

SERVLETS II

CS 434

T. Asma Alsaleh and T. Lamees Alhazzaa

Outlines

2

- Overview of servlets.
- **Servlets life cycle.**
- Handling the client request:
 - ▣ Form data.
 - ▣ **HTTP request headers.**
- Generating the server response:
 - ▣ HTTP status codes.
 - ▣ HTTP response headers.
- Handling cookies.
- Session tracking.

Servlets life cycle

Servlet Lifecycle

4

- ❑ Servlets are Java programs which run inside a Servlet Container.
- ❑ A Servlet Container is much like a Web Server which handles user *requests* and generates *responses*.
- ❑ Servlet Container is responsible for loading and maintaining the lifecycle of the a Java Servlet.

Servlet Lifecycle

5

- Each servlet has the same life cycle:
 1. A server loads and *initializes* the servlet (call `init()`).
 2. The servlet handles zero or more client requests.
 3. The server *removes* the servlet (call `destroy()`).

Servlet Lifecycle

6

1. When the servlet is first created, its **init** method is invoked.
 - ▣ So that is where you put onetime setup code.
2. After this, each user request results in a thread that calls the **service** method of the previously created instance.
 - ▣ Don't override this method!

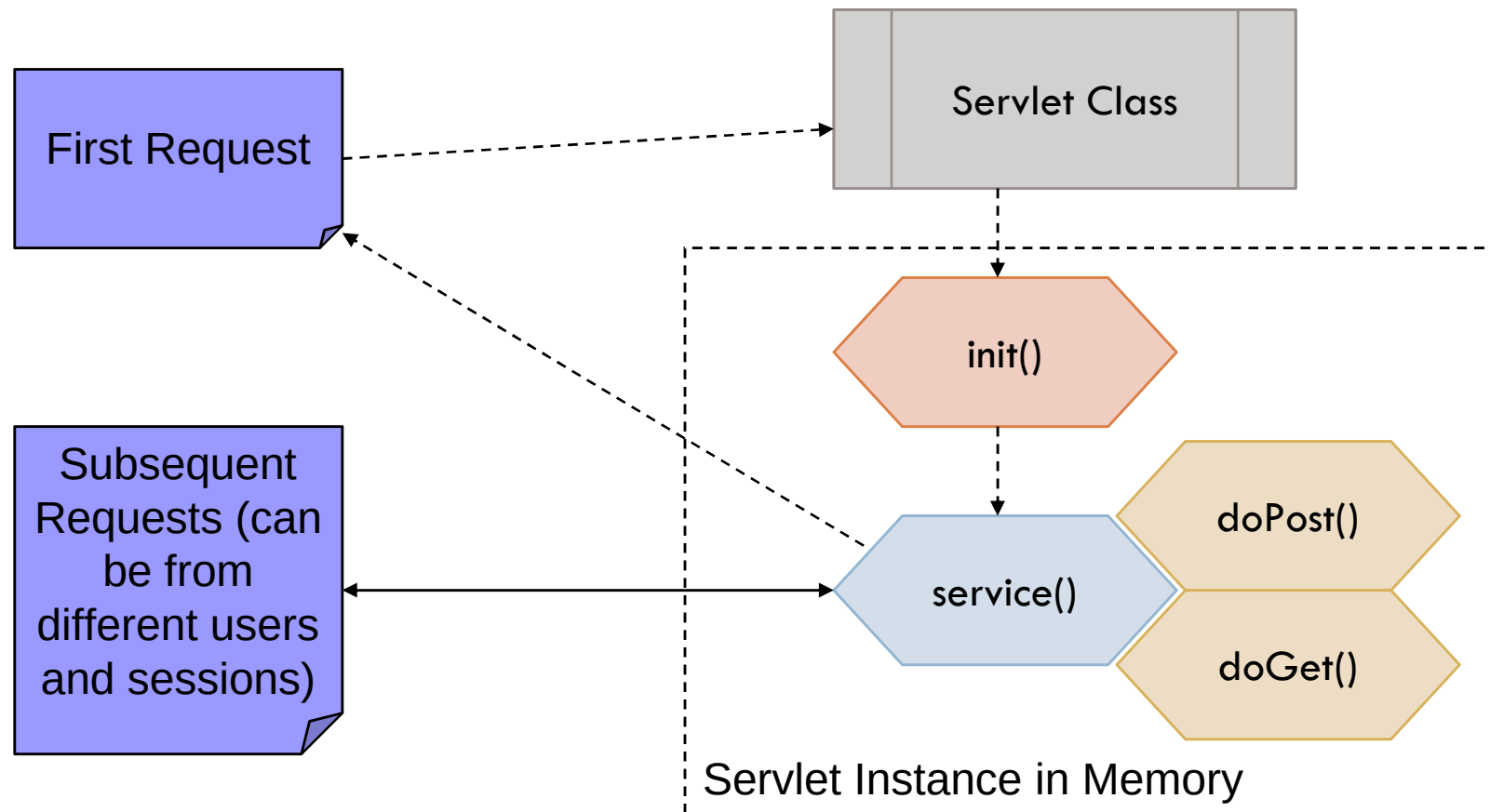
Servlet Lifecycle

7

3. The **service** method then calls **doGet**, **doPost**, or another **doXxx** method, depending on the type of HTTP request it received.
 - ▣ Handles GET, POST, etc. requests.
 - ▣ Override these methods to provide desired behavior.
4. Finally, when the server decides to unload a servlet, it first calls the servlet's **destroy** method.
