

INT202

COURSE DESCRIPTIONS

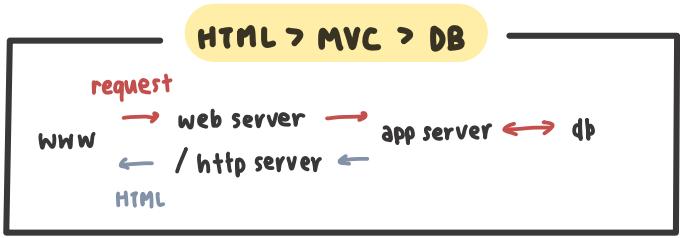
- Web (World Wide Web) architectures
 - HTML (Hypertext Markup Language)
 - HTTP (Hypertext Transfer Protocol)
 - request, responses, request parameters, header

- Web application architectures

- Application servers
- Simple server-side scripting
- MVC (Model view controller) Design Pattern
- Application state management: cookies + session management (ดูใช้แบบบุ๊ฟตอนไหนแล้ว = state)
- ORM (Object-Relational Mapping) 1 object 1 ตารางบันทึก
- Creating, testing, deploying simple web applications
- Standard tools, frameworks, technologies for web

DEPENDENCIES

- เก็บ dependencies ไว้ใน pom.xml
- เวลาหาเจ้าของ local (\Users\Spreeuwii\.m2\)
 - But ก้านไม่เจอก็ไปโหลดจาก central repositories (internet)
<http://repo1.maven.org/maven2/>



JCF (JAVA COLLECTION FRAMEWORK)

- Array (Hash)
 - ผู้ช่วย
 - ลับโดยสร้างบ้าได้
 - เอาข้อมูลมาใส่ในกล่อง
- Linked (Tree)
 - ไม่ต้องพัฒนาต่อ
 - รับบุช่องบ้าได้
 - เอาข้อมูลมาเรียงแบบห้องๆ กัน

ABSTRACT DATA STRUCTURE

- list > ArrayList, LinkedList
- Set > TreeSet (BST), HashSet
- Map > TreeMap, HashMap
- Queue, Stack > linkedlist

hash - obj นั่งกังวล implement hashCode แล้ว เดาต์จาก hash ไปเก็บ (ผู้ใดจะสังเกต ก็จะไปต่อหนัง)
 ปรับตัวอักษรบันทึก load factor ดูจากขนาด table กัน จน. บัณฑุล (1.0 ก็ต้องตั้งปอนด์ = ขนาด table)



HASH!

```
Set<Integer> set = new HashSet<> ( 16, 0.75f );
                                     ↓ loadfactor
- จากตัว.นี่เราระบบต้อง 16 ช่อง But
  สามารถใส่ได้เพียง 12 เพราะว่าตัว loadfactor อยู่ภายใต้ range
  แต่ 75% แห่งนั้น ป้องกันการใส่บ่อมูลมากเกินไป
```

tree - obj ที่มี comparable เท่านั้น
 เพราะว่าการใส่ต้น มาจาก การเรียง
 เพราะวันนี้เราลงต้องเมื่องตัวที่เท่บันได
 (บ่สนใจขนาด !)



TREE!

```
ก้าใช้ TreeSet และ
set.add(new Student(01,"Som",3.56));
↑: error เพราะว่าต้องเป็นตัวที่เท่บันได
* เพราะว่า obj student ยังไม่ implement
  comparable
```

www

ล แล้วอยู่ com Ex email upload download

1. Clients = browsers [ส่งไปด้วย HTTP Requests] ที่ server ต้อง url
2. Server = Web Server [ตอบกลับด้วย HTTP Responses]

HTTP Protocol

- 1) Client ใช้ url เพื่อไป server $\text{http://host:port/path/file} \rightarrow \text{ip address (domain name)}$
- 2) Browser send "request" message เป็น text protocol
- 3) Server แปลงเอกสารนั้นๆ
- 4) Server send "response" message เป็น text protocol
- 5) Browser formats + display

Web Application * จำเป็นต้องมี Application Server

- ต้องมี Application Server เก็บ รับ request + ให้ response (แทน web server)
- ล ต้องบันทึก HTML ไว้ส่งกลับไปใน web server

Application Server

- มี 2 container
- 1) Web container (เริ่มนั้น) ดูแลร่อง JSP เช่น HTML ทำงาน data ของ servlet มากแสดง
 - 2) EJB Container servlet 1 ต่อ / flow (ก่อกราก)
- รับ request + เขียน HTML บุฟ มาก : (

* glass fish เป็น Java app server

Servlet = เป็น standard ! เขียนเหมือนกัน deploy หนึ่งตัว เป็น java class ทำงานบน jakarta EE ทำงานบน HTTP request

- ↓ - จะมีการส่ง parameter 2 ตัว 1) HttpServletRequest request → ใจ text มาต่อกับ obj ใจมาทั้ง header + data
รับจาก url

