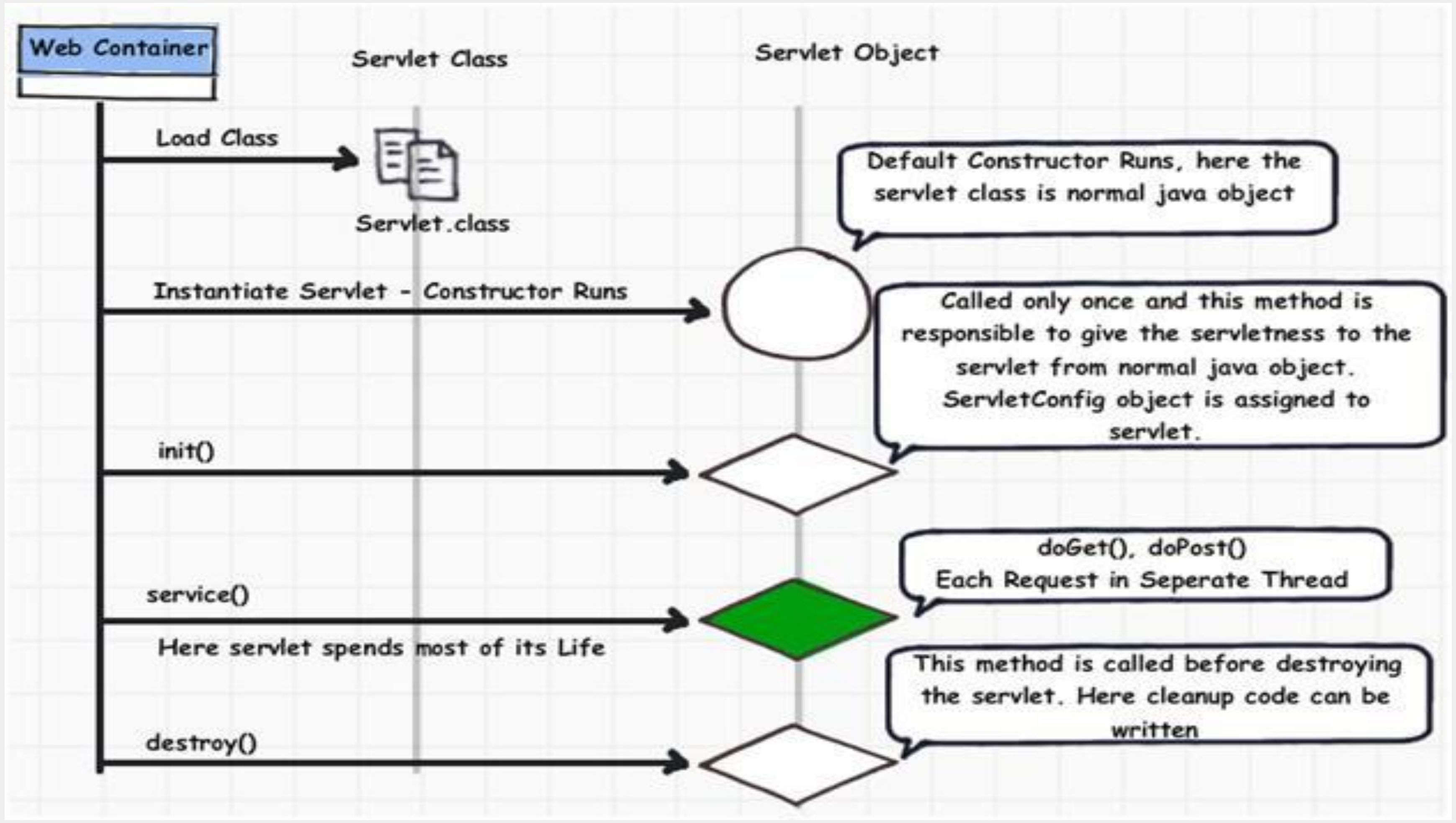
Servlet II

Life of a servlet and beyond

Agenda

- Servlet lifecycle
- Key servlet methods
- Servlet config
- Servlet context
- The request and response objects
- Workflow of a web application

Servlet lifecycle


Web Container
Servlet Class
Servlet Object
Load Class
Servlet.class
Default Constructor Runs, here the servlet class is normal java object
Instantiate Servlet - Constructor Runs
Called only once and this method is responsible to give the servletness to the servlet from normal java object. ServletConfig object is assigned to servlet.
init()
doGet(), doPost() Each Request in Seperate Thread
service()
Here servlet spends most of its Life
This method is called before destroying the servlet. Here cleanup code can be written
destroy()

init( )

- The first method to be called by the web container immediately after instantiating the servlet.
- Called only once.
- javax.servlet.ServletConfig object is passed as argument by the container.
- A convenient without parameter version is available for overriding.
- If you override the parameterized version, you must call super.init(config)
- Can throw javax.servlet.UnavailableException if initialization fails for some reason

Persistent

service( )

- The method that is invoked next in order and is responsible for calling the appropriate *doXXX()* method based on the HTTP method sent by the client.

