Page Navigation

Sandeep, SunBeam

Introduction

- Process of taking user from one page to another is called page navigation.

- Techniques for navigating user across multiple web pages:
    1. Client pull.
    2. Server pull.

Client Pull Technique

- Navigating client to the next page in the new request.
- It can be done using two ways:
    1. By clicking on button/hyper-link etc.(Here client(person) is involved)
    2. By calling "resp.sendRedirect()" method i.e using redirect scenario.(Here client(browser) is involved)

- Request scope attribute will not be available to the next page, if navigated via redirect scenario i.e. minimum scope required here is session scope.

Server Pull Technique

- Navigating client to the next page in same request.

- It can be done using resource chaining i.e. RequestDispatcher scenario.

    1. Neither human client nor client browser is involved. Web container will chain the resources dynamically using include or forward scenario.

- Request scope attribute will be available to the next page, if navigated via RequestDispatcher scenario.

Redirect scenario

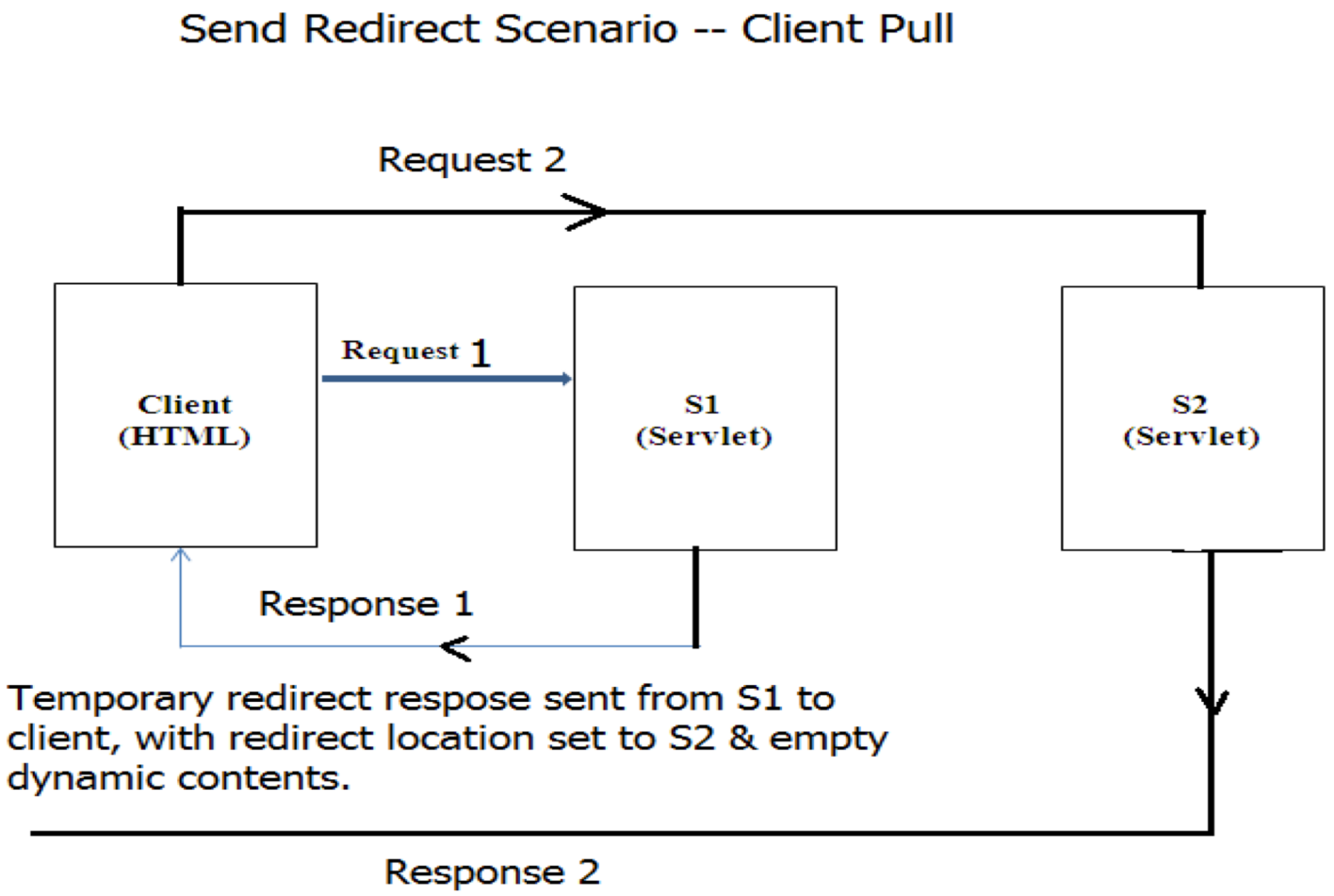
- It is a client pull technique. Client browser is responsible for pulling the resources.

