→ World Wide Web Communication

↳ HTTP ย่อมาจาก Hyper Text Transfer Protocol → เวลาส่งต้องมีข้อมูลที่ชัดเจน

↳ เป็นการสื่อสาร ระหว่าง clients และ server โดยอาศัย Internet เป็นตัวกลางในการสื่อสาร โดยมีการส่ง HTTP Request , HTTP Response

Client → Server ↑ Server ตอบกลับ ↑

↳ Client คือ Browser , Server คือ Cloud

↳ การ click หรือ พิมพ์ link คือ request ไปที่ server

↳ HTTP จะ encryt เป็นรหัส code ก่อนจะส่งไปที่ server

→ Request Method

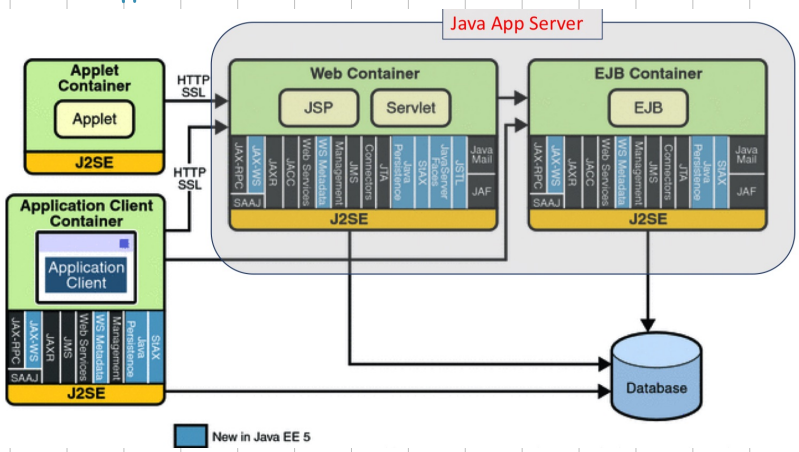
↳ GET ข้อมูลอยู่ใน URL → โชว์ข้อมูลที่เราเห็น ต่างไม่ใหญ่

↳ POST ข้อมูลอยู่ใน body ของตัว Request → ข้อมูลขนาดใหญ่

* Web Application จะ: request เอกสารที่เป็น program

→ JAVA EE Platform API

↳ ใน app ประกอบด้วย JSP หลายตัว / Servlet หลายๆ ตัว



→ ใน Web Container (e.g. Tomcat) ประกอบด้วย JSP , Servlet

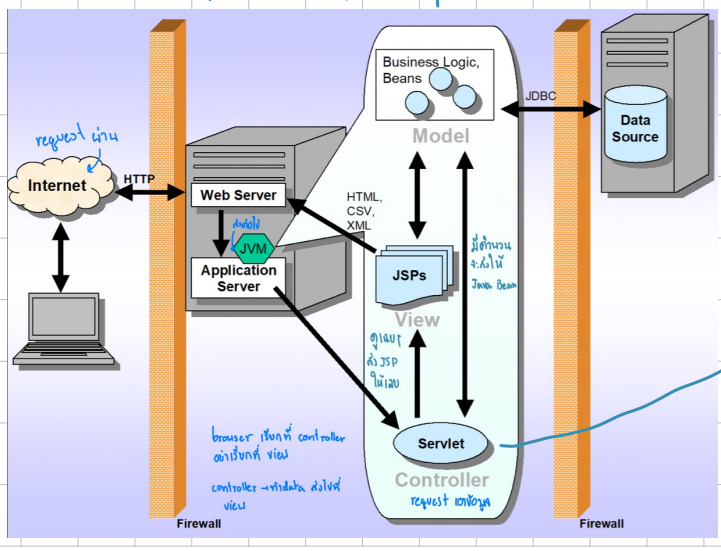
มีหน้าที่คุยกับ Web server

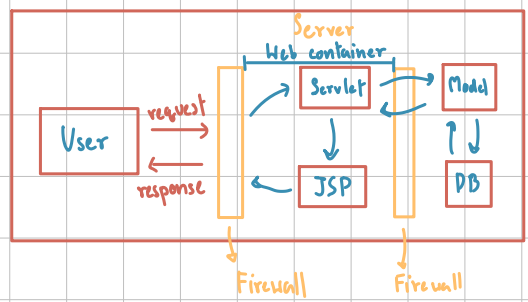
→ MVC (Model View Controller)