ทำงานในตัว server

- manage by
web container

ล อาจมี url mapping

```
package com.ibm.example.servlet;
import javax.servlet.http.HttpServlet;
...
public class VerySimpleServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response) throws ServletException, IOException {
        String browser = request.getHeader("User-Agent");
        response.setStatus(HttpServletResponse.SC_OK);
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HTML><HEAD><TITLE>Simple servlet</TITLE></HEAD><BODY>");
        out.println("<TITLE><HEAD><BODY>");
        out.println("Browser details: " + browser);
        out.println("<BODY><HTML>");
    }
}
```

flow - ต้อง servlet จะเป็นส่วนหนึ่งของ url (mapping) ไม่ใช่ชื่อ class

- Web server → App Server → Web container → ไปนา ก้าวต่อไป: ห้าม new

L เรียก method ชื่อ service ก็จะแบบ get → doGet | ไฟล์รันแรก: เขียน

- forward data ให้ jsp เพื่อส่งกลับ ! post → doPost | method in 2 ตัวนี้

MVC

- servlet เป็นตัวรับมา → ส่งข้อมูลให้ jsp มากัดผิดพลาด

ล ไปดึง model มาทำงาน → ต้องจาก db

Benefits

1. ใจ Model นำไปใช้ได้กับหลายที่
2. ลดเวลาการพัฒนา App But ต้องเพิ่มคนมาทำ (model, view, controller)
3. ใจ view, model, controller ได้แบบไม่ว่าผูกขาดกัน

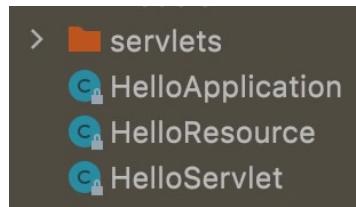
SERVLET + JSP

JSP!

`<h1></. = "Hello world" </.>`
↓
บันทึกว่าข้างในเป็น java

การเรียกไฟล์ servlet นี้ไง

``
↓
การท่องเว็บด้วยชื่อที่ไปต่อหน้า url บน



↑ เวลาเรียกใช้อ่านจาก value

```
@WebServlet(name = "helloServlet", value = "/hello-servlet")
public class HelloServlet extends HttpServlet {
```

in servlet นี้ method หลักๆ 2 อย่าง 1) doget ()
2) dopost ()

```
private String message;
```

```
public void init() { message = "Hello World!"; }
```

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
    response.setContentType("text/html");
    // Hello
    PrintWriter out = response.getWriter();
    out.println("<html><body>");
    out.println("<h1>" + message + "</h1>");
    out.println("</body></html>");
}
```

JAKARTA SERVER PAGE (JSP)

content

- static ผูก logo / Header ไม่พึ่งการเปลี่ยนแปลง
- dynamic ฟังก์ชันทางโปรแกรม

ทำให้เรา mix static+dynamic ได้

- นำไปอ่าน JSP syntax, HTML

= JSP คือ ในการเขียนข้อมูลจากท่อนุญาตให้เราเขียน HTML กับ Scripting เพื่อให้เราเขียนได้ด้วยภาษาที่เราถนัด

JSP SYNTAX

- Directives
- Scripting (Declaration, Expressions, Scriptlets)
- Actions
- ~ Comments

JSP SCRIPTING

1. Scriptlet (พิจารณา lnsn)

`<% valid_code_fragment %>`

ทำให้เราเขียนภาษา Java ห้องในไฟล์

`<% if (Calendar.getInstance().get(Calendar.AM_PM) == Calendar.AM) { %>`

How are u this morning?

`<% } else { %> How are u this afternoon? <% } %>`

VARIABLES (ใช้ถูกที่)

- request: HttpServletRequest Object
- response: HttpServletResponse Object

2. Expressions (สามารถใช้กับมีผลลัพธ์ได้ = ตัวที่มันไม่ผลลัพธ์ออกแล้ว)

`<% = expression %>`

Ex calls incrementCounter method ประจำวัน

`<% = incrementCounter() * 3 + y %>`

```

package sit.int202.simpleweb.servlets;

import ...