- API for redirection: **HttpServletResponse**
    - **Method : void sendRedirect(String location)throws IOException**

- It sends a temporary redirect response to the client using the specified redirect location URL and clears the buffer.

- If the response has already been committed, this method throws an IllegalStateException.

Redirect scenario



Request Dispatcher Scenario

- Neither human client nor client browser is involved.
- Web container will chain the resources dynamically using include or forward scenario.
- It is faster than redirection as round trip delay avoided.
- We can chain the resources belonging to the same web application.

Request Dispatcher Scenario

- It is interface declared javax.servlet package.
- The SC creates the RD object, which is used as a wrapper around a server resource located at a particular path.
- It is intended to wrap servlets, but a SC can create RD objects to wrap any type of resource.
- There are 2 methods declared in RD interface:
    1. void forward(req,resp)throws SE,IOE
    2. void include(req,resp)throws SE,IOE

How to get the reference of instance of RequestDispatcher?

- Using ServletContext

```
ServletContext context = this.getServletContext();
RequestDispatcher rd = null;
rd = context.getRequestDispatcher("/url");
```

  - **The url must begin with a / and is interpreted as relative to the current context root.**
  - **IllegalArgumentException : If path does not start with a "/" character**

- Using ServletRequest

```
RequestDispatcher rd = null;
rd = request.getRequestDispatcher("url");
```