- Called each time the servlet processes a request. Each request runs in a separate thread.

- Need not be overridden

- Receives two arguments from the container which are references to objects of javax.servlet.ServletRequest and javax.servlet.ServletResponse

destroy( )

- Last method to be invoked in the lifecycle by the container to indicate that the servlet is being taken out of service.

- Called only once when a servlet is unloaded either by the server administrator or by the server itself after a long period of inactivity.

- Can be overridden.

Servlet initialization parameters

- A servlet may be passed configuration values.

- These static values can be anything ; for example :
  - name of a log file that the servlet writes to
  - name(s) of classes that the servlet dynamically loads
  - name of a properties files it reads
  - path of a folder it uploads files to

    etc. etc. etc.

Configuring and retrieving servlet's init. params.

- Configuring in the web.xml

```
<servlet>
        <servlet-class>………….</servlet-class>
        <init-param>
                     <param-name>SOME_PARAM_NAME</param-name>
                     <param-value>Some value for the parameter</param-value>
        </init-param>
<servlet>
```

- getInitParameter(String)
  - Returns the value as string of any initialization parameter passed to a servlet.
  - In the servlet's init( ) or doXXX() method

getInitParameter("SOME_PARAM_NAME");

Where is this configuration stored ?

- getServletConfig( )
  - Returns a reference to an object of javax.servlet.ServetConfig
  - Inherited from javax.servlet.GenericServlet
  - Is a collection (map)
  - Methods in the servlet config object can also be used to retrieve a servlet’s initialization parameters

Context initialization parameters

- Application wide configuration information accessible from any servlet or jsp.

- These static values can be anything ; for example :
  - e-mail address of the webmaster
  - image file path of the company’s logo
  - database url’s

    etc. etc. etc.

Configuring, storage and retrieval of context init. params.

- Configuring in the web.xml

```
    <web-app …………….>
                ………………………………
                ………………………………
                <context-param>
                        <param-name>WEB_MASTER_MAIL</param-name>
                        <param-value>themaster@website.com</param-value>
                </context-param>
      </web-app>
```

- getServletContext( )
  - Returns reference to an object of javax.servlet.ServletContext
  - Inherited from javax.servlet.GenericServlet
  - There is only one servlet context per web-application
  - Somewhere in the servlet

```
getServletContext().getInitParameter(“WEB_MASTER_EMAIL”);
```

Attributes

- Dynamic piece of updatable, retrievable and removable information stored in a specific scope.

- Scope defines the place (i.e. object) where an attribute is stored.

- What kind of information ?
  - total number of users online
  - details of the currently logged in user
  - user specific information like items in a shopping cart
  - error messages to be displayed to the user
  - temporary data required by one or more servlets to process the same request

    etc. etc. etc.

Scopes

- Scope decides the lifetime and visibility of an attribute

  - Application
    - represented through a javax.servlet.ServletContext object

  - Session
    - represented through a javax.servlet.http.HttpSession object

  - Request
    - represented through a javax.servlet.http.HttpServletRequest object

Servlet context attributes

- setAttribute(String, Object)
  - **Example:**

    javax.servlet.ServletContext servletContext;

    servletContext = getServletContext();

    servletContext.setAttribute(“TOTAL_USERS_ONLINE”, totUsersOnln);
- Object getAttribute(String)
  - **Example:**

    javax.servlet.ServletContext servletContext;

    servletContext = getServletContext();

    servletContext.getAttribute(“TOTAL_USERS_ONLINE”);
- removeAttribute(String)
  - **Example:**

    javax.servlet.ServletContext servletContext;

    servletContext = getServletContext();

    servletContext.removeAttribute(“TOTAL_USERS_ONLINE”);

More servlet context methods

- getContextPath()
  - Returns the context path (root) of the web application
  - **Given** : http://localhost:8080/sample-web-app/some-servlet
  - **Result** : /sample-web-app