@WebServlet(
    name = "StudentListServlet",
    value = {"student-list"}
)
public class StudentListServlet extends HttpServlet {
    public StudentListServlet() {
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        StudentRepository studentRepository = new StudentRepository();
        * request.setAttribute("students", studentRepository.all()); // ป้อน collection ให้ JSP ด้วย obj
        request.getRequestDispatcher("/StudentList.jsp").forward(request, response);
    }
}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
}

```

servlet

```

<%          ต่องบอกด้วยว่าต้องห้ามต่อ collection ที่เป็นตัวในพื้นที่ให้กับ JSP ด้วย
Collection<Student> students = (Collection<Student>) request.getAttribute("students");
for (Student student : students) {%
    <div class="col-2">           นำ java file student
        <div>Id: <%= student.getId()%></div>
        <div>Name: <%= student.getName()%></div>
        <div>gpax: <%= student.getGpax()%></div>
        <div><hr></div>
</div>

```

JSP

HTTP PROTOCOL : request

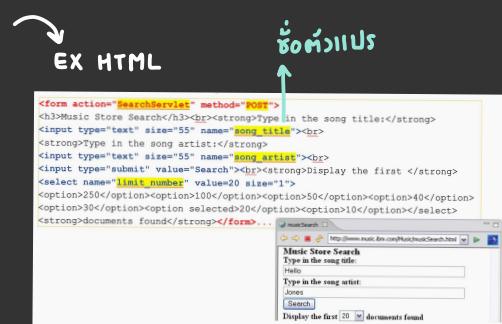
request phase

- Request (POST)
- Header values
(เก็บบรรทัด)
- Posted data

```
POST /Music/SearchServlet HTTP/1.1
Accept: */
Referer: http://www.music.ibm.com/Music/musicSearch.html
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; ...)
Host: localhost:9080
Content-Length: 50
Connection: Keep-Alive
Cache-Control: no-cache

song_title=Hello&song_artist=Jones&limit_number=20
```

↓
ชื่อตัวแปร
↓
↓
request parameter



การจับ Request (เวลาดูมองหน้าจอ)

- getParameter (String name) = ดึงออกมาเป็น String
- getParameterMap () = ส่วนรับ multiple (ในการมีลักษณะ value)
- getParameterName () = ดูเหมือน enum + เล่านามได้ชัด
- getParameterValue (String name) = return value ก็จะง่าย

```
public void processRequest(HttpServletRequest request,
HttpServletResponse response) throws...
String name = request.getParameter("name");
if (name==null) {
    name = "unknown";
} else if (name.length== 0) {
    name = "missing";
}
String height[] = request.getParameterValues("height");
if (height == null) {
    height = new String[] {"unknown"};
}
for (int i = 0; i < height.length; i++) {
    if (height[i].length== 0) height[i] = "missing";
}
```

Ex $x=50 \& y=30 \& name=Somchai \& subject=Jakarta \& subject=database$

`request.getParameter (" name ") = Somchai`

`request.getParameter (" x ") = 50`

`request.getParameter (" Subject ") = Jakarta`

↳ ถ้าต้องการดูตัวแปรเพิ่มเติม ก็ต้องการดูตัวแปรที่ตั้งค่าให้เป็นล่วงๆ

`request.getParameterValue (" subject ") = { Jakarta, database }`

`request.getParameterName () = enum { " x ", " y ", " name ", " subject " }`

HTTP PROTOCOL: response

response phase

- status information
- Header values
(เก็บบรรทัด)
- output document

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=ISO-8859-1
Content-Language: en-US
Set-Cookie: JSESSIONID=000093gEM_2OosI_mR6GBkZLJy9:-1; Path=/
Transfer-Encoding: chunked
Date: Mon, 12 Mar 2007 18:32:27 GMT
Server: WebSphere Application Server/6.1
Expires: Thu, 01 Dec 1994 16:00:00 GMT
Cache-Control: no-cache="set-cookie, set-cookie2"
<HTML>
<BODY>
<H1>Very simple dynamic document created on 01-Jun-2001</H1>
</BODY>
</HTML>
```

status information

1xx information

2xx Success

3xx Redirection (ไปหน้าจอเปลี่ยน)

4xx Client Error (request ไม่ถูกต้อง)

5xx Server Error (code พัง)

Request Dispatcher

- ค้นหา resource ที่อยู่ในโปรเจกต์
 - แบ่งออก 2 ที่ 1) ServletContext
2) HttpServletRequest
- Forward
↓
content returned
to Browser

- use

1) Forward to jsp page

`getServletContext().getRequestDispatcher("/pages/webBalance.jsp").forward(request,response);`

2) include static HTML

`getServletContext().getRequestDispatcher("/pages/navigation-bar.html").include(request,response);`

SHARING OBJECT

- ServletContext

- 1) `getServletContext().setAttribute (" objname ", anObj);`
- 2) `getServletContext().getAttribute (" Objname ");`

- HttpServletRequest

- 1) `request.setAttribute (" Objname ", anObj);`
- 2) `request.getAttribute (" Objname ");`

↳ ก็จะ: get ต้องผ่านการ set มาก่อนต้องจะ: จึง servlet

```

<form action="add-new-student" method="post">
    ID: <input type="number" name="id" required><br>
    Name: <input type="text" name="name" required><br>
    Gpax: <input type="number" name="gpax" max="4.0" required><hr>
    <input type="submit" value="Save">
</form>

