Advanced Java

Lesson 3—Java Servlet II

Learning Objectives

- ✓ Discuss Session Management
- ✓ Explain Listeners in Java EE
- ✓ Describe Filters in Java EE

Advanced Java

Topic 1—Session Management

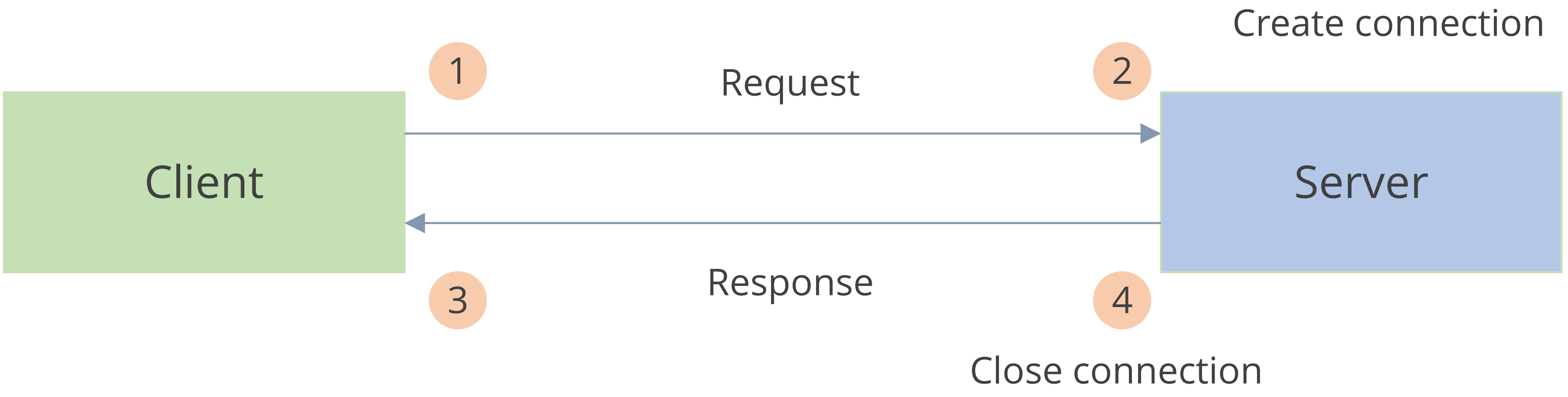
- Stateless Web Application
- Stateful Web Application
- Session Management and its types

Stateless Web Application

We have learned that web applications are stateless (by default) if given protocol is HTTP (stateless protocol).

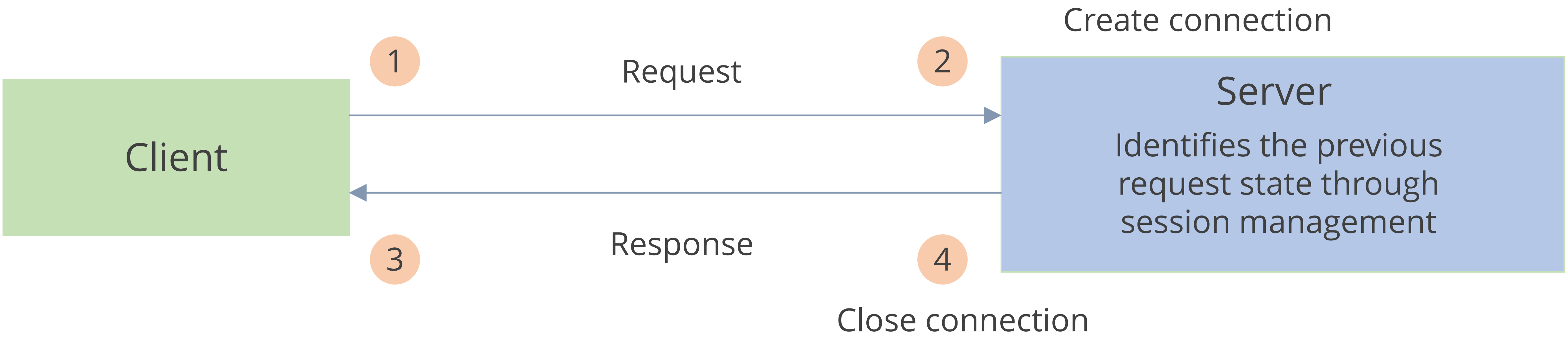
It does not allow or provide data access to previous request during the processing of current request in the Servlet or JSP.