↳ Model คือตัวที่จัดการกับ data มี Java Class กำหนด Attribute

↳ View คือส่วนที่แสดงผลให้ client / สื่อสารกับ Client

↳ Controller ควบคุมการทำงาน e.g. กรอกปุ่ม / เป็นตัวที่จัดการกับ browser





interface จาก Jakarta EE → มี JSP / Servlet จัดการกับ request / response ของ HTTP

→ MVC : Benefit

1. Promote code reuse
   → model ไฟสลับ ที่เปลี่ยนคือ View กับ Controller
   → ทำให้ ไม่ต้องเขียน code เพิ่ม หากใช้ logic เดียวกัน
2. Reduces development time
   → ลดเวลาในการเขียน code
3. More maintainable
   → สามารถปรับปรุงได้ง่าย โดยหากต้องการแก้ view , controller , model สามารถแก้ไขได้โดยที่ตัวอื่นไม่ได้รับผลกระทบ

→ Servlet

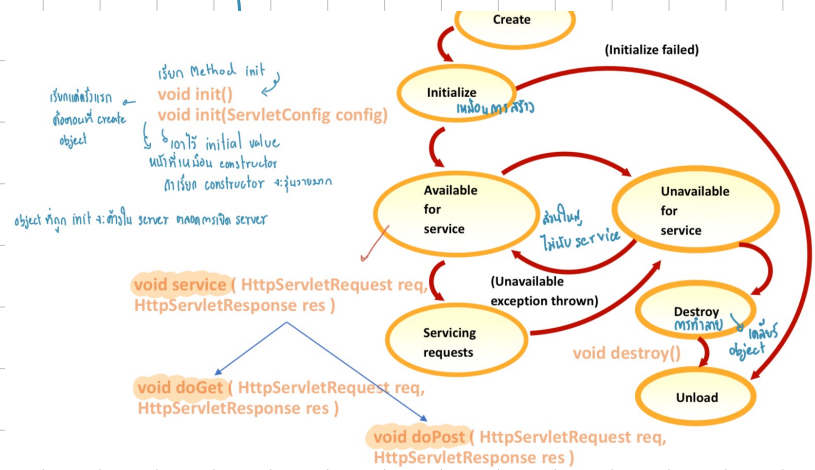
- เป็นส่วนนึงของ Java EE
- รับ HTTP request เข้ามา
- ไม่มี method main
- ทำงานบน JVM ฝั่ง server / ทำงานแบบ multithread
- Parameter ที่ถูกส่งมา จะถูกนำไป deploy บน server
- จัดการภายใต้ Web Container

flow การทำงานของ Servlet

1. Client request เข้ามาในรูปแบบของ URL
2. Web server ส่ง request ไปที่ Web Container
3. Web container เรียก Servlet ที่ Client request มาใช้แบบ Override Method (การเขียนซ้ำ)
   เรียกมาแบบไหน ส่งกลับแบบนั้น (get → get , post → post)
4. Servlet จะสร้าง Response แบบชั่วคราว และส่งไปให้ Server
5. Web Server ส่งกลับให้ Client ในรูปแบบ JSP

→ Java Servlet Life Cycle

- ตัวที่ควบคุม Servlet life คือ Web Container



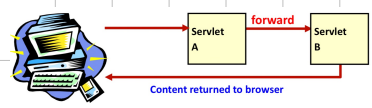
1. Load servlet class → เปิด server
2. Servlet instance is created → สร้าง obj.
3. Init method is invoked → เรียกใช้ method สร้าง
4. Service method is invoked → เรียกใช้ service
5. destroy method is invoked → เรียกใช้ method