```

JSP
FORM

```

package sit.int202.simpleweb.servlets;

import ...

@WebServlet(
    name = "AddNewStudentServlet",
    value = {"/add-new-student"}
)
public class AddNewStudentServlet extends HttpServlet {
    public AddNewStudentServlet() {
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        this.getServletContext().getRequestDispatcher(s: "/StudentForm.jsp").forward(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String id = request.getParameter(s: "id");
        String name = request.getParameter(s: "name");
        String gpax = request.getParameter(s: "gpax");
        Student student = new Student(Integer.valueOf(id), name, Double.valueOf(gpax));
        file java
        StudentRepository studentRepository = new StudentRepository();
        studentRepository.save(student);  ↗ ชื่อส่วนต่อไปนี้เป็นตัวแทน object ก็จะบันทึกใน page นี้
        request.setAttribute(s: "newStudent", student);  ↗ ก็จะไปบันทึก !
        this.getServletContext().getRequestDispatcher(s: "/NewStudentInfo.jsp").forward(request, response);
    }
}

```

JSP EL §§

↗ ถ้า obj java มาอยู่ใน jsp

JSP expression Language (EL) = simple language

- ถ้าต้องไปส์ java มาได้

- ใช้แค่กันก็ได้

- ผู้ relational, logical, arithmetic

- ถ้า obj จาก web container ได้ (implicit)

- ห้าม function ไว้จะเละไปเรียก

- ใช้ร่วมกับ HTML tag , JSTL

↑ meaning = มันเรียกพ่อ method บ้าได้รีบกันผ่าน attribute
แปลว่าต้องมี getter, setter หรือห้องซึ่งสามารถเพื่อมาผ่านไปเรียกได้

* ถ้าเกิดไม่มี

<h2> Hello, \${user.firstname} \${user.lastname} </h2>

* เป็นกัน custom tag (JSTL)

<c:if test = "\${tag3}"> ... </c:if>

IMPLICIT OBJECTS

- | | |
|--------------------|---------------|
| - pageContext | - param |
| - pageScope | - paramValue |
| - requestScope | - headerValue |
| - sessionScope | - cookie |
| - applicationScope | - initParam |

Ex <!--= request.getAttribute("name") -->
~~~~~  
\${requestScope.name}

## SYNTAX OVERVIEW 🐝

- [] and .  
used to Maps, Lists, Arrays of obj
- Arithmetic, logical, Relational comparisons
- access ต้องไปร่องๆ ได้ ผ่านชั้นต่อไป attribute ที่อยู่ใน obj สะดวกนั้น

## BASIC SYNTAX ELEMENTS 🦷

- Literals : Boolean, int, float, string, null
- Operator : Accessor (. [ ] ), Arithmetic (+ - \* / . . ), Relational ( ==, !=, <, >, <=, >= ), Logical ( & &, ||, ! ), Empty (empty ), conditional ( ? : )

Ex \${param.cost \* 1.085}

~~~~~ \${empty sessionScope ? "yes" : "no"}

EX

```
package sit.int202.simpleweb.models;
public class Student {
    private Integer id;
    private String name;
    private Double gpax;

    public Integer getId() { return this.id; }

    public String getName() { return this.name; }

    public Double getGpax() { return this.gpax; }

    public void setId(Integer id) { this.id = id; }

    public void setName(String name) { this.name = name; }

    public void setGpax(Double gpax) { this.gpax = gpax; }

    public Student(Integer id, String name, Double gpax) {
        this.id = id;
        this.name = name;
        this.gpax = gpax;
    }
}
```

MODEL

<h2>New Student has been added</h2>

Student Id: \${newStudent.id}

Student Name: \${newStudent.name}

Gpax: \${newStudent.gpax}

JSP1



* การเรียกใช้ในได้รีบกัน attribute
พันเรียกจาก method getter

* จุดเด่นของการใช้ param และ ตัวแปร

```
<input type="hidden" name="x" value="100">
<input type="hidden" name="x" value="900">
<input type="hidden" name="x" value="400">
```

request.param.id: \${param.id}

request.param.name: \${param.name}
 param = parameter
request.param.gpax: \${param.gpax}

x = { \${paramValues.x[0]}, \${paramValues.x[1]}, \${paramValues.x[2]} }

x = {
 <c:forEach items="\${paramValues.x}" var="v" varStatus="vs">
 \${vs.count}: \${v} |
 </c:forEach> |
}

 User-Agent X
User-Agent = \${header["User-Agent"]}

JSP2



JSTL (library von custom tag)

- ໃນໂນໂລບໍ່ທີ່ສ່ວງເພື່ອເປັນນັບນັກ JSP
- custom tag = ກາຮສ່ວງ tag HTML ຫັນພາກຕ່າງໆ
- └ ໂປດໄຈ້ XML (ໃຫດນີ້ນັກ java ທ່ານີ້ດັນທີ່ view)
- └ ຈະເກີດ JSTL ທີ່ເປັນນາມຕ່າງໆ!
- ຜິ 4 ຊົດ
- * 1) Flow (iteration + conditionals)
- 2) Manipulation of XML document
- * 3) Internationalization tags
- 4) SQL tags

