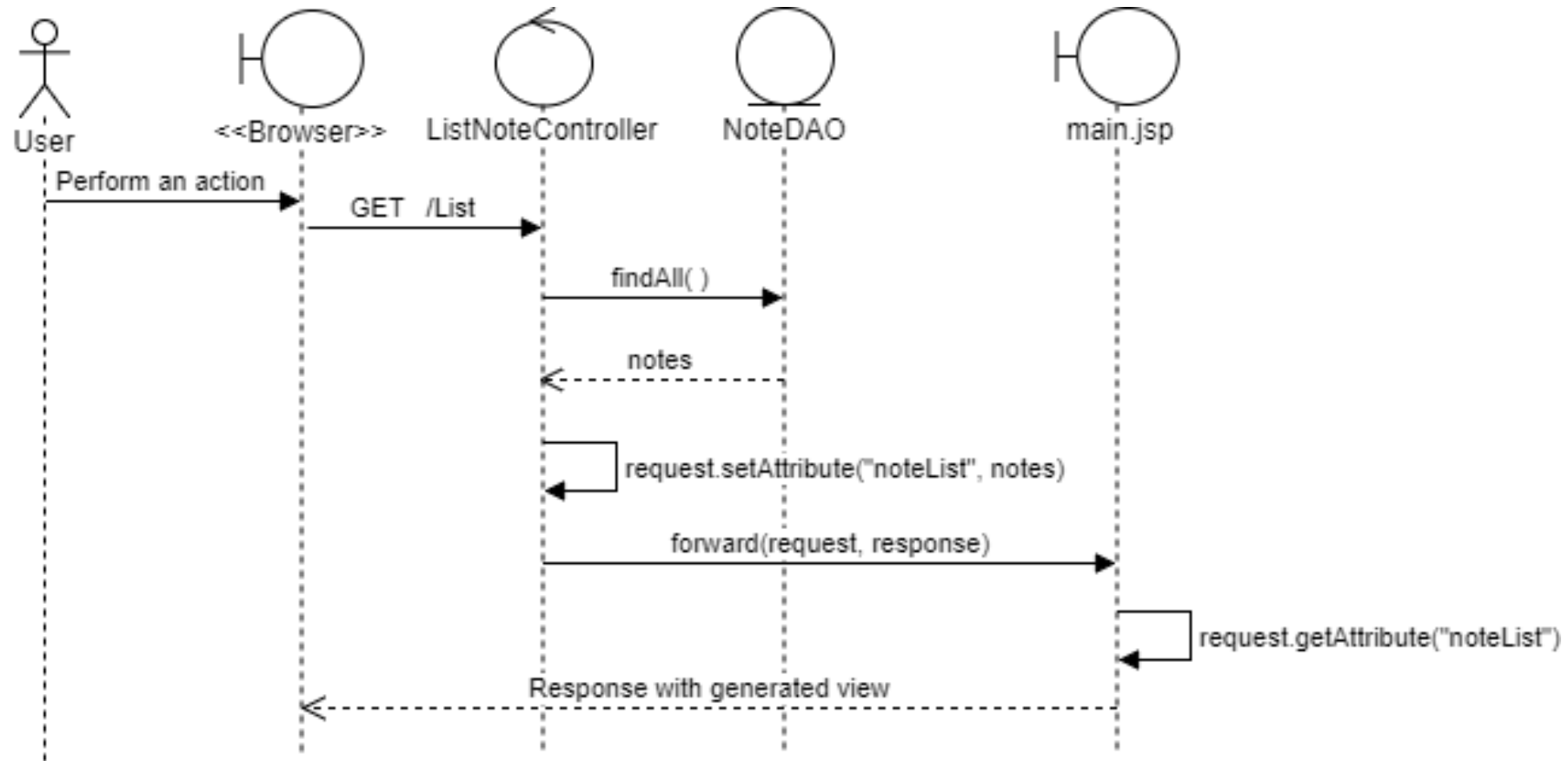
- สร้าง Servlet ใหม่ที่รองรับการทำงานในแต่ละ usecase
 - ListNoteController - ใช้ควบคุมการดึงรายการ note
 - AddNoteController - ใช้รับข้อมูล note ที่จะเพิ่ม
 - DeleteNoteController - ใช้รับรหัส note ที่ต้องการลบ
- แต่ละ Controller เมื่อมีการรับข้อมูลแล้วจะส่งต่อไปทำงานในส่วนของ Model ที่ได้สร้างไว้ และเมื่อ Model ให้ข้อมูลใดๆ จะเก็บลง request object เพื่อส่งต่อไปแสดงผลที่ส่วน View

