

JSTL : JSP Standard Tag Library

pichet@sit.kmutt.ac.th

JSP Standard Tag Library (JSTL)

- Encapsulates as tags the core functionality of many Web applications
- Supports tasks such as:
 - Flow (iteration and conditionals)
 - Manipulation of XML documents
 - Internationalization tags
 - SQL tags
- Java EE 1.4 includes both JSP tags and the JSTL
 - JSTL taglibs are included with Rational Application Developer / NetBean IDE

Sample JSTL tags

- Set a variable in a specific scope to a value
`<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"/>!`
- Conditional execution
`<c:choose>`, `<c:when>` and `<c:otherwise>`

```

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

```

forEach tag

- The forEach tag provides flexible iteration through a set of items, Its targets include:
 - Collections, Maps, Iterators, Enumerations
 - Arrays
 - Comma-separated values (CSV) data

- Example:

```
<table>
  <c:forEach items="{customers}" var="cust">
    <tr>
      <td>${cust.name}</td>
      <td>${cust.addr}</td>
    </tr>
  </c:forEach>
</table>
```

Exercise

Menu

1) [ลงทะเบียน](#)

2) [ประวัติการลงทะเบียน](#)

3) [Clear History](#)

~~4) [Logout Cookies](#)~~

Semester : ภาค 2/ ปีการศึกษาที่ 4

Search

ลำดับ รหัสวิชา	ชื่อวิชา	หน่วยกิต	เลือกลงทะเบียน
1 INT308	Security II	2.0	<input checked="" type="checkbox"/>
2 INT319	Information Technology Professional Practice	4.0	<input type="checkbox"/>
3 INT321	Information Technology Seminar I	1.0	<input type="checkbox"/>
4 INT339	Preparation for Career Training	1.0	<input type="checkbox"/>
5 INT340	Career Training**	2.0	<input checked="" type="checkbox"/>

Submit

Back

ประวัติการลงทะเบียน

ภาค 1/ ปีการศึกษาที่ 1

ลำดับ รหัสวิชา	ชื่อวิชา	หน่วยกิต
1 INT100	Information Technology Fundamentals	3.0
2 GEN101	Physical Education	1.0