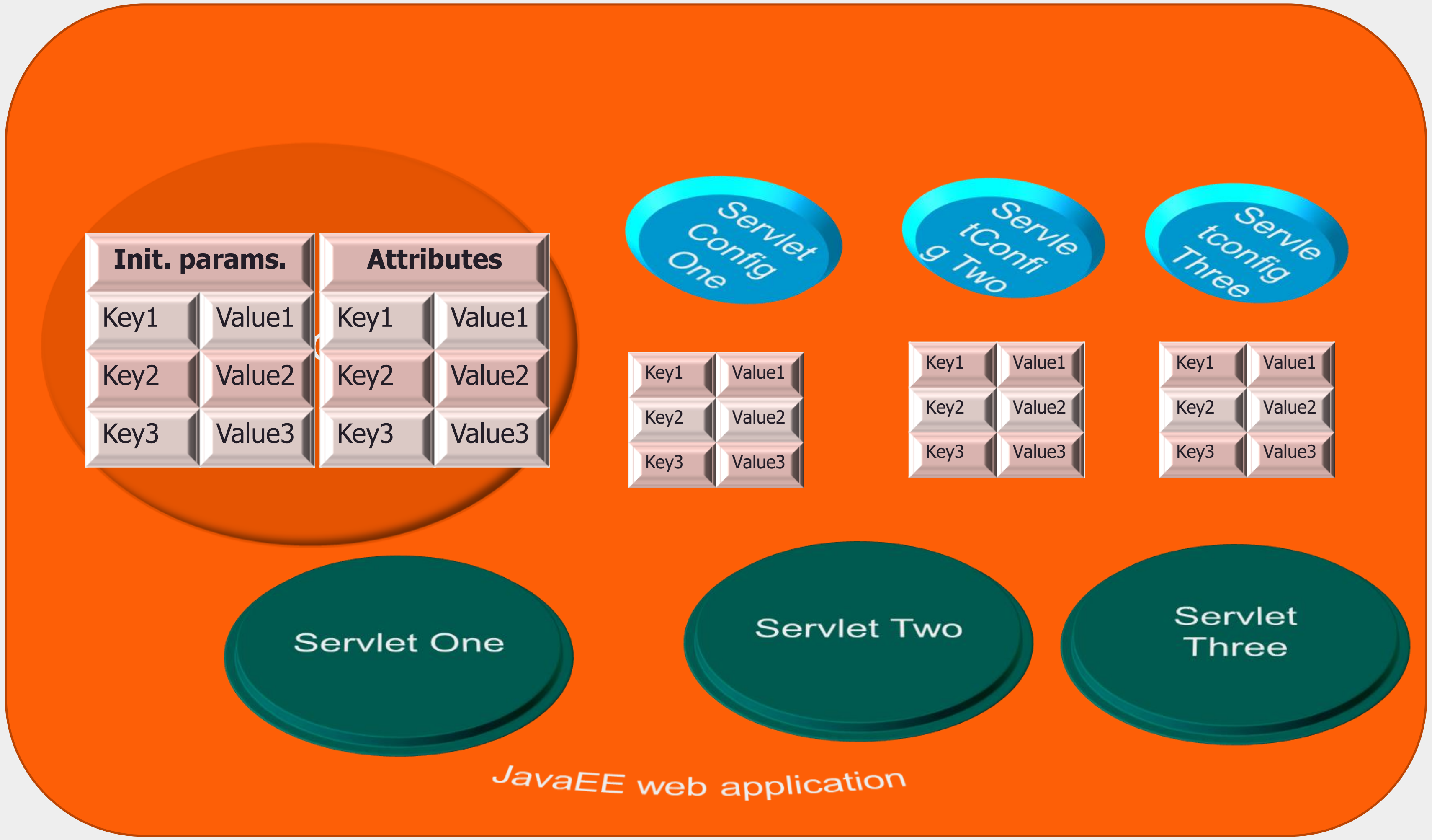
- getRealPath(String)
  - Returns the absolute file system path of a given resource as a string.
  - **Given:** A file named sample.jsp under the context path
  - **Example:** getRealPath("/sample.jsp");
  - **Result:** C:\Program Files\Apache Software Foundation\Tomcat 6.0\webapps\sample-web-app\sample.jsp

Key servlet methods (server logs)

- log(String)
  - Logs the specified message string to a log file.
  - **Example:** log(“Inside the doGet method”);

- log(String, Throwable)
  - Writes a message string and the stack trace of the specified exception object to a log file.
  - **Example:** log(“Exception in doPost”, exception);

Recap config, context and the web app.



The request and response objects

- Automatically created by the container each time a servlet receives a request and passed as argument to the service() method.

- For a http servlet, a request is an object of javax.servlet.http.HttpServletRequest and response is an object of javax.servlet.http.HttpServletResponse.