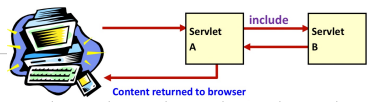
destroy ≠ unload → ปิด server

* จะให้เน้นงานต้อง ลองใช้งานกับการเรียกใช้ DB *

→ Request Dispatcher

- เป็นการส่ง request ไปให้ servlet ตัวอื่น หรือ JSP
- สามารถเรียกใช้ได้จาก 2 interface : ServletContext , HttpServletRequest
  - HttpServletRequest เกี่ยวกับตัวที่ request
  - ServletContext เกี่ยวกับ Servlet Context
- มี 2 แบบ คือ forward และ includes
  - forward ยืมใช้
  - includes ใช้กับการแบ่งงานเป็นส่วนๆ

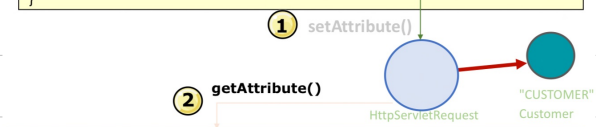




* ถ้า forward ไปแล้ว จะไม่สามารถ handle กับ request , response ได้ แต่ include ทำได้ *

→ Sharing Object

```
// Servlet "A"
public void doGet (HttpServletRequest request, HttpServletResponse resp)... {
  // process request headers & query data
  Customer cust;
  ...
  request.setAttribute("CUSTOMER", cust);
  String viewPage = "/CustomerInfo.jsp";
  getServletContext().getRequestDispatcher(viewPage).forward(request, resp);
}
```



Java สร้าง EL ${} มาเพื่อเขียนบน JSP ส:ดวกขึ้น

```
// CustomerInfo.jsp
<%
  Customer aCust = (Customer) request.getAttribute("CUSTOMER");
%>
```

→ Syntax โบราณ

→ JSP

↳ Jakarta Server Pages Technology

↳ mix กัน ร:หว่าง static HTML + dynamically-generated HTML

↳ JSP file จ:ปร:กอบด้วย JSP Syntax + Markup tags (HTML, XML)

↳ รันบน Web Container → ใช้ JVM ในการทำงาน ทำงานได้เฉพา:ไฟล์ .class

↳ มีหน้าที่ในการส่ง response ที่ Client request มา กลับไปให้ Client

หรือเป็นตัวรับ request จาก client เรียกว่า page compilation นำมาแปลงเป็น servlet และ:แปลงเป็น .class

→ Scope Attribute

↳ Object จ:ผูกกับ User 1 คน สามารถ รันทำงานได้ 4 Scope

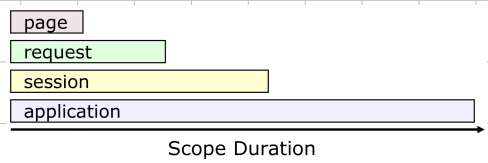
→ Page → หน้า JSP ปัจจุบัน

→ Request → HttpServletRequest Obj ปัจจุบัน
(1 คน/1 req)
↳ สามารถ share ข้อมูลข้าม page ได้

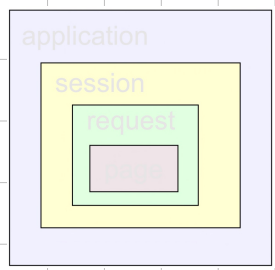
→ Session → Http Session Obj ปัจจุบัน
(เปลี่ยงจนกว่าจะ: ปิด browser)
↳ สามารถ share ข้อมูลข้าม request ได้
↳ เปิด browser = session scope ปิดแล้ว session หาย

→ Application → ServletContext Obj ปัจจุบัน
↳ Share ทุกคน
↳ ทั้ง App มี Obj เดียว

Share ของแต่ละ:คน



" เปลืองเมม ฝั่ง server ถ้าคนเข้ามาเยอะ: "