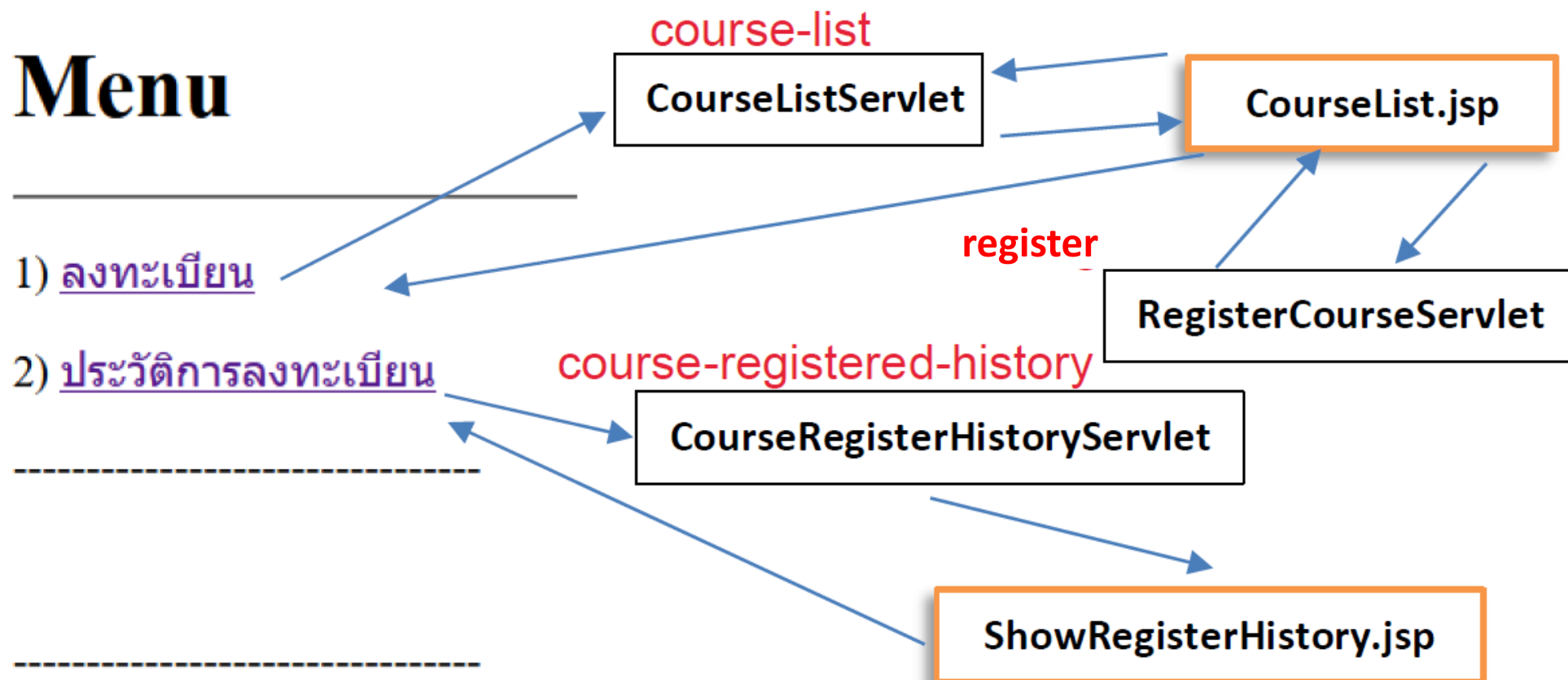
ภาค 2/ ปีการศึกษาที่ 4

ลำดับ รหัสวิชา	ชื่อวิชา	หน่วยกิต
1 INT308	Security II	2.0
2 INT340	Career Training**	2.0

ภาคพิเศษ โครงการ WIL

ลำดับ รหัสวิชา	ชื่อวิชา	หน่วยกิต
1 GENxxx	GEN Elective I	3.0
2 INT366	Capstone Information Technology Project II	3.0

URL Mapping



Application State Management

The session problem

- Servlets and JSP pages should be stateless
 - They should not have instance variables
 - They support multiple concurrent threads
- Application state information specific to a user must be stored outside of the servlet
 - This is called Session State
 - Determining what to do with it is one of the most challenging problems in servlet programming
 - Session state is session data, and is only used for a set of linked Web pages (your Web application)

Session strategies: review

- Session state can be stored on either the client or the server
- Examine several session management strategies and figure out where each applies

Client Side

- Cookies
- Hidden Fields
- URL Rewriting

Server Side

- HTTP Session
- Content Based Routing
- Store state in a database
- JWT: JSON Web Token

*store session state
in server's main memory*

Cookies

ข้อมูลที่เราเก็บไว้ที่ client

- Cookies are a way to place persistent information on the client machine (accessible from the browser) เข้าถึงข้อมูลได้ผ่าน browser
 - A good way to handle preferences or shortcuts
- Cookies have a name and a value

browser เก็บ cookies ที่ client ส่งไป header ของ response

server ส่งกลับ cookies มา: กำหนดค่า/ตัว

browser ใช้ cookies



Cookie API

- Creating cookies *String only*
 - `Cookie(String name, String value)`
- Sending a cookie back to the browser
 - `HttpServletResponse.addCookie(Cookie aCookie)`
- Retrieving cookies *has collection of 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)`

Cookie usage example

```
public void doGet(HttpServletRequest req, HttpServletResponse res) {  
    String userType = "novice";  
    Cookie[] cookies = req.getCookies();  
    if (cookies != null) {  
        for (int i=0; i<cookies.length; i++) {  
            if(cookies[i].getName().equals("userType"))  
                userType = cookies[i].getValue();  
        }  
    }  
    if (userType.equals("expert"))  
        // do expert HTML  
    else  
        // do novice HTML  
}
```