SAMPLE JSTL TAGS

- set

ກະບຸ scope ວ່າໃນຍຸແນ
↑ ມີຕ່າມເບີນວິທີ
`<c: set var="name" scope="scope" value="expression" />`

- out ແລະ ດັວວັດ

`<c: out value="expr" default="expr" escapeXml="boolean" />`
Ex Hello `<c:out value="${user.name}" default="Guest" />`

- conditional (if else)

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

Ex `<c:choose>`

`<c:when test="${user.role == 'member'}">`

`<p> welcome, member! </p>`

`</c:when> // ພັນການ case ອີ່ when ຢ່າງດັນ!`

`<c:otherwise>`

`<p> welcome, guest ! </p>`

`</c:otherwise>`

`</c:choose>`

ກໍ່ ມອງຂອງການສະໝັກ
`<c:forEach items="${students}" var="s" varStatus="vs">`
`<div class="col-2 p-1 m-2 border border-secondary">`
`${vs.count%5==1 || vs.count%5==3 ? 'bg-primary' : ''}>`

ເປັນການແກ່ຈາກໂວກເກ່າ
ນາຟ el + jstl ແກ້ນ!

- forEach ຮູ່ນີ້ collections, Maps, Iterators, Enum, Array

Ex `<table>` ເປັນພາກ collection, array

`<c:forEach items="${customer}" var="cust">`

`<tr> <td> ${cust.name} </td>`

`<td> ${cust.addr} </td> </tr>`

`</c:forEach>`

`</table>`

↑ ໂຄນຈາກ request in StudentListServlet
`<c:forEach items="${students}" var="s"> ກຳນົດໄສຕ່າງໆໃນ loop ເປັນ s`
`<div class="col-2 p-1 m-2 border border-secondary">`
`<div> Student Id: ${s.id} </div>`
 EL
`<div> Name: ${s.name} </div>`
`<div> Gpax: ${s.gpax} </div>`
`</div>`
`</c:forEach>`

`x = {` EL `ແລ້ວໄຫວ່າຕົວ (v: value)`

`<c:forEach items="${paramValues.x}" var="v" varStatus="vs">`

`${vs.count}: ${v} > ຖອນດີດູບ!`

`</c:forEach>`

`}
`

↓ ເກັບບົດຂອງການກວດນຸບ
ຝ 1) index ສອນ 0, 1, 2, 3
2) COUNT ລົດວາ!

ເຖິງ ຄົນ ຕໍ່
`<c:forEach begin="1" end="100" var="value">`
 `<div class="box"> ${value} </div>`
`</c:forEach>` ຮູ່ 1 - 100

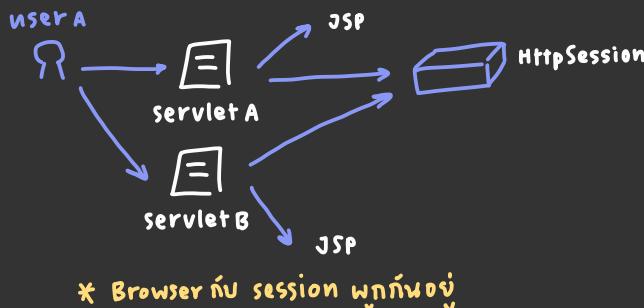
ງານນາ!

HTTP SESSION (obj ທີ່ເກີບ session)

- ກໍາໄລຈະເກີບຂອງລາຍ request → forward → response ແລ້ວຂອງລາຍ request ຈະໄສສາມາດໃຫ້ວາໄດ້ປັດ (ຕ້ອງບໍ່ດີ)
- Obj ທີ່ເຮັດວຽກກີບໃຫ້ user ໂຕລະຄົມແລ້ວຕົວໆການໃນເຖິງ request ກົນໄດ້ຕົວໆໃຊ້ Session ຂອງລາຍ: 0ບຸ້ຈະກ່າວ
- Method `request.getSession(boolean create)`
 - ↳ ກໍາເກີດຢັງໄວ້ເຊັ່ນພື້ນນະ: ສ້າງ session ໃນພື້ນນະ
 - But ກໍາພື້ນ session ອຸ່ນ ແລ້ວກົງ: ໄກສົງເຄີນ
- ການໄຫວຂອງລາຍຈາກເກີບ `void setAttribute(String key, Object value)`
 - ↳ servlet → object
 - But ກໍາເມີນ JSP ກໍາລື່ງ retrieve

```
request.getSession()
(true) = ກໍາມີຂອງລາຍເຕັມນາໄຟ
↓
ມີມານີ້ແລ້ວນີ້ຈະສ້າງ session ຖ້າແນວ
(FALSE) = ກໍາມີຂອງລາຍເຕັມນາໃຫ້
↓
ມີມານີ້ຈະເປີນ null ໄຟສ້າງໃນນີ້!
* () ຕັ້ງຈະ parameter = (true)
```