In stateless web app, there is no way to preserve client data across multiple requests:



Stateful Web Application

- A web application that can remember or use the data of previous request during the processing of current request is a called stateful web application.
- In servlet API, there are different ways of making web application stateful so that it remembers client data across request from same browser.
- Making web application stateful is known as session tracking or session management.


Create connection
1
Request
2
Server
Identifies the previous request state through session management
Client
3
Response
4
Close connection

Introduction to Session Management

Session Management is a mechanism used by the **web container** to store **session** information for a particular user.

Session is a conversional state between client and server, and it can consist of multiple requests and responses between client and server. Sessions are used for maintaining user specific state, including persistent objects.

User Session: It refers to a series of user application interactions that are tracked by the server.

Types of Session Management

1. Hidden Form Field
2. URL Writing
3. Cookies
4. HttpSession

Hidden Form Field

- Hidden Form Field
- URL Writing
- Cookie
- HttpSession

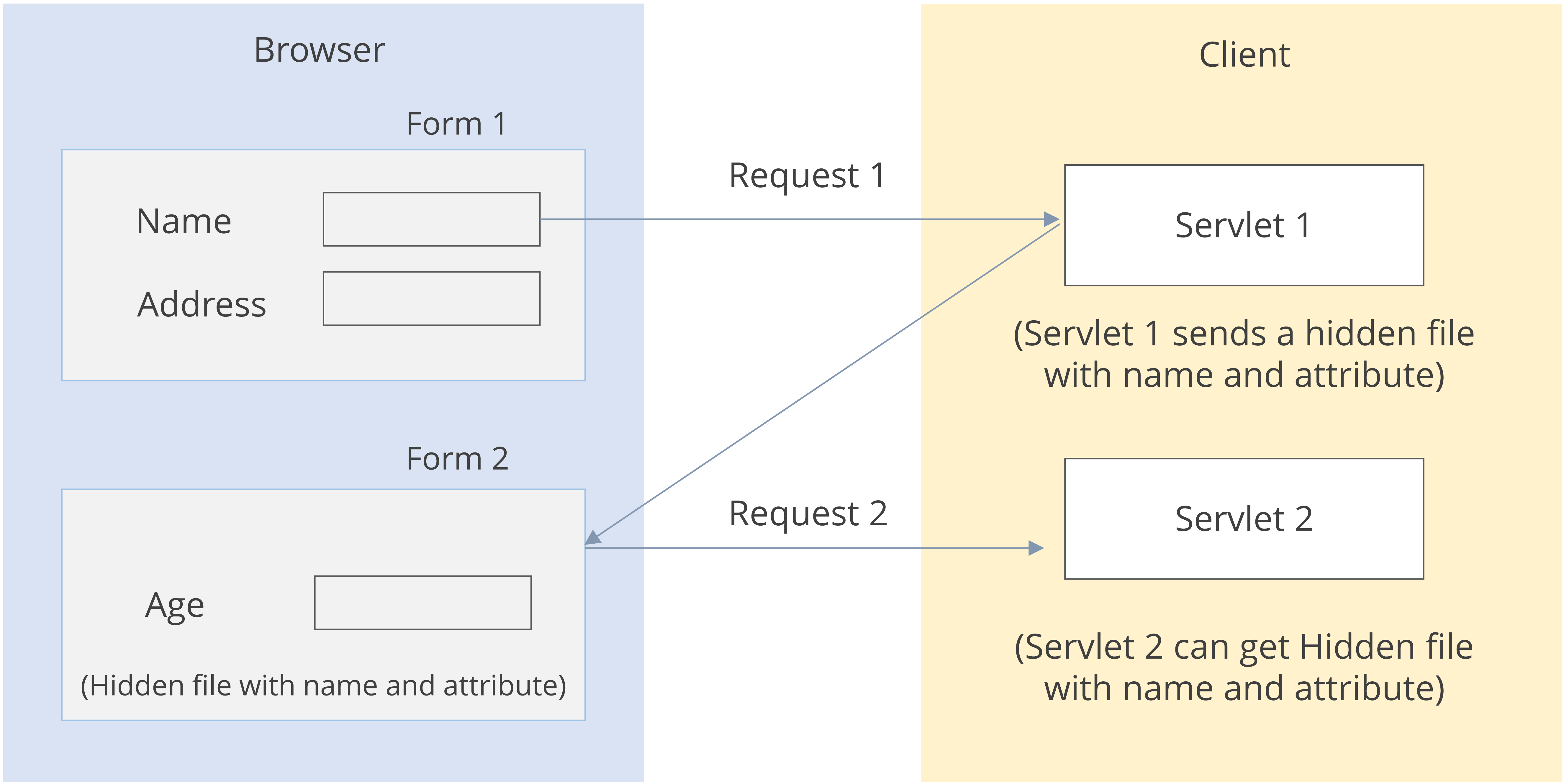
- When Server sends response for one particular request, it sends one hidden file attached to it
- Client gets response with one Hidden Field
- In Hidden Field name and value are set
- If the same page sends request the next time, server can recognize it by getting the parameter of Hidden Field

```
<input type =”hidden field” name=”somename” value=”somename”/>
```

