Note 1

1.GET là default method của HTTP.
Độ lớn của Query string trong URL phụ thuộc vào từng Web Server. GET là Idempotent và có thể bookmark

2.POST là not-idempotent và có thể bookmark

3.Web component (JSP & Servlet) được deploy trên servlet container (e.g Tomcat
Servlet container thường có Web Server bundled. Một JEE AS bao gồm EJB container (deploy business component là EJB) và Servlet container.

4.Vai trò của Servlet container :

- Communications support : handles communication between the servlets and the webserver
- Lifecycle management : controls life and death of servlets, class loading, instantiation, initialization, invocation the servlets' methods, and making servlet instances eligible for GC
- Multithreading support : Automatically creates a new thread for every servlet request received. When the Servlet service() method completes, the thread dies.
- Declarative security : manages the security inside the XML deployment descriptor file. Security can be configured and changed only by modifying this file.
- JSP support : Manages the JSPs.

5.Khi user (browser) make một HTTP request, container sẽ tạo 2 object tương ứng là 2 instant của HttpServletRequest và HttpServletResponse class:

Container xác định servlet cần gọi dựa vào URL (url-pattern), tạo hoặc cấp phát thread cho request, sau đó call service() method của servlet và truyền request/response object đã tạo cho method này. Service() method gọi các doXXX() method của servlet tùy theo HTTP request từ client. Sau khi service() method hoàn thành, thread kết thúc (die state), request/response không được tham chiếu đến nữa và chúng sẽ đc thu dọn bởi GC

6.Servlet life cycle:

a.init(): Chỉ được gọi một lần trước khi servlet xử lý các request (trước khi service() được gọi). Method này có thể override, được sử dụng để khởi tạo các resource.
b.service(): Được gọi bởi container khi client make request, có thể được gọi nhiều lần để xử lý các request. Service() dựa vào HTTP request để gọi các doXXX() tương ứng. Method này được xử lý trong từng thread riêng biệt. Không nên override.
c.doXXX(): được gọi bởi service() tùy theo HTTP request. Các method này chứa code thực sự xử lý request. Nếu doXXX tương ứng với HTTP request không được override, thì doXXX trong HttpServlet class sẽ được gọi và trả về message "the HTTP method is not implemented."
d.destroy(): Chỉ được gọi chính xác 1 lần bởi container. Có thể override để clean resource

7.Chỉ có một (và chỉ một) instant của servlet (trong một JVM). Tất cả các request đến servlet được thực thi trong các thread riêng biệt

8.ServletConfig:

Chỉ một ServletConfig chi một servlet
ServletContext: Chỉ một ServletContext cho một web app (trong một JVM). ServletContext được dùng để lấy các thông tin về môi trường thực thi của container.
Có 2 cách để lấy ServletContext:
-getServletContext(); (method của Servlet interface)
-getServletConfig.getServletContext();
Hai cách trên đều dùng được trong Servlet. Cách thứ 2 có ích khi ta muốn get ServletContext từ các object không phải là Servlet như các POJO class bằng cách truyền ServletConfig qua lại giữa các class.
Chú ý: Chúng ta chỉ có thể get một object từ một object khác mà có life time nhỏ hơn. VD: get ServletContext từ ServletConfig vì ServletConfig có life time (scope) nhỏ hơn.

9.Các thành phần trong URL request:

http://www.domain.com:80/myapp/path/resource?name=value
/myapp --------> Context Path (request.getContextPaht())
/path --------> Servlet Path (request.getServletPath())
/resource --------> Path Info (request.getPathInfo())
name=value --------> Query String (request.getQueryString())
/myapp/path/resource?name=value ---------> Request URI (request.getRequestURI())
http://www.domain.com:80/myapp/path/resource?name=value --> Request URL (request.getRequestURL())

10.response.setHeader():

Tạo mới header hoặc ghi đè giá trị cũ nếu có
response.addHeader(): tạo mới header hoặc thêm giá trị vào header đã có

11.Listener:

Các listener được notify theo thứ tự được khai báo trong web.xml, nhưng khi shutdown thì thứ tự được notify theo chiều ngược lại

12.Attribute:

Các attribute không thread-safe là các attribute của ServletContext và Session. Như vậy để thread-safe các attribute này, chúng ta phải synchronize

```
synchronized (getServletContext()) {
//code goes here
}

synchronized (getSession()) {
//code goes here
}
```

Tóm lại:

- Các attribute là thread-safe bao gồm:
o Request attributes
o Method local variables
- Các attribute không là thread-safe bao gồm:
o Context attributes
o Session attributes
o Servlet instant variables (vì chỉ có duy nhất 1 instant của servlet cho mọi request được run đồng thời trong các thread khác nhau)
o Servlet static variables
SingleThreadModel hiện nay được coi như deprecated, bản chất là synchronize service() method của servlet do đó sẽ chỉ có 1 thread cho một request (đến servlet đó) được run tại một thời điểm nhưng vẫn không thể thread-safe các attribute của ServletContext và Session.