timeout
ເບີນໂດດທ່ານໃຫຍ່ໄປກ່າງ
ປັດ browser



- * ກໍານີ້ user B ມາເຮັດ servlet A, B ຈະໄປເປັນໄຟນ໌
HttpSession ວິວຕົວໄອງ (ໄວຣະວັນບັ່ງລາຍໄຊເຫັນບັ່ງກັນເພື່ອຫຼັງ)
- * ກໍາພົດພເວົ້າໄຟ 10,000 ອຸ່ນ ຜົກ໌ session
- ບັນຍຸກົບຈຸນ, user ທີ່ຜູ້ໃຊ້ການໃຊ້ມານີ້ນີ້

```
@WebServlet(
    name = "CourseRegisterHistoryServlet",
    value = "/CourseRegisterHistory")
public class CourseRegisterHistoryServlet extends HttpServlet {
    public CourseRegisterHistoryServlet() {
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        HttpSession session = request.getSession( b: false); // ກອນສ່າງເບີນການໃຫ້ False ແບບເຊົາດນວກເບີນນາເກີບຂອງລາຍແລ້ວກົງ: ດັວງຕົວທີ່ຕົວໆ session ມີປັດ
        if (session == null || session.getAttribute( s: "courseRegistered") == null) { // But ກໍານີ້ໃນໄວ້ຈຳເປົ້າໃຫ້ session ໃນພັກ: ຢູ່ຈຳວັນ
            request.setAttribute( s: "message", o: "ໄມ້ມີຂອງມູດ ກາລົງທະເບີນ ຂອງທີ່ລັງທະເບີນກອນ");
            // ຢູ່ໃຫ້ showເປັນນັອງ jsp
            this.getServletContext().getRequestDispatcher( s: "/ShowRegisterHistory.jsp").forward(request, response);
        }
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    }
}
```

ລົກປະການ: ນອງ EL ໂຈາຕ້ອງຕົວແປງຈະດີ

- 1) PageScope
- 2) RequestScope
- 3) SessionScope
- 4) ApplicationScope

ຕໍ່ໄຟ້ນີ້ກັບ 4 ຕົວກົງກ່ອງທີ່ຈະ=1

- ຊື່ຕົວກົງແປງເສັບກົນກໍາໃຫ້ຕ້ອງ: ບຸ້ scope ເຮັດ!

ສະນຸ້ມ ເຊັ່ນຕົວໄປຮັດ X ໃນກົງ 4 scope

X ທີ່ໄວ້ໄດ້ຈະ ພາຈາກ PageScope!

ກໍາຮະບຸ scope ກົງ: ຮູ່ວ່າໄຮຈະ: ເຕົາ X ນອງ scope 1 ນັ້ນ

ໃປທ່າ method doPost

✓ voic ↑

ການໃຊ້ EL+JSTL in JSP

```

<select name="semester" id="semester" class="px-4">
    <c:forEach items="#${semesters}" var="semester" varStatus="vs">           ↗ ຈຳລັງຈະ 1) index
        <c:if test="#{semester != null}"> ຕະຫີມເວົ້າດ້ານ semester ເມນ null ສັນ
            <option value="#{vs.index}" ${vs.index==selectedSemester ? 'selected':''}>#${semester}</option>
        </c:if>           ↘ index
    </c:forEach>
</select>

```

1ພລ servlet ທີ່ດັນບັນດຶງໄປ !

```

@WebServlet(
    name = "CourseListServlet",
    value = {"course-list"}
)
public class CourseListServlet extends HttpServlet {
    public CourseListServlet() {
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        request.setAttribute("semesters", Semester.getAllSemesterText());
        this.getServletContext().getRequestDispatcher("/CourseList.jsp").forward(request, response); //ສ່ວນ course list
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Map<String, String[]> parameterMap = request.getParameterMap(); //ເຫັນມູນຫໍ່ຄົມໄສ ຕອບ
        request.setCharacterEncoding("UTF-8"); * ສ້າງການເີ່ມນກາຍໄກນ
        if (parameterMap.get("semester") == null) {           | ກໍໄລຍ່ສ່ວນຈະໄນ້ກໍສັງ semester ມໃນບັນກລຸນາຈັງ course list ( doGet() )
            this.doGet(request, response);                   | ມີມີມົດຕ່າງ array
        } else {                                         / ມີມີມົດຕ່າງ
            int semester = Integer.valueOf((String[])parameterMap.get("semester"))[0];
            request.setAttribute("semesters", Semester.getAllSemesterText());
            request.setAttribute("selectedSemester", semester); //ມານັ້ນກ່ອນທີ່user ເລືອດເລືອດນິ້ນໄນ
            request.setAttribute("subjects", CourseRepository.getSubjects(semester)); //ນີ້ມີ 1 ດັວວນ collection
            this.getServletContext().getRequestDispatcher("/CourseList.jsp").forward(request, response);
        }
    }
}