Proper cookie usage

- Because cookies are stored as plain text on the client machine, cookies can be viewed and altered by the client
- Cookies should not be used for information such as:
 - Validation information
 - Secure information (credit card numbers and the like)

Cookie applicability

- Cookies have an expiration date
 - `setMaxAge(int expiryInSeconds)`
- Default expiration date is -1 *only exists in browser*
 - This means that the cookie is not stored persistently
 - It lasts only as long as the browser is open
- Can restrict the applicable URLs to which a cookie will be sent
 - `setDomain(String)`
 - `setPath(String)`

Client data and session tracking with cookies

- Cookies can be made to persist within or across browser interactions
- Cookies are passed to the Web server in the header of the request
- Any updates are passed back in the header of the response
- Session data tracking via HTTP cookies is the most commonly used session tracking mechanism
 - Required to be supported by all Web containers

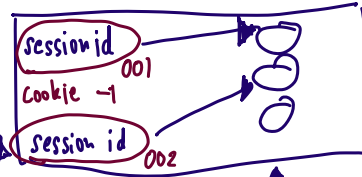
browser request assign
app server assigns session id via browser

Browser



1st request to
no cookie

App server



http session

Http Session

1 http session ကို user 1 အား

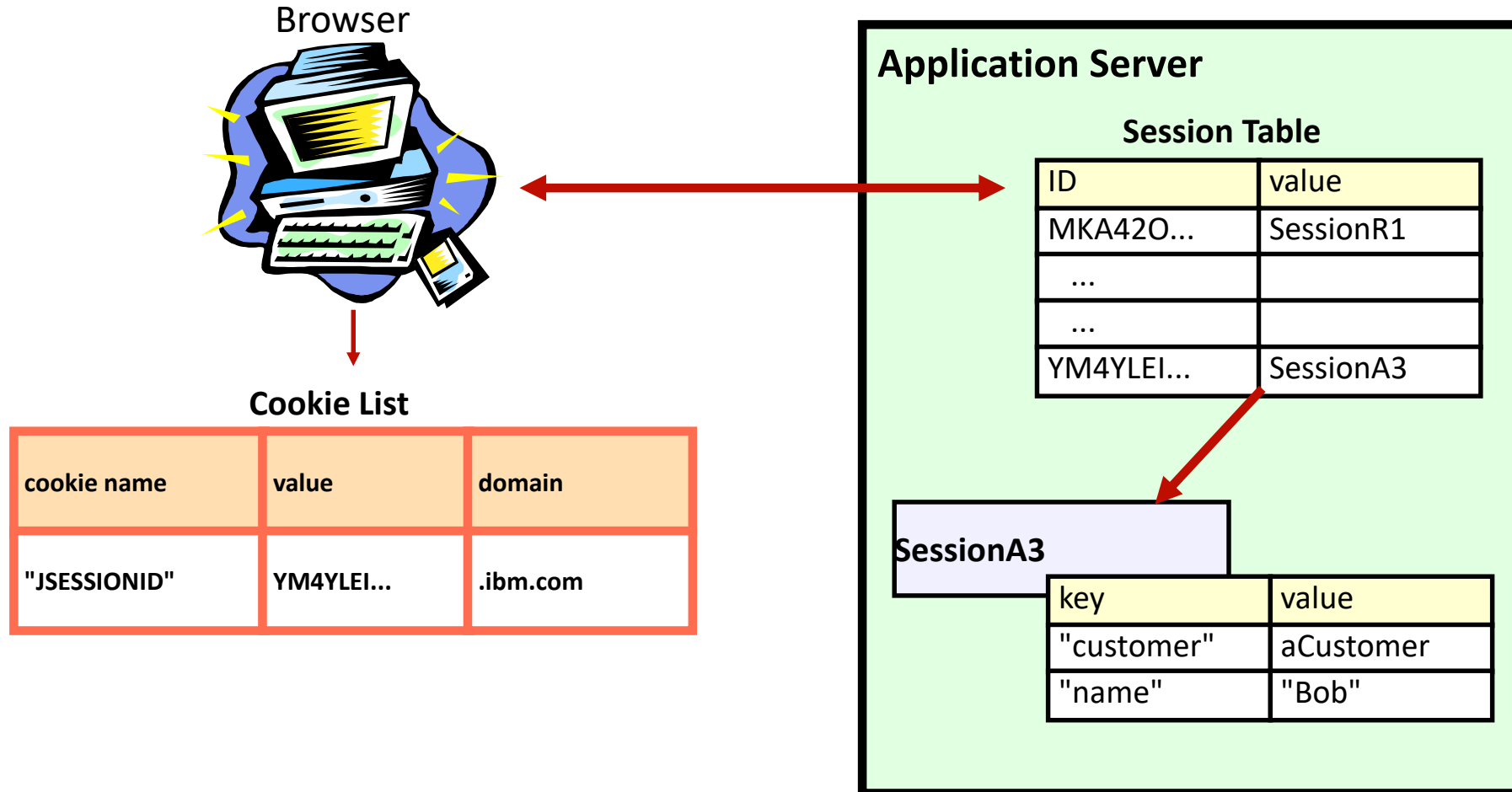
- The HttpSession interface, part of the Servlet API, provides an interface for managing application state on the server
- Session Usage:
 - Servlet asks to bind to the Session object representing the current session:
 - A session is requested – request.getSession(boolean create)
 - The method returns 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 (method), ခုတ်ပစ် လာ log out
 - The session times out
မသုံးတော့မှ အား resource ပေါ် server