One of the limitations of a Hidden Form Field is that it requires extra form submission on each page.

Using Hidden Form Field to Manage Session

- Hidden Form Field
- URL Writing
- Cookie
- HttpSession



Browser

Form 1

Name

Address

Form 2

Age

(Hidden file with name and attribute)

Request 1

Request 2

Client

Servlet 1

(Servlet 1 sends a hidden file with name and attribute)

Servlet 2

(Servlet 2 can get Hidden file with name and attribute)

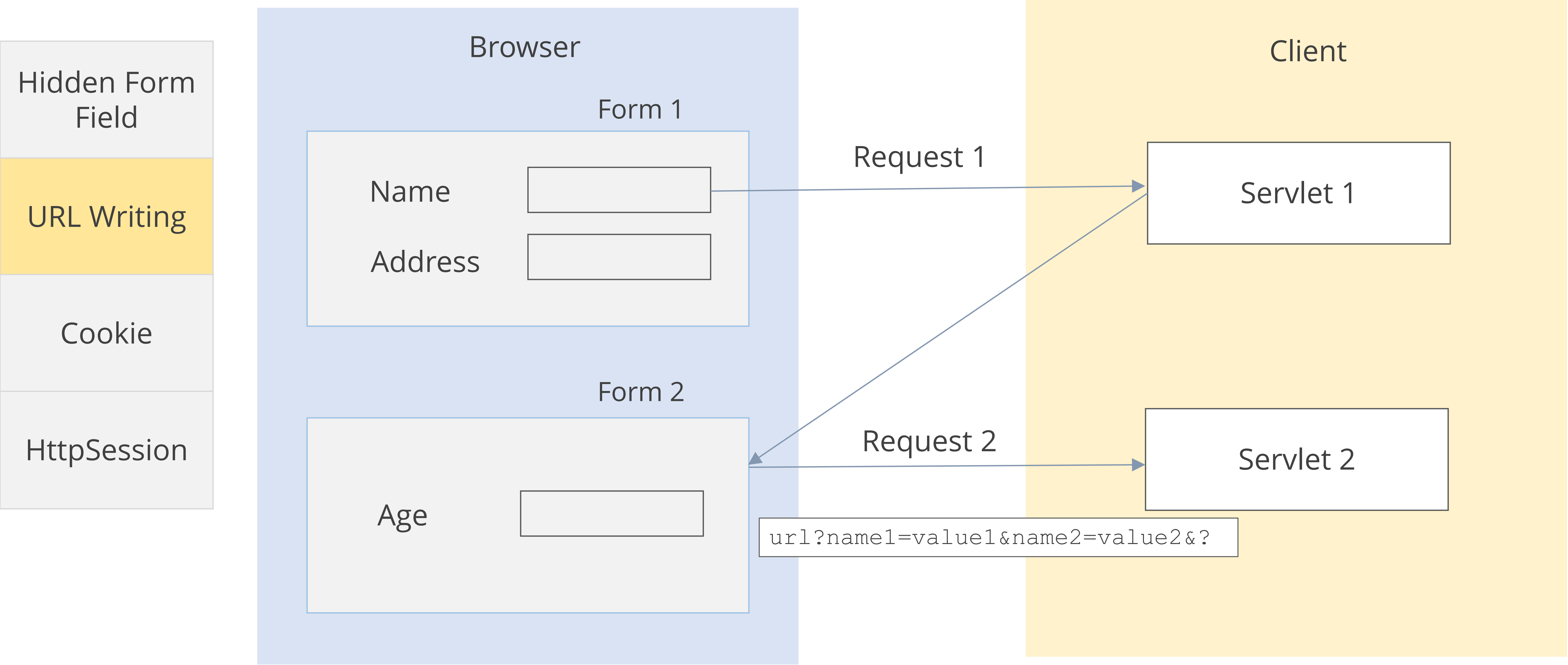
URL Writing

- Hidden Form Field
- URL Writing
- Cookie
- HttpSession

- In URL writing, one token is appended to URL itself
- This token contains one name and one value
- With the help of hyperlink, token’s name and values are passed to server
- A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&)
- Code: `url?name1=value1&name2=value2&?`

URL writing works with hyperlinks and sends text information.