Request methods summary - I

- **getContextPath()**
  - Returns the context path of the current request in a string
  - **Given**   **:** http://localhost:8080/sample-web-app/sample-servlet
  - **Result**   **:** /sample-web-app
- **getDateHeader(String)**
  - Returns the date (time since “epoch”) in a long value for the specified request header
  - **Given**   **:** getDateHeader(“If-Modified-Since”);
  - **Result**   **:** 8062791888144911110
- **getHeader(String)**
  - Returns the value of the specified request header as a string
  - **Given**   **:** getHeader(“accept-language”);
  - **Result**   **:** en-us
- **getMethod( )**
  - Returns the HTTP method for the current request as a string
  - **Given**   **:** getMethod( )
  - **Result**   **:** POST

Request methods summary - II

- getPathInfo( )
  - Returns extra path I nformation which follows the servlet’s path
  - **Given** **:** In web.xml

```
<servlet-mapping>
        <servlet-name>something</servlet-name>
        <url-pattern>/my-app/my-servlet/*<url-pattern>
</servlet-mapping>
```

  - **Accessed like this:** http://localhost/my-app/my-servlet/one/two
  - **Result** **:** /one/two

- getQueryString( )
  - Returns the query string
  - **Given** **:** http://somehost/my-app/my-servlet?query=answer;key=value
  - **Result** **:** query=answer;key=value

- getRequestURI( )
  - Returns a part of the servlet’s request URL starting from the context root to the servlet’s mapping URL
  - **Given** **:** http://somehost/my-app/my-servlet
  - **Result** **:** /my-app/my-servlet

Request methods summary - III

- **getRequestURL( )**
  - Returns the complete URL used by the client for the servlet’s request
  - **Given** **:** http://somehost/my-app/my-servlet
  - **Result** **:** http://somehost/my-app/my-servlet
- **getServletPath( )**
  - Returns a part of the request URL that maps to the servlet as specified in the web.xml
  - **Given** : http://somehost/my-app/my-servlet
  - **Result** : /my-servlet
- **setAttribute(String, Object)**
  - Stores an attribute in the request.
  - **Example** : setAttribute(“com.webapp.LOGGED_USER”, user);
- **getAttribute(String)**
  - Retrieves the value of the named attribute as an Object.
  - **Example** : getAttribute(“com.webapp.LOGGED_USER”);
- **removeAttribute(String)**
  - Removes the named attribute from the request.
  - **Example** : removeAttribute(“com.webapp.LOGGGED_USER”);

Persistent

Request methods summary - IV

- **getParameter(String)**
  - Returns the value of a request parameter as a string. For http servlets parameters are contained in the query string or post form data.
  - **Example** : getParameter("username");
- **getParameterValues(String)**
  - Returns an array of strings containing all the values of a request parameter.
  - **Example** **:** getParameterValues("subjects");
- **getInputStream( )**
  - Returns reference to an object of javax.servlet.ServletInputStream which can be used to read the body of a request as a binary stream.
- **getReader( )**
  - Returns reference to an object of java.io.BufferedReader which can be used to read the body of a request as a character stream.

Response methods summary - I

- **getOutputStream( )**
  - Returns reference to an object of javax.servlet.ServletOutputStream which can be used to write bytes to a response.
- **getWriter( )**
  - Returns reference to an object of java.io.PrintWriter which can be used to write characters/strings to a response.
- **setContentType(String)**
  - Sets the content type of the response being sent to the client.
  - **Example :** setContentType(“text/html”);

    setContentType(“application/pdf”)
- **addDateHeader(String, long)**
  - Adds a header field to the response with the specified name and date value (time since epoch).
  - **Example :** addDateHeader(“Last-Modified”, 1283152439681714084L);

Response methods summary - II

- **addHeader(String, String)**
  - Adds a header field to the response with the given name and value.
  - **Example :** addHeader(“Cache-Control”, “no-cache”);
- **addIntHeader(String, int)**
  - Adds a header field to the response with the specified name and integer value
  - **Example :** addIntHeader(“Retry-After”, 120);
- **sendError(int)**
  - Sends one of the pre-defined error codes as status of the response
  - Example :sendError(HttpServletResponse.SC_FORBIDDEN);
- **setDateHeader(String, long)**
  - Sets the value of a header field in the response to a given date value (time since epoch)
  - **Example :** setDateHeader(“Expires”, 1283152439681714084L);

Response methods summary - III

- **setHeader(String, String)**
  - Sets a header field in the response to a given name and value.
  - **Example : ** setHeader(“Server”,”Apache Tomcat/6.0.29”);