↳ taglib ไว้ใช้อ้างถึง Library jstl e.g. action foreach

↳ EL ${-}

- <h2>
- Hello, ${user.firstName} ${user.lastName}
- </h2>

สามารถอ้างได้ตามธรรมเนียม

. ใช้ได้กับทุกแบบแต่ถ้าเจอ "-" ต้องใช้ [-]

e.g. ${Cal.age-max} X → ${Cal["age-max"]} ✓

Class user ต้องมี method getFirstName

→ JSTL (JSP Standard Tag Library)

Set a variable in a specific scope to a value (default = page scope) → ตัวอย่าง

```
<c:set var="name" scope="scope" value="expression"/>
```

Display a value, or an alternative if the first value is null

- `<c:out value="expr" default="expr" escapeXml="boolean"/>`
- Example:

```
Hello <c:out value="${user.name}" default="Guest"/>!
```

(ถ้ามีข้อมูลก็แสดง ถ้าไม่มีก็แสดง Guest)

Conditional execution

<c:choose>, <c:when> and <c:otherwise>

(optional)

↳ ใช้ when ต้องมี choose (choose ครอบ when)

```
<c:choose>
  <c:when test="${user.role == 'member'}">
    <p>Welcome, member!</p>
  </c:when>
  <c:otherwise>
    <p>Welcome, guest!</p>
  </c:otherwise>
</c:choose>
```

(:ถ้าว่าง '-', "-")

→ Application State Management

↳ Session Problem ↳ สถานะ: (ขั้นตอนของ user)

- Servlet และ JSP ไม่ควรมี state
- สถานะของ Application ต้องถูกเก็บข้างนอก servlet : Session State

Client Side

- Cookies
- Hidden Fields → User ไม่เห็นบน web แต่ยังมี value อยู่
- URL Rewriting

Server Side

- HTTP Session → นำไปเก็บบน DB ได้
- Content Based Routing → plug-in ของ web application (ใช้กับ ตรงทำ app หลาย server)
- เก็บ สถานะ: ใน database
- JWT : JSON Web Token

→ Cookies

↳ มี ชื่อ และ values เก็บที่ฝั่ง Client

↳ มี Cookie API เอาไว้สร้าง Cookie

↳ Cookie จะถูกเก็บในรูปของ plain text (text file)

↳ ห้ามเก็บข้อมูลที่ต้องการความปลอดภัยไว้ใน cookie

- Creating cookies (เก็บเป็น string)
  - Cookie(String name, String value)
- Sending a cookie back to the browser
  - HttpServletResponse.addCookie(Cookie aCookie)
- Retrieving cookies
  - HttpServletRequest.getCookies()
- Retrieving a cookie's name
  - aCookie.getName()
- Retrieving a cookie's value
  - aCookie.getValue()
- Changing a cookie's value
  - aCookie.setValue(String)

→ สามารถส่งข้าม browser ได้

→ Cookie จะถูกเก็บใน header ของ request เมื่อผ่านไปที่ server

และจะถูกเก็บใน header ของ response ตอนส่งกลับมา

→ มีการใช้ session data tracking

Cookie ทรานซ์เซียนต์ (or session) → Cookie ที่ browser สร้าง

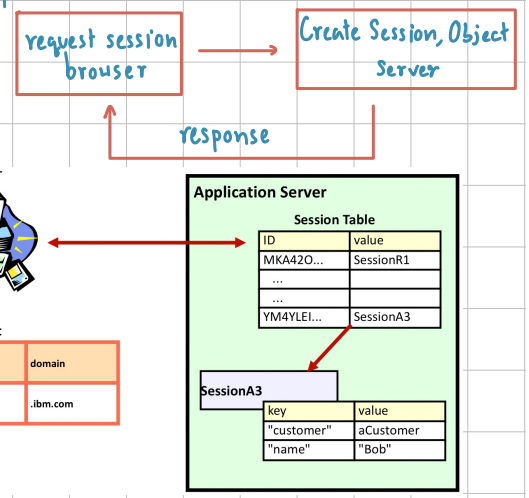
Cookie persistence → Cookie ที่เรากำหนดอายุเอง