การพัฒนาส่วน View

- สร้างเว็บเพจด้วย JSP โดยจัดรูปแบบด้วย HTML (main.jsp)
 - ข้อมูลที่จะแสดงผลจะดึงจาก request object โดยใช้เมธอด `getAttribute()`
 - แทรก link สำหรับลบ note โดยส่งไปยัง `DeleteNoteController`
- สร้าง `<form>` เพิ่มข้อมูลในส่วนล่าง โดยส่งไปเพิ่มที่ `AddNoteController`



Sequence Diagram ของ Usecase แสดงรายการ note



โค้ดของ Usecase แสดงรายการ note

NoteDAO.java

```
10 public class NoteDAO {
11     public ArrayList<Note> findAll() {
12         ArrayList<Note> notes = new ArrayList<Note>();
13         try {
14             Class.forName("com.mysql.jdbc.Driver");
15             String dbURL = "jdbc:mysql://localhost/mydb?characterEncoding=utf-8";
16             Connection con = DriverManager.getConnection(dbURL, "root", "");
17             PreparedStatement pStatement = con.prepareStatement("SELECT * FROM note");
18             ResultSet resultSet = pStatement.executeQuery();
19             while (resultSet.next()) {
20                 int noteId = resultSet.getInt("note_id");
21                 String noteDetail = resultSet.getString("note_detail");
22                 System.out.println(noteId + "," + noteDetail);
23                 Note note = new Note();
24                 note.setNoteId(noteId);
25                 note.setNoteDetail(noteDetail);
26                 notes.add(note);
27             }
28         } catch (ClassNotFoundException e) {
29             System.err.println("Error loading driver: " + e);
30         } catch (SQLException e) {
31             System.err.println("Error database connection: " + e);
32         }
33         return notes;
34     }
35 }
```

M

Note.java

```
1 package model;
2
3 public class Note {
4     private int noteId;
5     private String noteDetail;
6
7     public int getNoteId() {
8         return noteId;
9     }
10    public void setNoteId(int noteId) {
11        this.noteId = noteId;
12    }
13    public String getNoteDetail() {
14        return noteDetail;
15    }
16    public void setNoteDetail(String noteDetail) {
17        this.noteDetail = noteDetail;
18    }
19 }
20
21 }
```

JavaBean

ListNoteController.java

```
1 package controller;
2
3 import java.io.IOException;
4
5 @WebServlet("/List")
6 public class ListNoteController extends HttpServlet {
7     private static final long serialVersionUID = 1L;
8
9     protected void doGet(HttpServletRequest request, HttpServletResponse response)
10         throws ServletException, IOException {
11         NoteDAO notedao = new NoteDAO();
12         ArrayList<Note> notes = notedao.findAll();
13         request.setAttribute("noteList", notes);
14         request.getRequestDispatcher("main.jsp").forward(request, response);
15     }
16 }
```