- **setIntHeader(String, int)**
  - Sets the value of a header field in the response to a given integer value.
  - **Example : ** setIntHeader(“Expires”, 0);

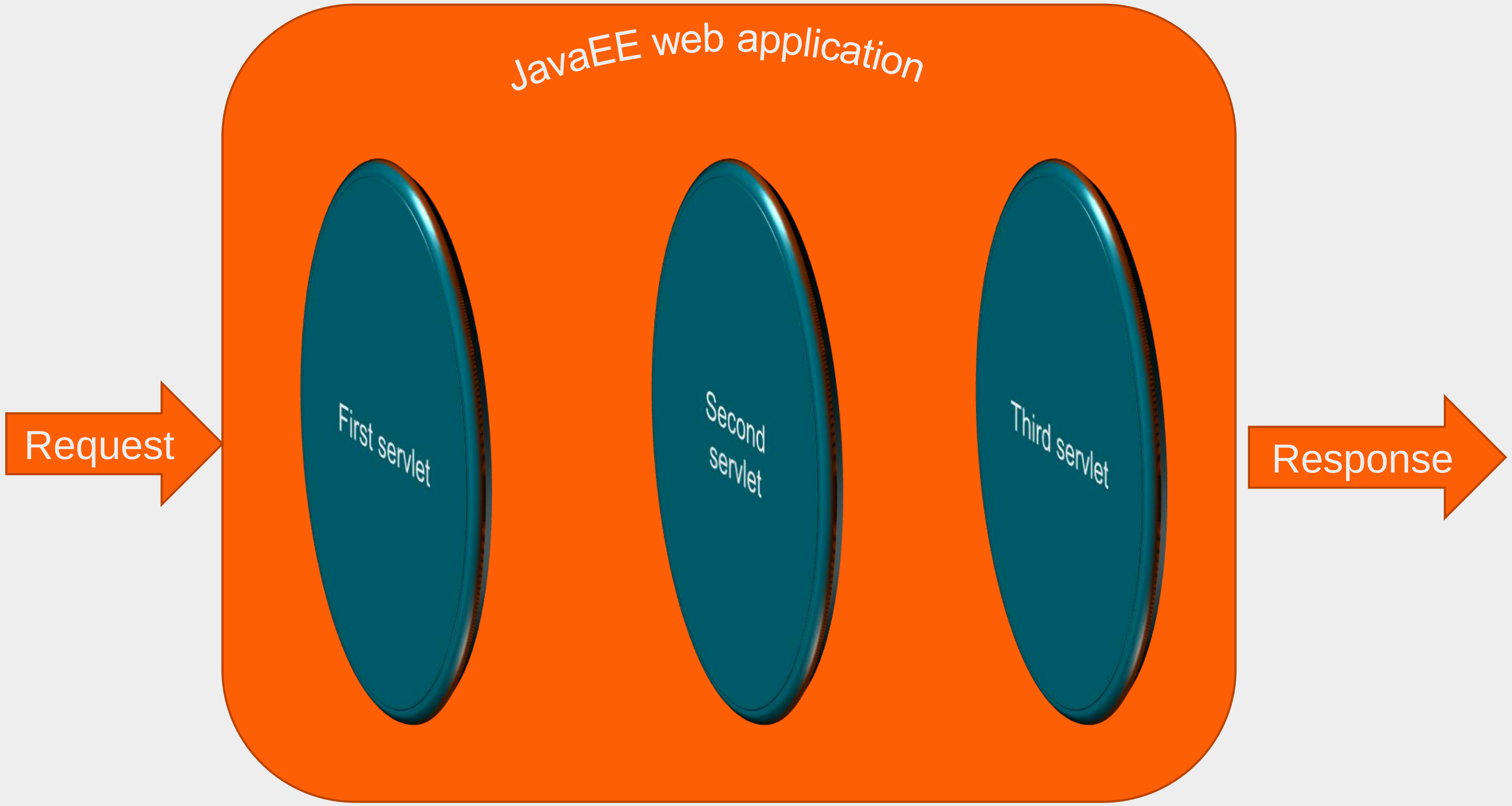
- **setStatus(int)**
  - Sets the status code for a response.
  - **Example : ** setStatus(HttpServletResponse.SC_OK);

    setStatus(HttpServletResponse.SC_MOVED_TEMPORARILY);

Workflow of a web application

- Web components in a JavaEE web application can communicate with each other and co-ordinate the flow of a web application in three distinct ways:
  - Forward
  - Include
  - Redirect

Workflow of a web application (forward)



How stuff works ?