parameter : true ကိုယ် ခုတ်ပစ်/ဖန်တီး
false return null

HttpSession data store

- HttpSession store application-specific information
 - Stored as <"key", object> pairs
 - void setAttribute(String, Object)
 - Object getAttribute(String)

Sessions at run time



Session invalidation

မှ method မျှော် session ကို ဖျက်ပစ်ရန်

- Release HttpSession objects when finished
 - An Application Server can only maintain a certain number of HttpSession objects in memory
- Sessions can be invalidated either programmatically or through a timeout
 - session.invalidate
 - Removes all values from the session
- The session timeout (inactive interval) can be set for the application server as a whole
 - The default timeout is 30 minutes
- Also session.setMaxInactiveInterval(int) can provide session-specific timeout value

set ကို သတ်မှတ်ရန်

Session invalidation example

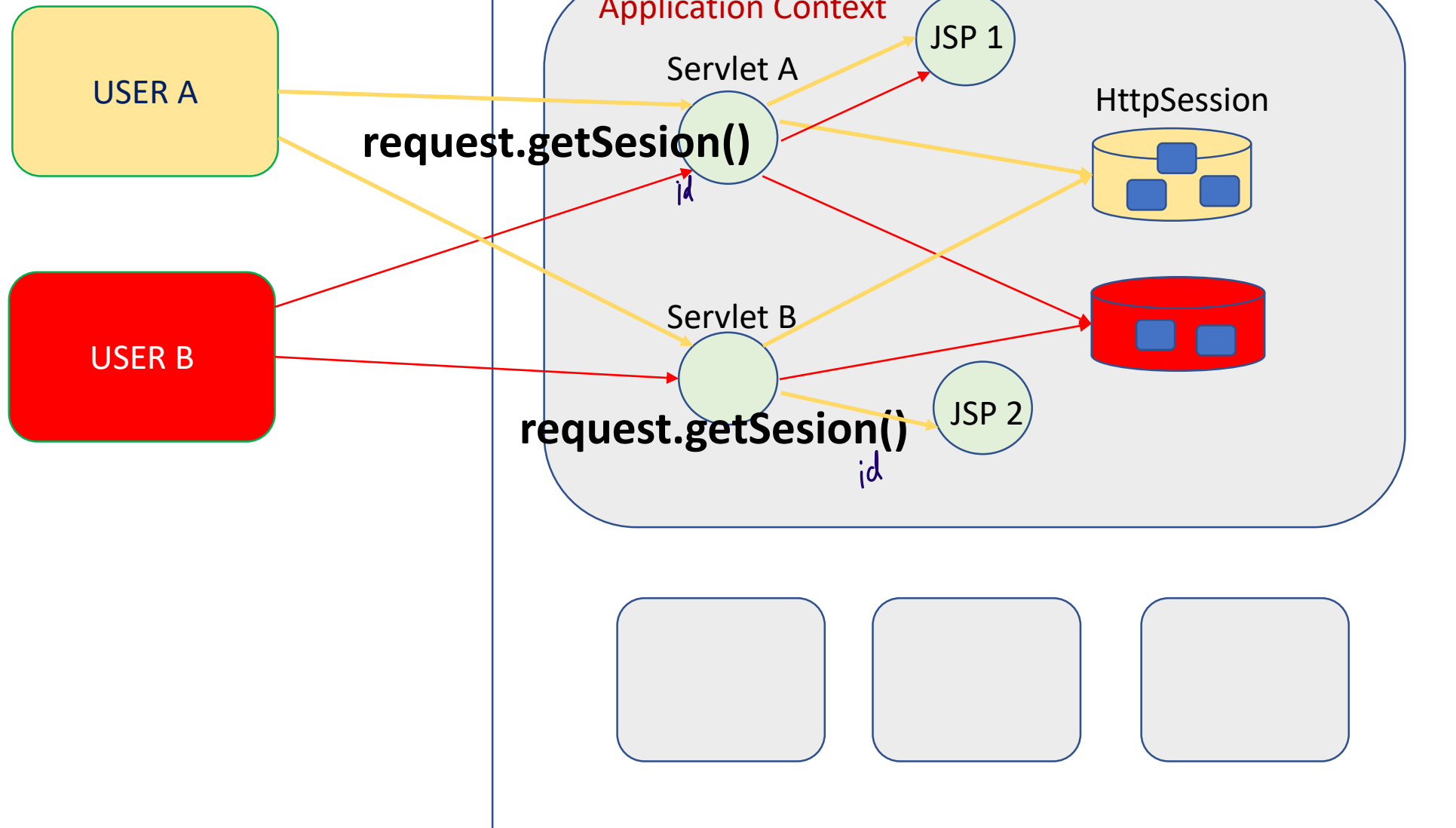
```
public class ApplicationLogoutServlet extends HttpServlet {  
    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws  
        ServletException, IOException {  
  
        HttpSession mySession = req.getSession(false);  
  
        // Invalidate session  
        if (mySession != null) {  
            mySession.invalidate();  
        }  
        // Perform additional application logoff processing  
        // and send output response to browser here  
    }  
}
```