อายุ Cookie มีหน่วยเป็นวินาที

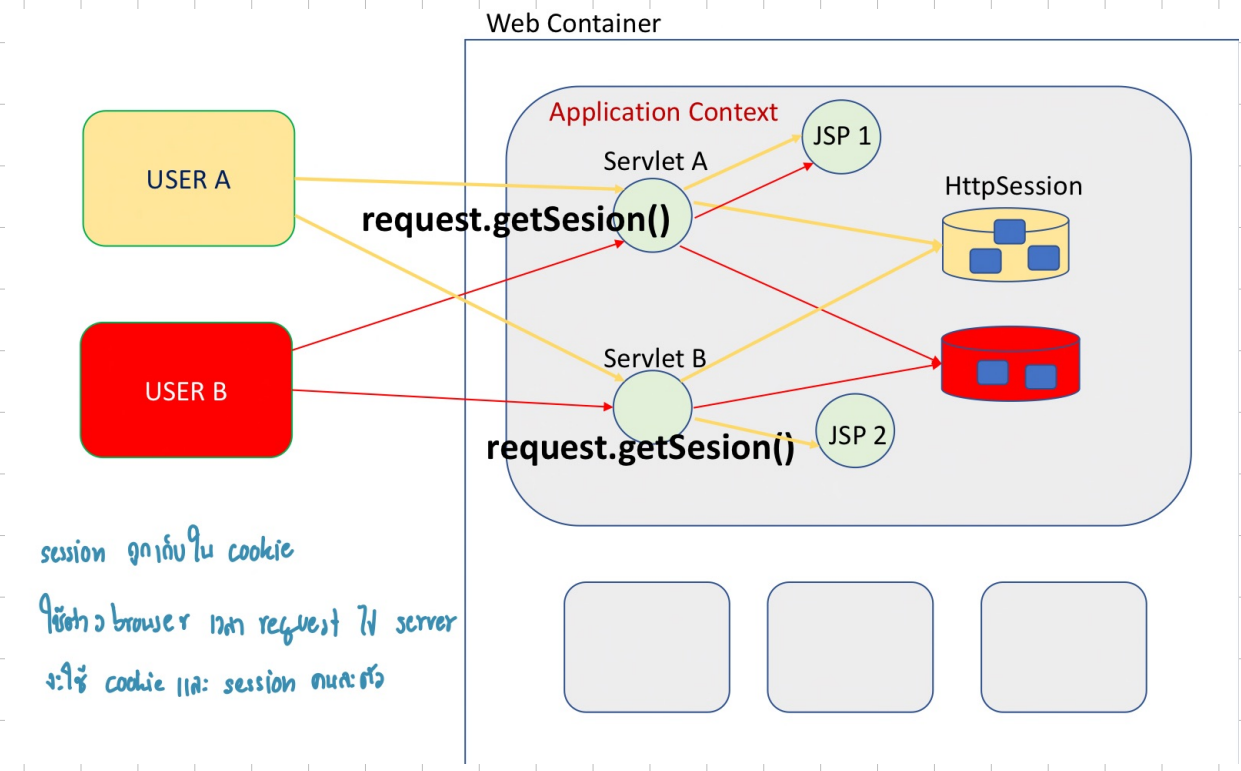
→ HTTP Sesion

↳ HTTPSession Interface เป็น API ของ Servlet ใช้ในการจัดการสถานะของ application บน Server

- Servlet asks to bind to the Session object representing the current session:
  - A session is requested – request.getSession(boolean create)
  - The method returns the current HttpSession, if it exists
  - If create is true (or no parameter is specified) AND no current Session exists, a newly created session is returned
- The session is unavailable when:
  - The client browser is closed
  - The session is explicitly invalidated
  - The session times out
- HttpSessions store application-specific information
  - Stored as <"key", object> pairs
    - void setAttribute(String, Object)
    - Object getAttribute(String)



* ถ้าเกิดการ request บ่อยๆ จะทำให้ memory ของ Server เต็ม



session ถูกเก็บใน cookie

ใช้ข้อมูล browser เวลา request ไป server

จะใช้ cookie แค่: session คนละตัว