- A forward allows a single request to be processed by multiple components before a response is sent to the client.
- If the path begins with a / it is interpreted as relative to the context root
- The getRequestDispather() and getNamedDispatcher() methods return a reference to an object of javax.servlet.RequestDispatcher
- The difference between the request dispatcher methods of the request and servlet context objects is that the former can accept relative paths
- The getNamedDispatcher( ) method takes a servlet name as configured in the web.xml as argument.

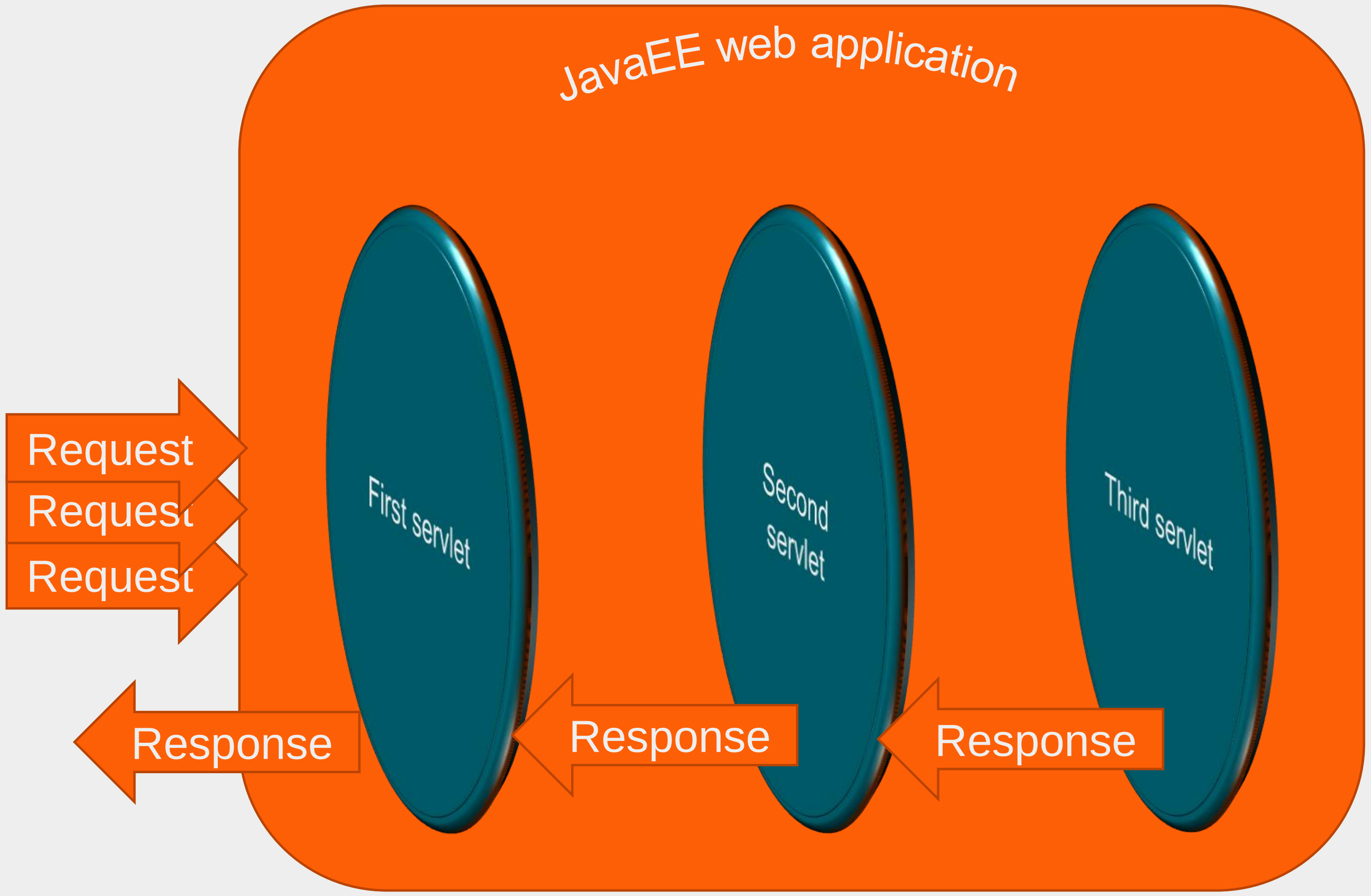
Example:

```
request.getRequestDispatcher(“/somepage.jsp”)
.forward(request,  response);

getServletContext()
.getRequestDispatcher(“/somepage.jsp”)
.forward(request, response);

getServletContext().
getNamedDispatcher(“someservlet”)
.forward(request, response);
```