Session serialization

(only session to database)
must implement serializable

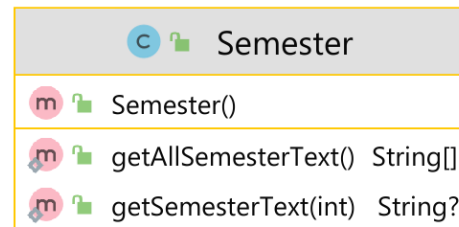
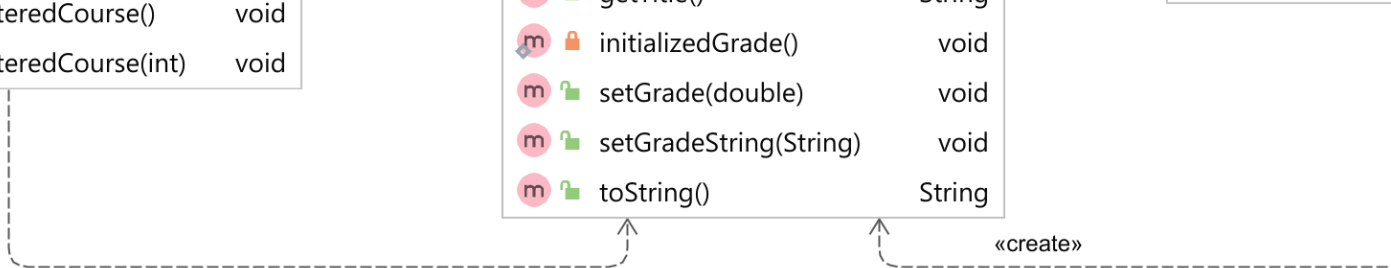
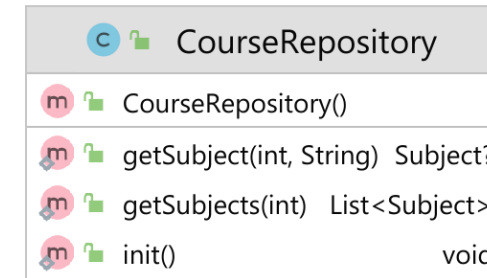
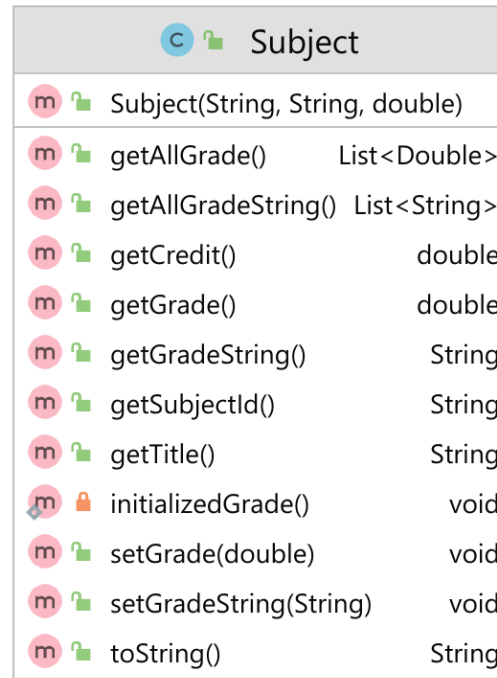
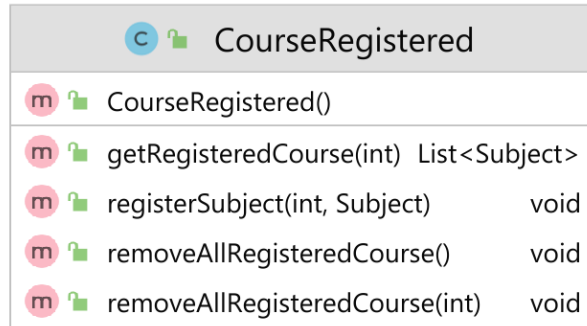
- Objects stored in a session must be serializable:
 - To share between servers in a clustered server configuration
 - For persistence to work
- Make sure that objects reachable from the session are also serializable
- When creating objects to be stored in the session, implement the serializable interface as follows :

```
public class NewObject implements java.io.Serializable {  
    ...  
}
```



Excercise

Model



CourseRegistered & CourseRepository

list

CourseRegistered		
f	registeredSubjects	Map<Integer, List<Subject>>
m	getRegisteredCourse(int)	List<Subject>
m	registerSubject(int, Subject)	void
m	removeAllRegisteredCourse()	void
m	removeAllRegisteredCourse(int)	void

CourseRepository		
f	DATA_FILE	String
f	courses	Map<Integer, List<Subject>>
m	getSubject(int, String)	Subject?
m	getSubjects(int)	List<Subject>
m	init()	void

ข้อมูลในแต่ละ semester (resources/data.txt)

- Semester: 1

INT100	Information Technology Fundamentals	3.0
INT101	Programming Fundamentals	3.0
INT102	Web Technology	1.0
INT114	Discrete Mathematics for IT	3.0
GEN101	Physical Education	1.0
GEN111	Man and Ethics of Living	3.0
LNG120	General English	3.0
LNG220	Academic English	3.0

- Semester: 2

INT103	Advanced Programming	3.0
INT104	User Experience Design	3.0
INT105	Basic SQL	1.0
INT107	Computing Platforms Technology	3.0
INT200	Data Structures and Algorithms	1.0
GEN121	Learning and Problem Solving	3.0
LNG220	Academic English for Science	3.0