```

1ພລ model (JAVA)

```

public class Semester {
    private static final String[] TITLE = new String[]{null, "ການ 1/ ປຶກສຶກຂາທີ 1", "ການ 2/ ປຶກສຶກຂາທີ 1",

    public Semester() {
    }

    public static String[] getAllSemesterText() { return TITLE; }

    public static String getSemesterText(int semesterNumber) {
        return semesterNumber >= TITLE.length ? null : TITLE[semesterNumber];
    }
}

```

```

<c:if test="${subjects != null}">
    <div class="container m-auto h-auto">
        <form action="register" method="post">
            <input type="hidden" name="semester" value="${selectedSemester}"> ↑ ชี้ว่าที่ต้องการส่งกลับไป
            <div class="row bg-white">
                <div class="col-1">ลำดับ</div>
                <div class="col-1">รหัส</div>
                <div class="col-5">ชื่อวิชา</div>
                <div class="col-1">หน่วยกิต</div>
                <div class="col-1">เลือก</div>
            </div>
            <c:forEach items="${subjects}" var="subject" varStatus="vs">
                <div class="row bg-transparent">
                    <div class="col-1">${vs.count}</div>
                    <div class="col-1">${subject.subjectId}</div>
                    <div class="col-6">${subject.title}</div> ↑ ที่พิมพ์มาแล้วกันดี
                    <div class="col-1">${subject.credit}</div> ↑ registeredSubject
                    <div class="col-1">
                        <input type="checkbox" name="registeredSubject" value="${subject.subjectId}"> ↑ เมนูแบบ checkbox ก็จะได้ตัว multiple value
                    </div>
                </div>
            </c:forEach>
            <hr>
            <div class="form-group"><input type="submit"></div>
        </form>
    </div>
</c:if>

```

JSP

```

@WebServlet(
    name = "RegisterCourseServlet",
    value = {"register"}
)
public class RegisterCourseServlet extends HttpServlet {
    public RegisterCourseServlet() {
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Map<String, String[]> parameterMap = request.getParameterMap(); // แปลง成 map value เป็น array
        int semester = Integer.valueOf(((String[])parameterMap.get("semester"))[0]); // ① ตัวแปรที่ร่วมกันระหว่างหน้าเบื้องหน้า
        * HttpSession session = request.getSession(); // เก็บค่าไว้ใน session
        CourseRegistered courseRegistered = (CourseRegistered)session.getAttribute(s: "courseRegistered");
        if (courseRegistered == null) { // เช็คต่อเมื่อแรกๆ
            courseRegistered = new CourseRegistered(); // ส่วนของข้อมูลจะมาใส่
            session.setAttribute(s: "courseRegistered", courseRegistered); ↑ ผูกของตัวอยู่บ้าง?
        } else {
            courseRegistered.removeAllRegisteredCourse(semester); getSession → vo session
            getAttribute → vo voใน Session
        }

        String[] var7 = (String[])parameterMap.get("registeredSubject"); // ② ตัวแปรที่ส่งมาแบบ multiple value
        int var8 = var7.length;

        for(int var9 = 0; var9 < var8; ++var9) {
            String subjectId = var7[var9]; ↑ หัวข้อเพื่อตรวจสอบว่าต้องการส่งกลับ!
            courseRegistered.registerSubject(semester, CourseRepository.getSubject(semester, subjectId));
            System.out.println(CourseRepository.getSubject(semester, subjectId));
        }

        System.out.println(); ↑ จบการเขียนผลลัพธ์
        this.getServletContext().getRequestDispatcher(s: "/index.jsp").forward(request, response);
    }
}

```

JPA

(Java Persistence API)

- layer ที่ชื่อ JPA ที่มี interface สำหรับการจัดการข้อมูล
- ฟังก์ชัน library , dependency
- ต้องมีตัวกลางฐานข้อมูล

(Auxiliary Device / External memory)

* Secondary Storage

- เอกชนุลักษณ์
- เป็นได้ทั้ง input + output
- กิ่งก้าน
- database
- Persistence data (เก็บกาว)

└ 临时数据 transient data

COMPUTER SYSTEMS

- เป็นไปตามแนวคิดของ Von Neumann

- ห้าใจสำคัญ! 1) data 2) instructions 2 คำสั่งทั้งหมดอยู่ใน main memory

└ เช่น 1 statement อาจแทนไปได้หลาย instructions
method เป็น instructions

* obj เป็นทั้ง data + instruction



class ที่ใช้เก็บข้อมูล
จาก table เราเรียก
กันว่า "Entity"