 - ▣ Not called after each request

Servlet Lifecycle

8



(The slides of Jack G. Zheng <www.java-web.net/cis3270fall06/lecture/Servlet1.ppt>)

Why You Should *Not* Override `service`

9

- ❑ Overriding `service` precludes the possibility of
 - ❑ Add support for modification dates by calling `getLastModified()`.
 - ❑ Add support for other HTTP requests.
- ❑ If you have a servlet that handle both POST & GET requests identically?

```
public void doPost(HttpServletRequest request,
                    HttpServletResponse response) ...
{
    doGet(request, response);
}
```

HTTP request headers

A Typical HTTP Request

11

GET /servlet/Search?keywords=servlets+jsp HTTP/1.1

Accept: image/gif, image/jpg, */*

Accept-Encoding: gzip

Connection: Keep-Alive

Cookie: userID=id456578

Host: www.somebookstore.com

Referer: http://www.somebookstore.com/findbooks.html

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)

Reading Request Headers

(Some HttpServletRequest Methods)

12

1. General Methods:

- ▣ **getHeader**: returns a String if the header was supplied on this request, null otherwise.
 - `request.getHeader(headerName)`
- ▣ **getHeaderNames**: an Enumeration of all header names received on this particular request.
 - `request.getHeaderNames()`

Reading Request Headers

(Some `HttpServletRequest` Methods)

13

2. Specialized Methods:

- ▣ E.g. **`getCookies`**: returns the contents of the Cookie header, parsed and stored in an array of Cookie objects .
 - `Cookie[] cookies = request.getCookies();`
 - More examples in chapter 8.

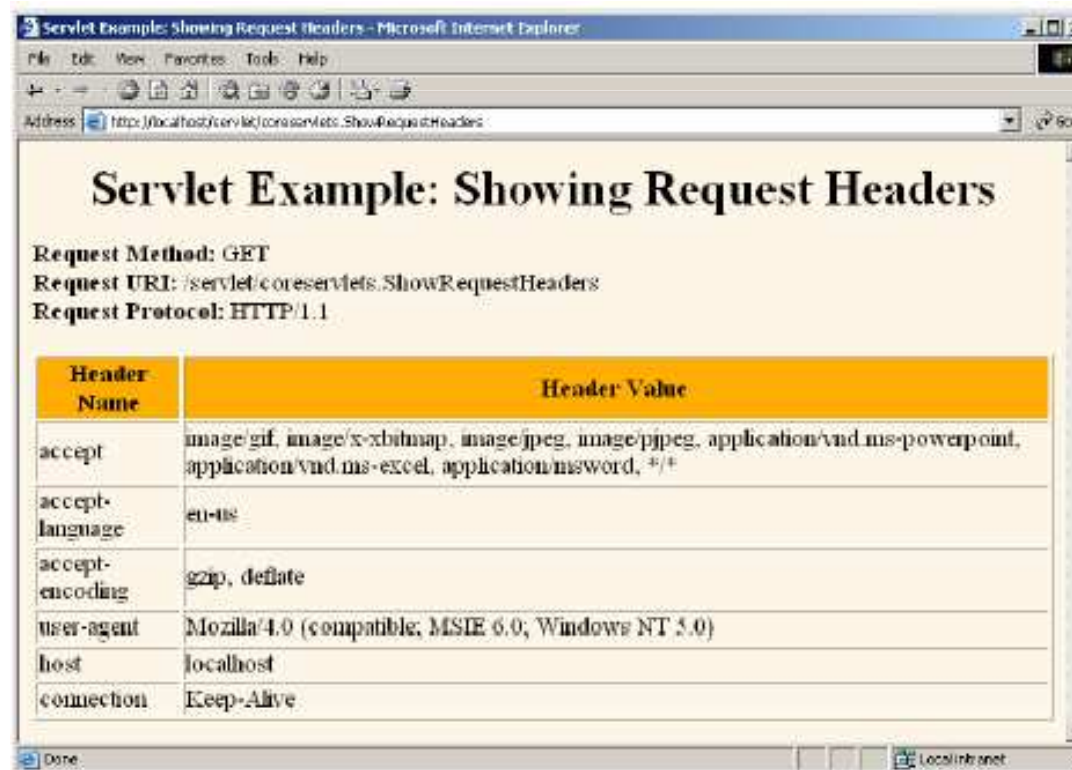
3. Related info:

- ▣ Get information of the main request line itself.
- ▣ **`getMethod`, `getRequestURI`, `getProtocol`**
 - `request.getMethod()`
 - `request.getRequestURI()`
 - `request.getProtocol()`

Example

14

- To see how these methods can be used look at the example (Making a Table of All Request Headers) in section 5.2, chapter 5.



The screenshot shows a Microsoft Internet Explorer window titled "Servlet Example: Showing Request Headers". The address bar displays "http://localhost:8080/servlet/coreservlets.ShowRequestHeaders". The page content includes the title "Servlet Example: Showing Request Headers" and the following information:

Request Method: GET
Request URI: /servlet/coreservlets.ShowRequestHeaders
Request Protocol: HTTP/1.1

Header Name	Header Value
accept	image/gif, image/x-bitmap, image/jpeg, image/pipe, application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*
accept-language	en-us
accept-encoding	gzip, deflate
user-agent	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
host	localhost
connection	Keep-Alive

The browser's status bar at the bottom shows "Done" and "Local intranet".

Checking For Missing Headers

15

- HTTP 1.0
 - ▣ *All request headers are optional*
- HTTP 1.1
 - ▣ Only Host is required
- Conclusion
 - ▣ Always check that request.getHeader is non-null before trying to use it

```
String val = request.getHeader("Some-Name");  
if (val != null)  
{  
    ...  
}
```

Some Common HTTP 1.1 Request Header (Accept)

16

- Indicates MIME types browser can handle.
- image/gif, application/msword, image/jpeg, etc.
- E.g. Can send different content to different clients. For example, PNG files have good compression characteristics but are not widely supported in browsers. A servlet could check to see if PNG is supported, sending:
 - if it is supported, and
 - if not

Some Common HTTP 1.1 Request Headers (Cookie)

17

- This header returns cookies to servers that previously sent them to the browser.
- Never read this header directly because doing so would require cumbersome low-level parsing.
 - ▣ Use `request.getCookies` instead.

Some Common HTTP 1.1 Request Headers

(Accept-Language)

18

- This header specifies the client's preferred languages.
 - ▣ Produce results in more than one language.
- E.g. en, en-us, da, etc.

Some Common HTTP 1.1 Request Headers

(Host)

19

- ▣ Indicates host given in original URL .
- ▣ This is a required header in HTTP 1.1.
- ▣ E.g. www.somebookstore.com.
- ▣ E.g. localhost.

Some Common HTTP 1.1 Request Headers (Referer)

20

- ▣ URL of referring Web page.
- ▣ Useful for tracking traffic.
- ▣ Can also be used to let users set preferences and then return to the page they came from.
- ▣ Tracking advertisers who refer people to your site.
- ▣ Can be easily spoofed.

Some Common HTTP 1.1 Request Headers (User-Agent)

21

- This header identifies the browser or other client.
- Best used for identifying *category* of client.

References

22

- Material for this lecture was obtained from:
 1. The core web programming, by Marty Hall and Larry Brown, A Sun Microsystems Press/Prentice Hall PTR Book.
 2. "Internet & World Wide Web How to Program"; Deitel, Deitel & Nieto; Prentice Hall, 3rd edition.
 3. The slides of Jack G. Zheng (www.java-web.net/cis3270fall06/lecture/Servlet1.ppt).