Semester: 8

INT322 Information Technology Seminar II	1.0
INTxxx INT Elective II	3.0
INTxxx INT Elective III	3.0
LNG304 Meetings and Discussions	1.0
XXXxxx Free Elective II	3.0

Semester: 9

INT372 Experiential Learning Project II	3
INTxxx INT Elective I	3
LNG224 Oral Communication I	3
GENxxx GEN Elective I	3
GENxxx GEN Elective II	3
XXXxxx Free Elective I	3
INT308 Security II	2
INT319 Information Technology Professional Practice	4
INT322 Information Technology Seminar II	1
LNG304 Meetings and Discussions	1
GEN351 Modern Management and Leadership	3
GENxxx GEN Elective II	3
XXXxxx Free Elective II	3

Creating Simple Course Registration App



1. Create new Project (Maven) Project
 - Name: **register**
 - Template: **Web Application**
 - Group: **sit.int202**
 - Application server: **Tomcat 10.0.xxx**

 - Version: **Jakarta EE 9**
2. Add Lombok, JSTL Dependency
3. Test test_css_jstl.jsp with BootStrap, JSTL & EL using code below

```
<div style="padding: 10px;display: block;width: 50%;margin: auto">
  <c:forEach begin="1" end="100" var="value">
    <div class="box">${value}</div>
  </c:forEach>
</div>
```

[illegible]

Creating Simple Course Registration App (2)

4. Create package `sit.int202.register.models`
5. Copy all models (4) to package `sit.int202.register.models`
6. Copy `data.txt` to resources
7. Create new servlet `CourseListServlet` then change url-mapping to `course-list`
8. place code below to `doGet()` method of `CourseListServlet`

```
request.setAttribute("semesters", Semester.getAllSemesterText());  
  
getServletContext().getRequestDispatcher("/course_list.jsp").forward(  
request, response);
```

Creating Simple Course Registration App (3)



9. Create view : `course-list.jsp`
10. Modify `course-list.jsp` (download from MS-Teams)
11. Implementing code in method `doPost` of `CourseListServlet`

```
Map<String, String[]> parameterMap = request.getParameterMap();
request.setCharacterEncoding("UTF-8");
if (parameterMap.get("semester")==null) {
    doGet(request,response);
    return;
}
int semester = Integer.valueOf(parameterMap.get("semester")[0]);
request.setAttribute("semesters", Semester.getAllSemesterText());
request.setAttribute("selectedSemester", semester);
request.setAttribute("subjects", CourseRepository.getSubjects(semester));
getServletContext().getRequestDispatcher("/course-list.jsp").forward(request,response);
```


Creating Simple Course Registration App (4)

12. Create new servlet **RegisterCourseServlet** then change url-mapping to **/register**
13. Implementing code in method doPost of **RegisterCourseServlet** (use **snipped code from next-slide**)
14. Create new servlet **CourseRegisterHistoryServlet** then change url-mapping to **/course-registered-history**
15. Create view : **user_registered.jsp**
16. Implementing **user_registered.jsp** for display all registered subject from HttpSession (DIY)

RegisterCourseServlet.java



```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    Map<String, String[]> parameterMap = request.getParameterMap();
    int semester = Integer.valueOf(parameterMap.get("semester")[0]);

    HttpSession session = request.getSession();
    CourseRegistered courseRegistered = (CourseRegistered) session.getAttribute("courseRegistered");
    if(courseRegistered == null) {
        courseRegistered = new CourseRegistered();
        session.setAttribute("courseRegistered", courseRegistered);
    } else {
        courseRegistered.removeAllRegisteredCourse(semester);
    }

    for(String subjectId : parameterMap.get("registeredSubjects")) {
        courseRegistered.registerSubject(semester, CourseRepository.getSubject(semester,subjectId));
    }
    getServletContext().getRequestDispatcher("/index.jsp").forward(request,response);
}
```