  - **Here url can be relative/absolute.**

Request Dispatcher Forward Scenario

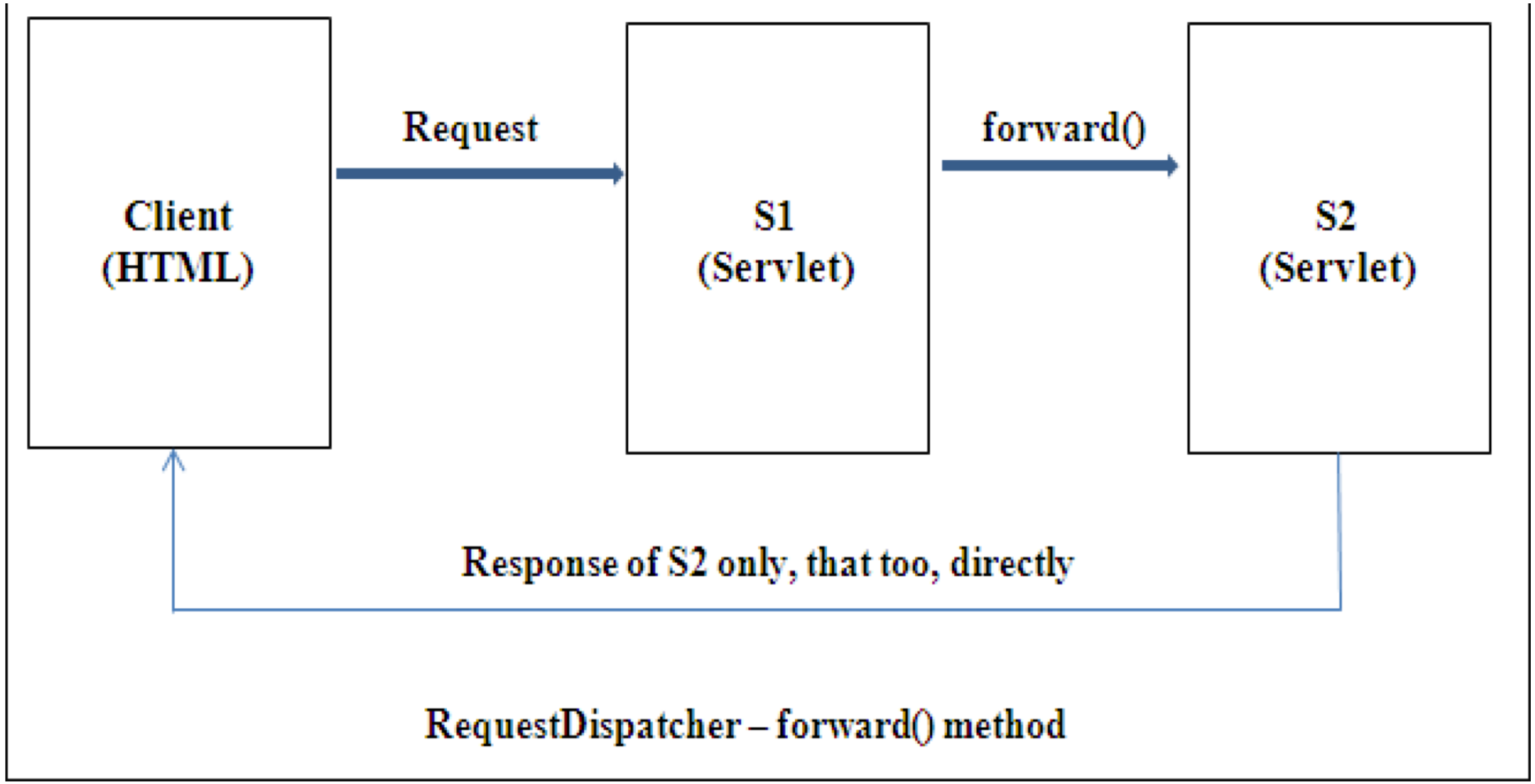
- forward() is a method of RequestDispatcher I/F.
  - **void forward(ServletRequest req,ServletResponse res) throws SE, IOE**
- Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
- It allows one servlet to do preliminary processing of a request and another resource to generate the response.
- forward(req,resp) method should be called before the response has been committed to the client, otherwise it may throws IllegalStateException.
- Consider following example:

```
RequestDispatcher rd =  null;

rd = request.getRequestDispatcher("second");

rd.forward(request, response);
```

Request Dispatcher Forward Scenario



Request Dispatcher Include Scenario

- include() is a method of RequestDispatcher I/F.
  - **void include(ServletRequest req,ServletResponse res) throws SE, IOE**