URL Writing to Exchange Session Information



Cookie

Hidden Form Field

Url Writing

Cookie

HttpSession

- Servlet API provides javax.servlet.http.Cookie class to maintain session
- This class extends Object class and implements Cloneable interface
- While creating a cookie, a small amount of information is sent by a Servlet to a web browser, saved by the browser, and later sent back to the server
- Cookie has a name, value, and optional attribute
- The browser returns cookies to the servlet by adding fields to HTTP request headers

Attributes of a Cookie

Hidden Form Field

Url Writing

Cookie

HttpSession

- Name: Name of the cookie
- Value: Value of the cookie
- Comment: Text explaining purpose of cookie
- Max-Age: Time in seconds after which the client should not send cookie back to server
- Domain: Domain to which the cookie should be sent
- Path: The path to which the cookie should be sent
- Secure: Specifies if cookie should be sent via https

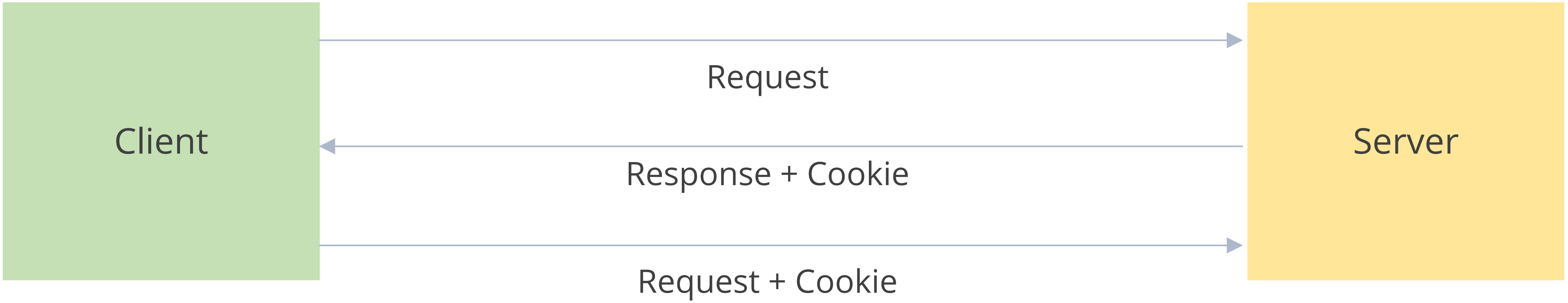
Cookies to Exchange Session Information

Hidden Form Field

Url Writing

Cookie

HttpSession



Constructor and Methods of Cookie Class

Hidden Form Field

Url Writing

Cookie

HttpSession

- Constructs a cookie with a specified name and value.

```
public Cookie(java.lang.String name,
              java.lang.String value)
```

- Cookie class has the following methods:
    - getName(): To get the Name of Cookie
    - getValue(): To get the value of Cookie
    - setMaxAge(): To set maximum age of Cookie
    - setValue(): To set value to Cookie

Creating Cookie Object

Hidden Form Field

Url Writing

Cookie

HttpSession

Code to create Cookie object:

```
Cookie   cookieObj = new Cookie(“name”,”value”);
```

Code to add Cookie to browser response:

```
response.addCookie(cookieObj);
```

Code to retrieve cookie object:

```
Cookie cookies[]=request.getCoookies();

for(Cookie cookie : cookies)
{
out.print("name = "+ cookie.getName());
out.print("value="+cookie.getValue());
}
```

Cookie: Limitations

Hidden Form Field

Url Writing

Cookie

HttpSession

- All data for a session are kept on the client. Corruption, expiration, or purging of cookie files can result in incomplete, inconsistent, or missing information.
- Cookies may not be available for many reasons: the user may have disabled them, the browser version may not support them, the browser may be behind a firewall that filters cookies, and so on.