C

main.jsp

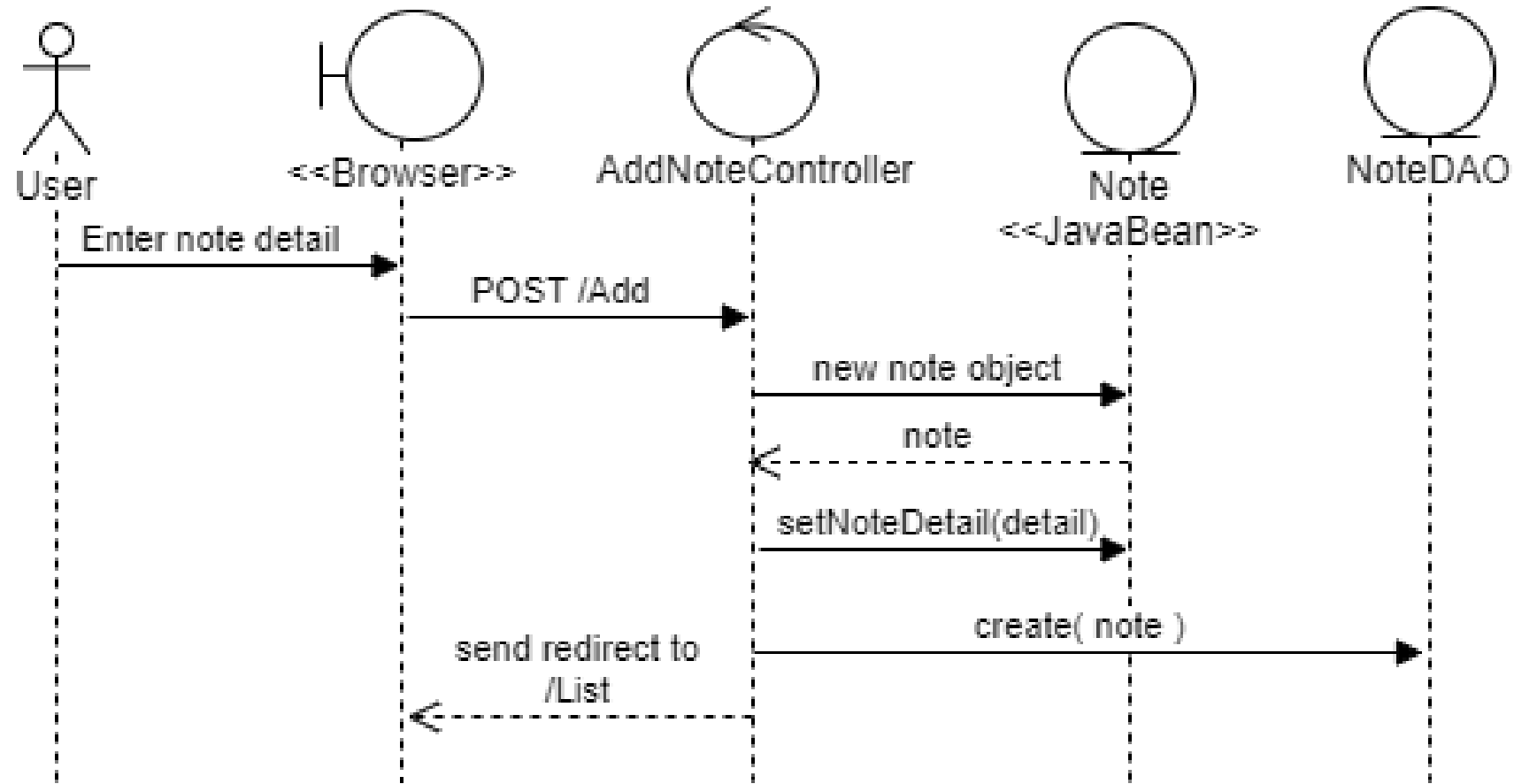
```
5 <body>
6 <%
7     ArrayList<Note> noteList = (ArrayList<Note>)request.getAttribute("noteList");
8     for (int i=0; i<noteList.size(); i++) {
9         Note note = noteList.get(i);
10        out.println(note.getNoteDetail() + "<br>");
11    }
12 %>
13 </body>
```

V

http://localhost:8080/note/List

เล่นเฟส
รับแฟนไปเรียน
ไปเซ็นทรัล
นอน

Sequence Diagram ของ Usecase เพิ่ม Note



Sequence Diagram បង្ហាញ Usecase ៣ Note