Workflow of a web application (redirect)



Persistent

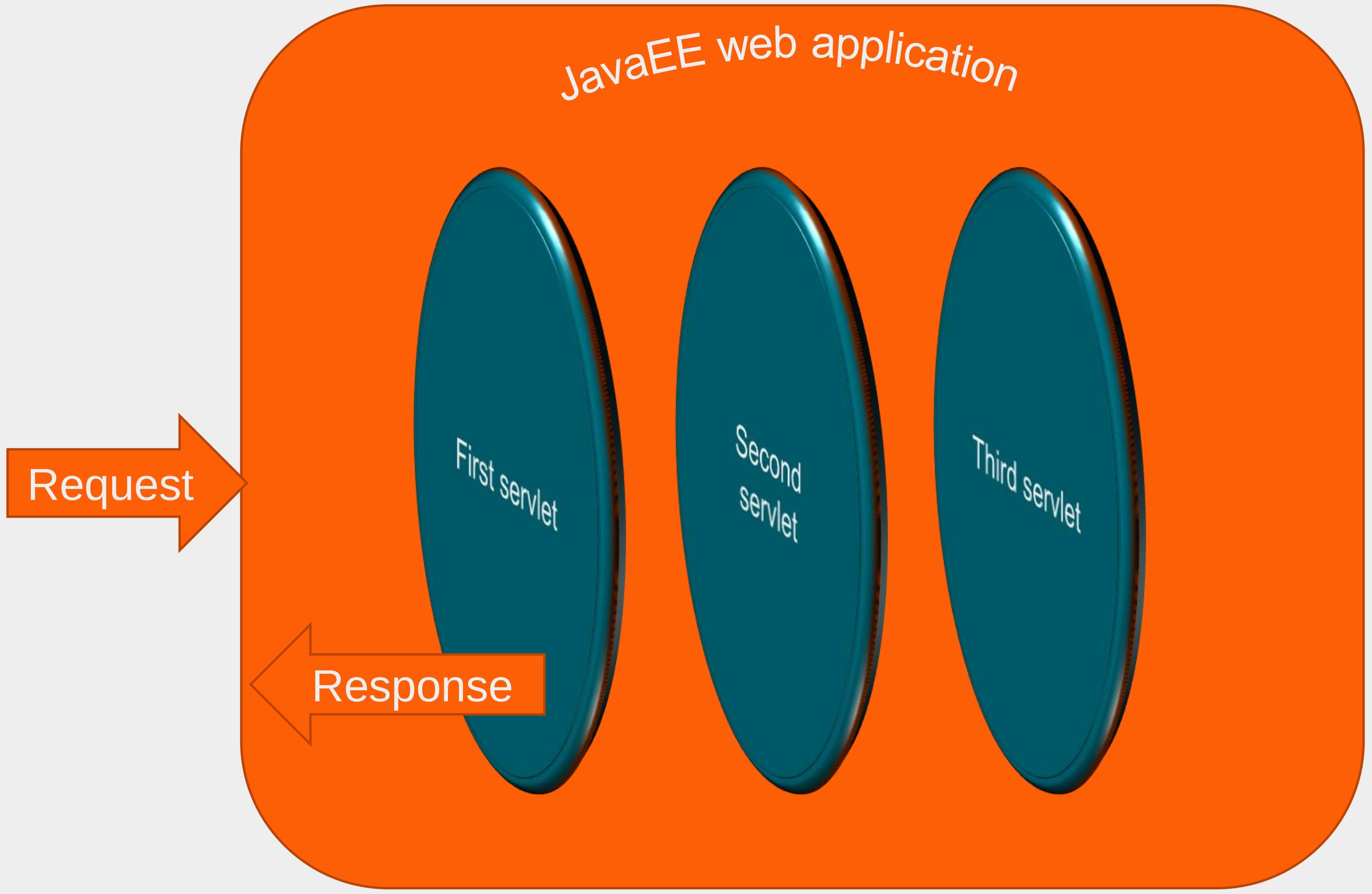
How stuff works ?

- A redirect sends a response to the client with a HTTP status code 302 and the absolute URL of the resource to which a redirect was issued by the web application.
- Upon receiving the response, the client initiates a new request for the resource specified in the response received earlier.
- If the path does not begin with a / it is interpreted as relative to the current web application.
- If the path begins with a / it is interpreted as relative to the servlet container’s root. This is often useful to issue a redirect from a web application to a resource in another web application deployed on the same container.

Example:

```
response.sendRedirect(“some-servlet”);
response.sendRedirect(“somepage.jsp”);
```

Workflow of a web application (include)



Persistent

How stuff works ?