ORM (Object Relational Mapping) 乃即把 row - object

- 1 class สร้างเก็บลง obj เท็อน database (ข้อมูล 1 row = 1 object)

db (Entity Relation ผังงาน)

- primary key
- foreign key

(one to many)

Person 1 : m Address เช่น code ปะจัง?

class Person { Address address[]; }

or class Person { List<Address> addresses; }

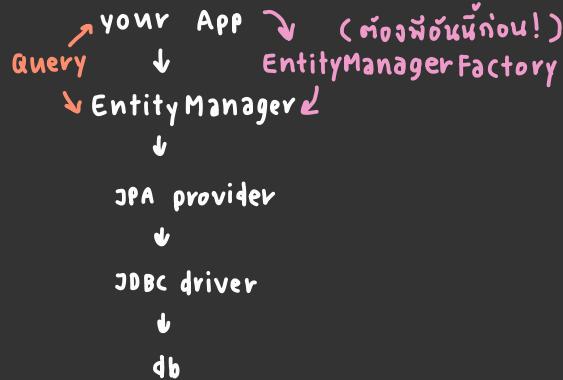
python (Object Relationship)

- Aggregation
- Inheritance

JPA 乃 package javax.persistence

- EntityManager Factory
 - Entity Transaction
 - EntityManager
 - Query
 - Persistence

JPA CLASS ARCHITECTURE



enterprise vendor (ที่ library ไม่ใช่) - Hybernate, Eclipselink, Toplink, Spring Data JPA

ENTITIES (ព័ត៌មានកុំគោលចាយពីការបង្កើត) *

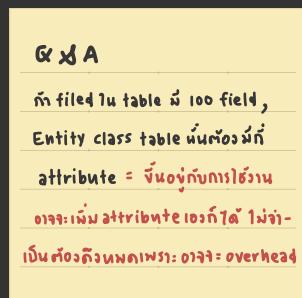
- plain old java object (POJO) java class ដែលមានរាយការណ៍ (Getter, Setter) នៅក្នុងការបង្កើត Entity ទាំងអស់
- class represent a table in a relational db
- Instances(java) = row(db) [class person ម៉ោង table person] + នៅក្នុង annotation នៃលោខាងក្រោមនេះ គឺជា entity
- Requirement
 - import ពី javax.persistence.Entity
 - class ត្រូវបាន public or protected (X private), ត្រូវបាន default constructure
 - អាមេរិយៈ final *

```
@Entity
@Table(name="USER")
public class User {
    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE, generator="UserSequence")
    private Long id; // primary key

    @Column(name="USERNAME", nullable=false)
    private String username;

    @Column(name="FULL_NAME")
    private String fullName;
    ...
}
```

| USER TABLE | | |
|------------|----------|----------------|
| ID | USERNAME | FULL_NAME |
| 1 | eatrocks | Bruce Campbell |
| 2 | mark | Mark Jones |



EX A

ការ map ពី table ដែលមាន 100 field, Entity class table ដែលមាន 3 field
attribute = ចំណាំការបង្កើត
បញ្ជី: ដែល attribute ខ្លះ នឹងត្រូវបង្កើតឡើង ដូចខាងក្រោម
បែងចែក overhead: 0.01 = overhead

ENTITY MANAGER

- create, remove និងការចុះឈ្មោះ
- find ឬស្វែនតារាបន្ទូន
- សរ៍វា queries ។

Ex User user = entityManager.find(User.class, 1002);

create

L មិនមែន method នៃ class persistence javax.persistence.Persistence

L root class voi EntityManager

L បានចិត្តពី persistence unit (persistence.xml)

L ត្រូវបង្កើតឡើង driver db

1) ចិត្តពី persistence unit

2) ចិត្តពី class persistence របស់ក្នុង method (ធ្វើការបង្កើត EntityManagerFactory)

3) សរ៍វា EntityManager

- EntityManagerFactory emf = Persistence.createEntityManagerFactory("persistence unit name from persistence.xml");
- EntityManager em = emf.createEntityManager();

```
User user = em.find(User.class, 1002);
Customer c = em.find(Customer.class, 2001);
Subject subject = em.find(Subject.class, "INT202");
Product product = em.find(Product.class, "S12_101");
Employee emp = em.find(Employee.class, 1002);
```

ចិត្តពី path

```
<persistence-unit name="default">
    <class>sit.int202.classicmodelweb.entities.Office</class>
    <properties>
        <property name="jakarta.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver"/> 1) driver
        <property name="jakarta.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/classicmodels"/> 2) url
        <property name="jakarta.persistence.jdbc.user" value="root"/> 3) username
        <property name="jakarta.persistence.jdbc.password" value="143900"/> 4) password
    </properties>
</persistence-unit>
```

persistence.xml