Session Object APIs

Hidden Form Field

Url Writing

Cookie

HttpSession

- javax.servlet.http.HttpSession interface provides a way to identify one user across more than one page
- Servlet Container uses this interface to create a session between Http Client and Http Server
- Servlet can view and manipulate the information about servlet
- javax.servlet.http.HttpSession has method named as getSession
- getSession method gets the current valid session associated with this request if create is false or, if necessary, it creates a new session for the request

HttpSession Methods

Hidden Form Field

Url Writing

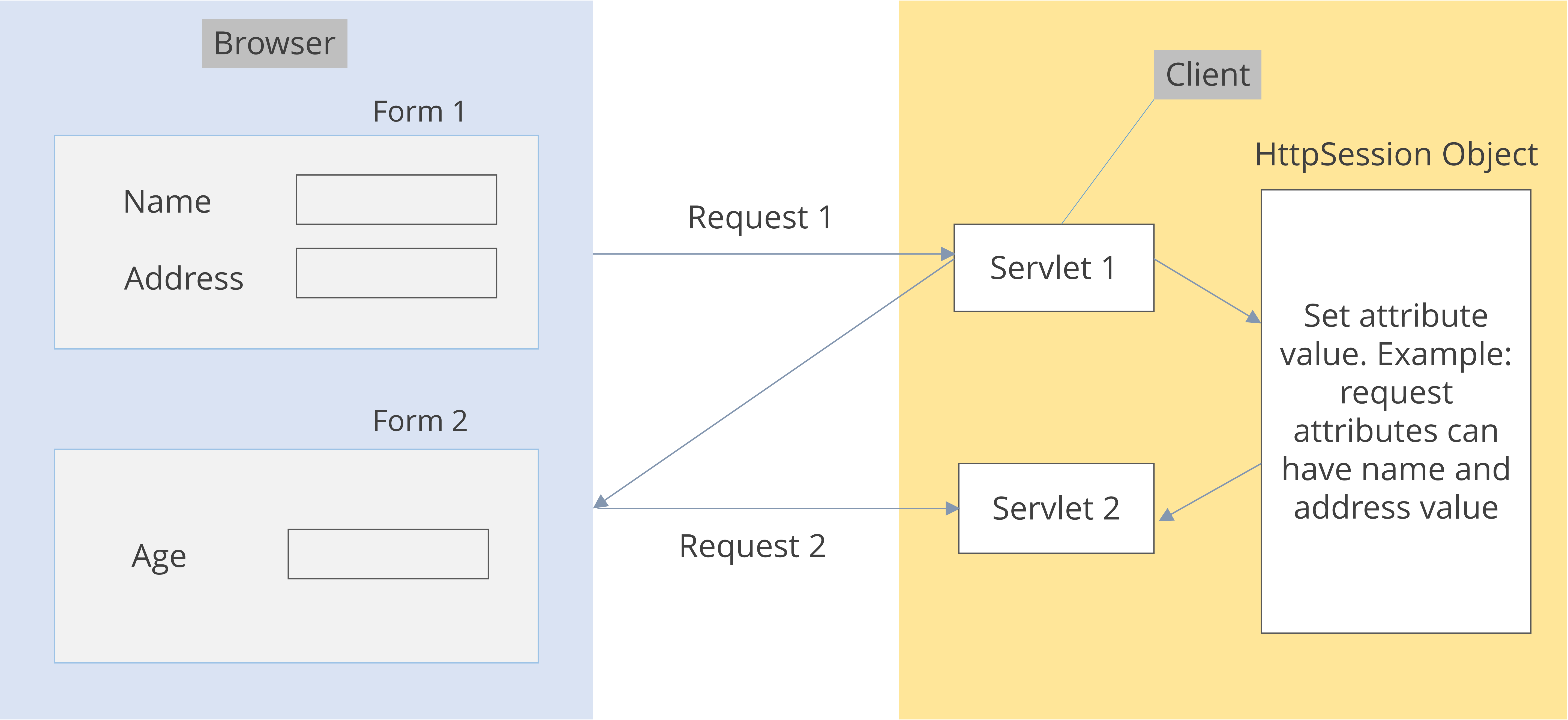
Cookie

HttpSession

int getMaxInactiveInterval()	Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.
String getId()	Returns a string containing the unique identifier assigned to this session.
void setMaxInactiveInterval(int interval)	Specifies the time, in seconds, between client requests before the servlet container will invalidate this session.
Void invalidate()	Invalidates this session and then unbinds any objects bound to it.
HttpSession getSession()	Returns the current session associated with this request, or if the request does not have a session, it creates one.
HttpSession getSession(boolean create)	Returns the current HttpSession associated with this request or, if there is no current session and create is true, it returns a new session.

