Finals Webtek Lec Notes

**Server-Side Web Scripts**

-Java (Servlets/JSP)

-PHP

-Node.js

-ASP.Net

Java Versions

* Java SE
* Java ME
* Java EE

-focuses on Java applications

-Server: Apache Tomcat 8.0.27.0

: GlassFish Server 4.1.1

Apache Tomcat- has a servlet container

XXX-standard-http methods

getInputStream- Binary

Conste- http connector

CRUD

* Create
* Retrieve
* Update
* Delete

HTTP servlet response- set headers  
\*all processing happens on the server side

Socket programming- host to a point

Java Bean- follow naming conventions of Java

Forward- call the next server

Forward- lose control

Include- control returns back to you

Request 🡪 Forward 🡪 Display

Java Servlet Pages (JSP)- server in disguise

JSP- you can embed dynamic elements ex. Javascript

-generate by Jasper from Apache Tomcat

<%---JSP comment---%>

[%@page import .”java.util.Random”%](mailto:%25@page%20import%20.”java.util.Random”%25)

JSP Template Text- static  
 - xhtml, json, etc.

JSP: include- standard action

<JSP: element name=”a”>  
 <JSP: attribute name =”href”> 🡪generates body  
 <http://www.slu.edu.ph>

EL- Expression Language

<%--EL Expression--%>  
<p>Hello, ${param.name};</p>  
<p>Your browser is ${header[‘User-Agent’]} </p>

JSP🡪 request scope

${requestScope.student.idno}

<%---...scriplet…---%>  
<%  
 int i=0;  
 for (; i<5; i++){  
 out.println(“hello”);  
}

JSP- for generating markup  
servlet- large application

Session Handling Java Applications- stateless

Cookie- small text info, establishes session

Set-cookie: cki

Non-persistent cookies- when browser is closed, the cookie jar is empty

Persistent cookies- usually for experience personalization

HTTP cookie

HTTP only cookie

request.getSession- to see if there is an incoming cookie

Server Side Development using PHP

Markup to scope in php scriptmode

<?php  
 echo “hi world@!”;  
 ?>

file\_get\_contents (“php://input”) 🡪 get contents from a file

Node.js 🡪 mean stack 🡪(Monggodb, express.js, Angular.js, node.js)

Node.js- come with node package manager

var http=require(‘http’); 🡪 load  
var server=http.createServer(); 🡪 setup  
server.listen(81);

function handler(request,response){  
 response.end(‘hello, node JS!’);

header [‘user-agent’] –returns value of user agent

Web Application Security

Client Server 🡪User-agent

Web server 🡪 App

Client Server 🡪🡨 Web Server

OWASP #1 🡪 Injection

Htmlspecialchars- So html will be rendered as plain text.