13.RequestDispatcher:

Có 2 cách để obtain RequestDispatcher:
- request.getRequestDispatcher(uri: String)
uri có thể là relative (không có slash ở đầu (/)) hoặc absolute với current resource
- getServletContext().getRequestDispatcher(uri: String)

uri bắt buộc phải là absolute với current resource
Các request attribute đc dùng để lấy thông tin từ trang đc forward/include qua RequestDispatcher

```
javax.servlet.forward.request_uri

javax.servlet.forward.context_path

javax.servlet.forward.servlet_path

javax.servlet.forward.path_info

javax.servlet.forward.query_string



javax.servlet.include.request_uri

javax.servlet.include.context_path

javax.servlet.include.servlet_path

javax.servlet.include.path_info

javax.servlet.include.query_string
```

14.Session

Session tracking được thực hiện qua cookie hoặc URL Rewriting
- Cookie là cơ chế default để quản lý session. Session tracking qua cookie là thêm một mẩu thông tin để lưu giữ id của session, id này được lưu trong header của request/response để truyền qua lại giữa client và server và thường có tên (cookie name) là jssseionid (tên này phụ thuộc vào từng container)
- URL rewriting là thêm sessionid vào url, phương pháp này có tác dụng khi mà cookie không làm việc (thường do browser disable cookie). Tuy nhiên để sử dụng cách này thì url phải được encode bằng cách:
response.encodeURL(url: String)
response.encodeRedirectURL(url: String): URL Rewriting trong trường hợp redirect

15.Session creation:

- request.getSession(): nếu session đã có thì trả lại, nếu chưa có thì tạo mới
- request.getSession(bool)
o bool = false: nếu session đã có thì trả lại, nếu chưa có thì return null
o bool = true: làm việc như request.getSession().

16.Session method:

- isNew(): return true nếu client không gửi lại session id, hoặc gửi lại session id không tồn tại, khi đó request.getSession() sẽ tạo session mới và isNew() sẽ return true.
- setMaxInactiveInterval(interval): interval có đơn vị là second (khác với session timeout được set trong web.xml là minute), nếu interval < 0 thì session sẽ invalidate ngay lập tức

17. Session Listener

18. JSP life cycle

Với request đầu tiên sau khi được deploy, JSP phải trải qua 2 step là translation và compilation
- Translation: translate jsp sang servlet. JSP syntax được kiểm tra tại step này
- Compilation: container compile servlet sang bytecode (.class file). Java language/syntax được kiểm tra tại step này.

19. Generated Servlet API

JSP translate thành Servlet bao gồm 3 methods sau:
jspInit(): được gọi bởi init() method (từ HttpServlet), method này có thể được override
jspDestroy(): được gọi bởi destroy() method, method này cũng có thể được override
_jspService(): được gọi bởi service() method, method này không nên override

Chú ý: Những method bắt đầu bằng underscore (_) có nghĩa không nên override

20. Standard action

a. Bean declaration
Standard action tag cho bean là <jsp:useBean>
Có các attribute:
- id attribute:
chỉ ra tên của attribute, được set trong servlet với setAttribute() method
`<jsp:useBean id="titi" class="" scope=".."> ⇔ page.setAttribute("titi",obj)`

-scope attribute
Chỉ ra scope mà attribute được lưu. Có các scope sau:
page (default scope), request, session, application

```
<jsp:useBean id="titi" scope="session" > ⇔ session.setAttribute("titi",obj)
```

-class attribute
-type attribute
`<jsp:useBean id="", class="", scope="">`
chỉ tạo instant của bean nếu như không có object nào của class có tên như id được lưu trong scope
<jsp:useBean> có thể có body và body chỉ được thực thi khi và chỉ khi bean được tạo mới.
b.Bean property tags

setProperty tag có 4 attributes:
- name: bean name
- class: bean type
- property: name of the bean property
- value: the value to give to the bean property
- param: name of the request parameter
param và value là loại trừ lẫn nhau. Sử dụng param attribute khi muốn set property giá trị từ request parameter
`<jsp:setProperty name="titi" property="userName" param="fName">:`

Lấy giá trị từ fName request parameter và gán cho userName property của titi bean

```
<jsp:setProperty name="titi" property="userName" >
```

(không có param attribute) sử dụng khi request parameter và bean property có cùng tên. Trong trường hợp này là set value cho userName property từ userName request parameter.
`<jsp:setProperty name="titi" property="*" >` dùng khi set tất cả các property trong bean (kể

cả các property được kế thừa) với giá trị của tất các request parameter nếu chúng có cùng tên.
Chú ý: String value của request parameter sẽ tự động convert sang primitive data type của bean propery, nhưng container sẽ không tự động convert trong trường hợp sử dụng scripting như sau:

```
<jsp:setProperty name=”n” property=”age” value=”<%=
request.getParameter(“age”) %>” />
```