HttpSession for Session Tracking

- Hidden Form Field
- Url Writing
- Cookie
- HttpSession



Advanced Java

DEMO—Session Management in Servlet

Advanced Java

Topic 2—Listeners in Java EE

- Events in Servlet API
- Events Class Servlet API
- Event Listener Interface

Events in Servlet API

Change in the state of an object is known as an event, which occurs during the lifecycle of an object.

There are important events related to a web application and Servlets which are handled by event listener.

To be notified of events, a custom class, that implements the correct listener interface~~,~~ needs to be coded and the listener class needs to be deployed via web.xml.

Event Class Servlet APIs

1. ServletRequestEvent
2. ServletContextEvent
3. ServletRequestAttributeEvent
4. ServletContextRequestArrtibuteEvent
5. HttpSessionEvent
6. HttpSessionEventBindingEvent

Event Listener Interface

Listeners are objects that notify whenever any event occurs. They contain event processing logics.

Listener Interfaces in servlet APIs are as follows:

1. ServletRequestListener
2. ServletContextListener
3. ServletRequestAttributeListener
4. ServletContextAttributeListener
5. HttpSessionListener
6. HttpSessionAttributeListener
7. HttpSesionBindingListener
8. HttpSessionActivationListener

- ServletRequestListener
- ServletContextListener
- ServletRequestAttributeListener
- ServletContextAttributetListener
- HttpSessionListener
- HttpSessionAttributeListener
- HttpSesionBindingListener
- HttpSessionActivationListener

Event Listener Interface

ServletRequestListener

- It handles servlet request event
- It notifies when servlet container is initialized and destroys request object
- Servlet container generates servlet request event when it creates and destroys servlet.request object

Methods:

```
void requestInitialized(ServletRequestEvent e)
// when Servlet creates request object

void requestDestroyed(ServletRequestEvent e)
// When Servlet destroy request object
```

- ServletRequestListener
- **ServletContextListener**
- ServletRequestAttributeListener
- ServletContextAttributeListener
- HttpSessionListener
- HttpSessionAttributeListener
- HttpSesionBindingListener
- HttpSessionActivationListener

Event Listener Interface

ServletContextListener

- It handles and processes ServletContextEvent.
- It receives notifications when servlet container creates or destroys context object.

Methods:

```
void contextInitialized()

void contextDestroyed()
```

ServletRequestListener

ServletContextListener

ServletRequestAttributeListener

ServletContextAttributeListener

HttpSessionListener

HttpSessionAttributeListener

HttpSesionBindingListener

HttpSessionActivationListener

Event Listener Interface

ServletRequestAttributeListener

- It receives notification of ServletRequestAttribute event.
- It is used to find out when an attribute has been added, removed, or replaced from request object.

Methods:

`void attributeAdded(`ServletRequestAttributeEvent srae`) :` Receives notification that an attribute has been added to the ServletRequest

`void attributeRemoved(`ServletRequestAttributeEvent srae : Receives notification that an attribute has been removed from the ServletRequest

`void attributeReplaced(`ServletRequestAttributeEvent srae`) :` Receives notification that an attribute has been replaced on the ServletRequest.

- ServletRequestListener
- ServletContextListener
- ServletRequestAttributeListener
- ServletContextAttributeListener
- HttpSessionListener
- HttpSessionAttributeListener
- HttpSesionBindingListener
- HttpSessionActivationListener

Event Listener Interface

ServletContextAttributeListener

It receives and processes ServletContextAtrribute event.

Methods:

`void attributeAdded(`ServletContextAttributeEvent event`) :`

Receives notification that an attribute has been added to the ServletContext.

`void attributeRemoved(`ServletContextAttributeEvent event`) :`

Receives notification that an attribute has been removed from the ServletContext.

`void attributeReplaced(`ServletContextAttributeEvent event`) :`

Receives notification that an attribute has been replaced from the ServletContext.

- ServletRequestListener
- ServletContextListener
- ServletRequestAttributeListener
- ServletContextAttributeListener
- HttpSessionListener
- HttpSessionAttributeListener
- HttpSesionBindingListener
- HttpSessionActivationListener

Event Listener Interface

HttpSessionListener

It receives and processes http session events. This listener is used to find out total active users.

Methods:

`void sessionCreated(HttpSessionEvent se) :` Receives notification that a session has been created.

`void sessionDestroyed(HttpSessionEvent se) :` Receives notification that a session is about to be invalidated.

- ServletRequestListener
- ServletContextListener
- ServletRequestAttributeListener
- ServletContextAttributeListener
- HttpSessionListener
- HttpSessionAttributeListener
- HttpSesionBindingListener
- HttpSessionActivationListener

Event Listener Interface

HttpSessionAttributeListener

- It handles and processes http session binding event.
- It is used to find out when an attribute has been added or replaced from session.