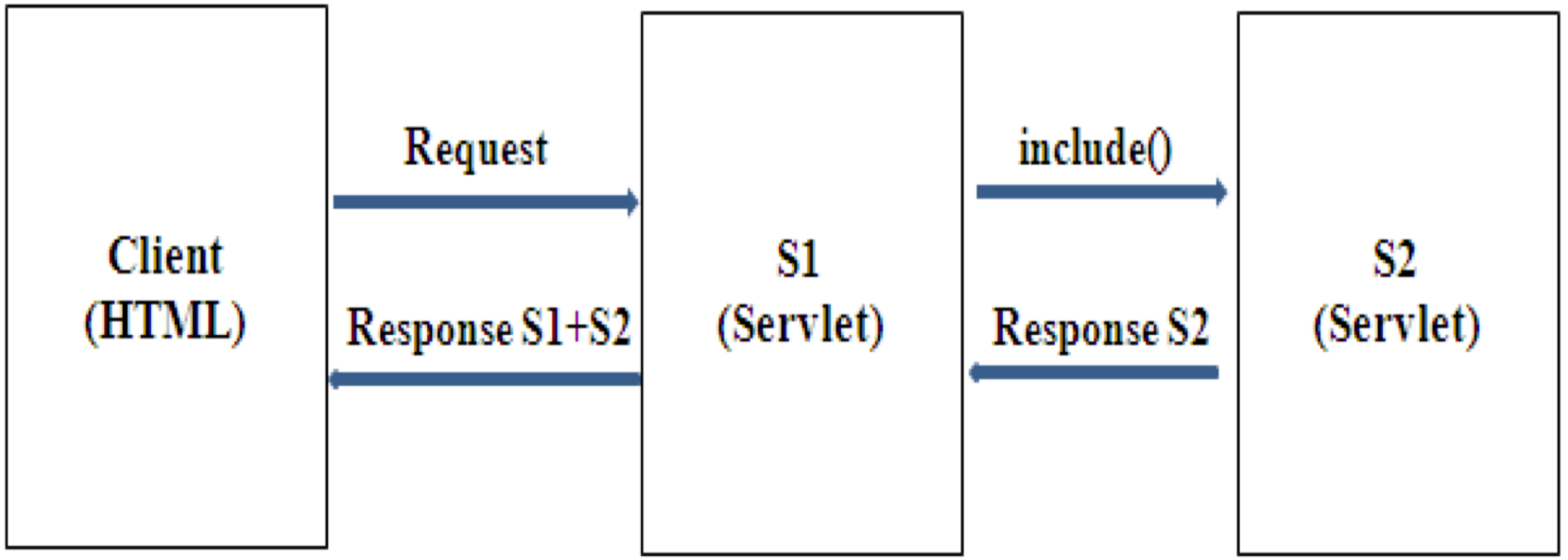
- It includes the content of a resource (servlet, JSP page, HTML file) in the response..

- Included page generates the response.

- Consider following example:

```
RequestDispatcher rd =  null;
rd = request.getRequestDispatcher("second");
rd.include(request, response);
```

Request Dispatcher Include Scenario



RequestDispatcher – include() method

Difference between forward and include?

- In forward scenario, control is forward type only i.e. control do not return to the earlier page.
- In include scenario, included page generates dynamic response & finishes the execution and control gets returned to the original page.

- In forward scenario, Only last page in the chain can generate and commit dynamic response to the client.
- In include scenario, included page as well as original page generates dynamic response.

What is the difference between redirect and RD scenario?

1. **Scope:**
    - Redirect requires minimum session scope.
    - RD requires minimum request scope.
2. **Attribute:**
    - In redirect scenario attributes are handled using HttpSession object.
    - In RD scenario attributes are handled using HttpServletRequest object.
3. **Speed:**
    - To reach to the resource redirect require extra round trip hence it is slower.
    - To reach to the resource RD do not require extra round trip hence it is faster.
4. **URL:**
    - In redirect, the URL seen by the client is of the redirected page.
    - In RD, the URL seen by the client is of the first page only.

Exception Handling

- An HTTP error code or an exception thrown by a serlvet can be mapped to a resource bundled with the application to customize the appearance of content when a servlet generates an error.
- This is done using error pages. These pages should be configured in web.xml.
- For HTTP error code, we can do mapping as follows:

```
<error-page>
        <error-code>404</error-code>
        <location>/Error-404.jsp</location>
</error-page>
```

- For exception, we can do mapping as follows:

```
<error-page>
        <exception-type> javax.servlet.ServletException </exception-type>
<error-page>
```

Auto-Refresh/Wait Pages

- Another response header technique that is uncommon but helpful is to send a wait page or a page that will auto-refresh to a new page after a given period of time.
- This tactic is helpful in any case where a response might take an uncontrollable time to generate response.
- The entire mechanism revolves around setting the Refresh response header.
- The header can be set using the following:
  - response.setHeader("Refresh", "time; URL=url");
  - time" should be replaced with the amount of seconds
- For example:

```
response.setHeader("Refresh", "10; URL=http://127.0.0.1/foo.html");
```

Thank you