Methods:

`void attributeAdded(`HttpSessionBindingEvent event`):` Receives notification that an attribute has been added to a session

`void attributeRemoved(`HttpSessionBindingEvent event`):` Receives notification that an attribute has been removed from a session

`void attributeReplaced(`HttpSessionBindingEvent event`):` Receives notification that an attribute has been replaced in a session

- ServletRequestListener
- ServletContextListener
- ServletRequestAttributeListener
- ServletContextAttributeListener
- HttpSessionListener
- HttpSessionAttributeListener
- HttpSesionBindingListener
- HttpSessionActivationListener

Event Listener Interface

HttpSesionBindingListener

- It handles and processes http session binding event.
- It notifies when the object of the class has been added or removed from the session.

Methods:

`void valueBound(`HttpSessionBindingEvent event`) :` Notifies the object that it is being bound to a session and identifies the session.

`void valueUnbound(`HttpSessionBindingEvent event`) :` Notifies the object that it is being unbound from a session and identifies the session.

- ServletRequestListener
- ServletContextListener
- ServletRequestAttributeListener
- ServletContextAttributeListener
- HttpSessionListener
- HttpSessionAttributeListener
- HttpSesionBindingListener
- HttpSessionActivationListener

Event Listener Interface

HttpSessionActivationListener

- It receives notification of http session event.
- It notifies when the session migrates from one JVM(Java Virtual Machine) to another.

Methods:

void **sessionDidActivate**(HttpSessionEvent se) : Notifies that the session has just been activated

void **sessionWillPassivate**(HttpSessionEvent se): Notifies that the session is about to be passivated

Application events provide notifications of a change in state of the servlet context (each Web Application uses its own servlet context) or of an HttpSession object.

Servlet Listener Configuration

@WebListener annotation is used to declare a class as Listener. However, the class should implement one or more of the Listener interfaces.

To define listener in web.xml:

```
<listener>
    <listener-class>
    packagename.ClassNameOfImplementedListener
    </listener-class>
</listener>
```

Advanced Java

DEMO—Implementing Listener Interface to Handle Events

Advanced Java

Topic 3—Filters in Java EE

- Why Filters?
- What is a Filter?
- Filter in Servlets

Why Filters?

Filters are used when there is a need:

- For an operation to occur every time a particular request is made
- To perform one operation on other request in the web application
- To allow one operation to be turned off at deployment

The filter API interface can be found in javax.servlet package

What is a Filter?

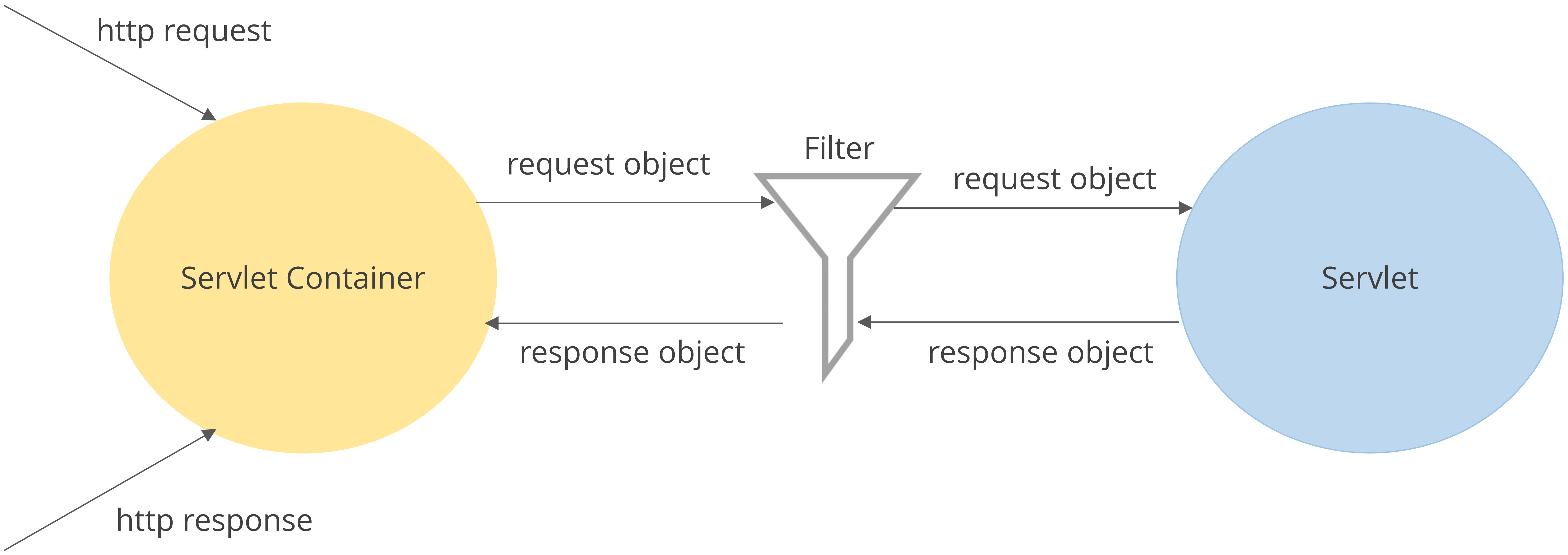
- A filer is an interface used to filter tasks on the request to a resource (a servlet or static content), on the response from a resource, or both

- It has 3 methods (lifecycle)

```
public void init(FilterConfig config);
public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain);
public void destroy();
```

- Filters are configured in deployment descriptor of a web application

Filter in Servlets



Filter Types

1. Request Filter: Contains only pre-request processing logic.

2. Response Filter: Contains post-response generation logic.

3. Request-Response Filter: Contains both pre-request processing and post-response generation logic.

Developing and Mapping a Filter Class

Declaring a Filter in the web.xml

```
<filter>
  <filter-name> onelogicalname </filter-name>
  <filter-class> filterclassname </filter-class>
  </filter>
```

Mapping a Filter in the web.xml

```
<filter-mapping>
  <filter-name> onelogicalname </filter-name>
  <furl-pattern> /urlpatter </url-pattern>
  </filter-mapping>
```

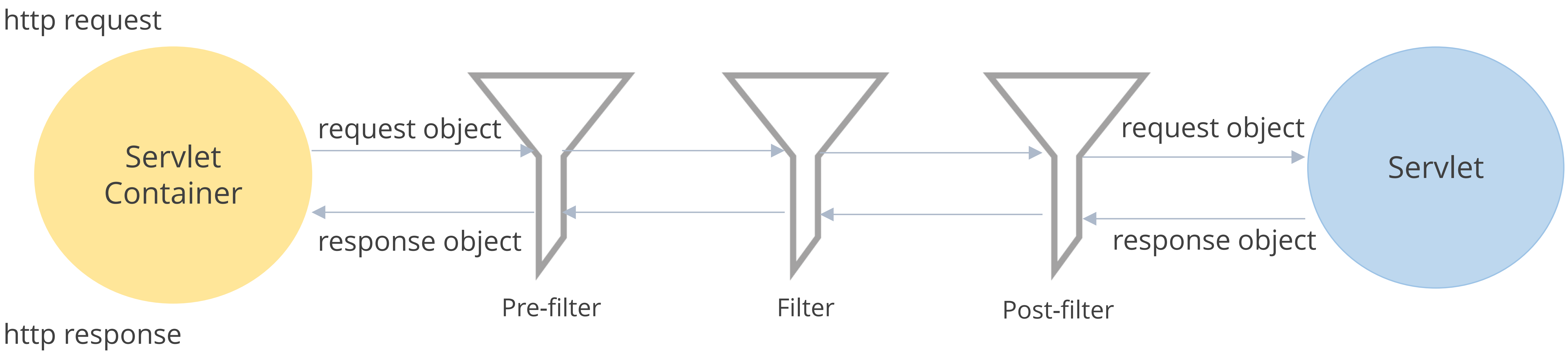
Writing a Filter Program

To write a filter program:

- Create a Java class to implement Filter interface that has the following three methods:
    - init() method – It is called once when the container instantiates a filter
    - doFilter() method – It is called for every request intercepted by the filter
    - destroy() method - Before the web container removes a filter instance from service, the destroy method is called
- Create one web.xml file to configure the filter

Filter Chain

If multiple filters are configured for servlet, they are executed in the same order in which they are defined in web.xml



Key Takeaways

- ✓ A web application that can remember or use the data of previous request during the processing of current request is a called stateful web application.
- ✓ Session Management is a mechanism used by the Web container to store session information for a particular user.
- ✓ Hidden Form Field, Url Writing, Cookies, and HttpSession are types of session management.
- ✓ To be notified of events, a custom class, that implements the correct listener interface, needs to be coded and the listener class needs to be deployed via web.xml.

Quiz

Thank You