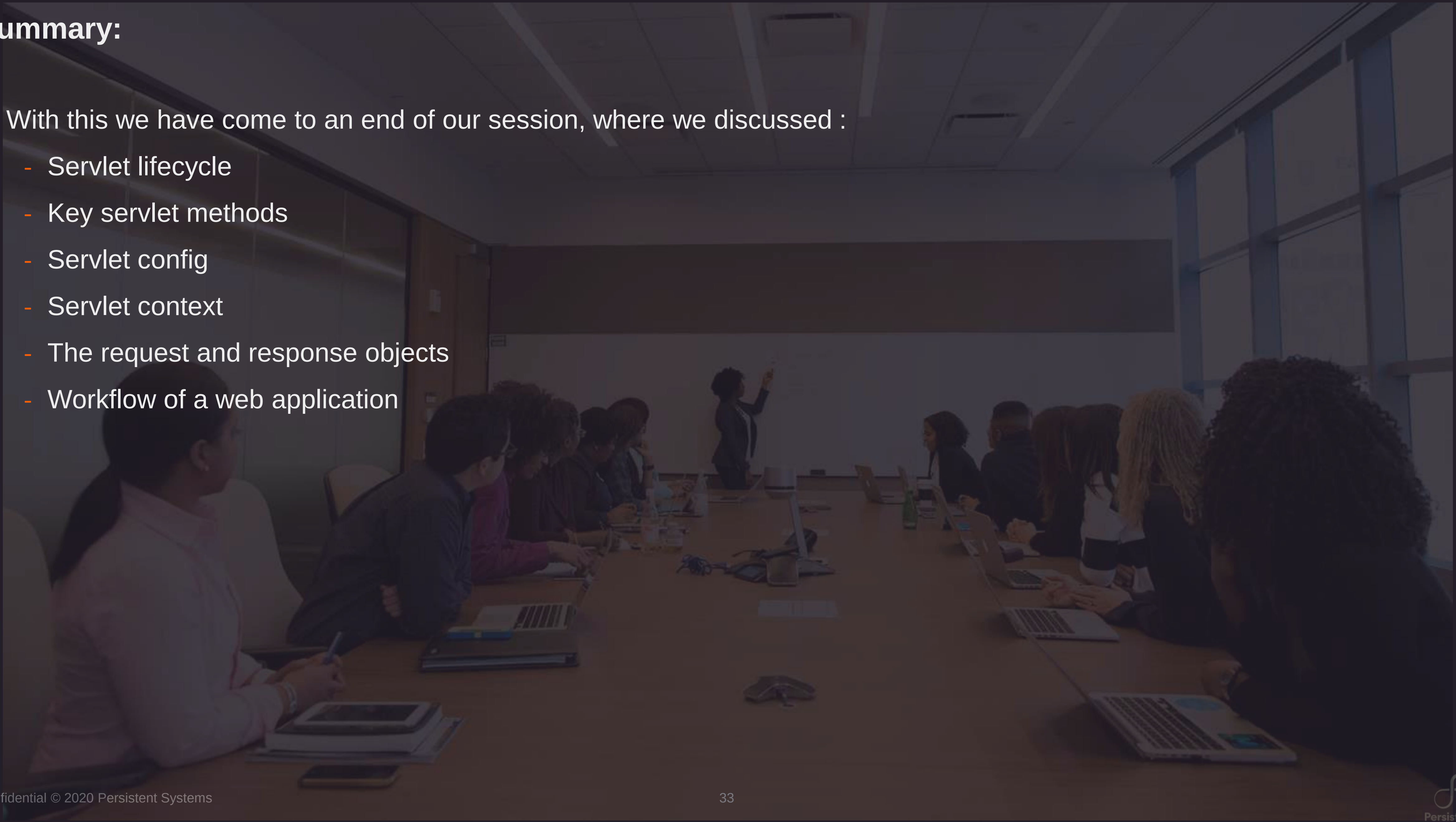
- An include is used to allow multiple web components process the same request and have their responses literally “include-d” or inserted into the current web component’s response.

- Most aspects of an include are similar to that of a forward, seen earlier, expect that an include builds one consolidated response by allowing a request to be processed by multiple web components whereas a forward actually carries the flow of a web application from one web component to another thereby allowing the last component in the forward chain to produce a response.

Example:

```
request.getRequestDispatcher(“/somepage.jsp”)
                                .include(request, response);
getServletContext().getRequestDispatcher(“/somepage.jsp”)
                                .include(request, response);
getServletContext().getNamedDispatcher(“someservlet”)
                                         .include(request, response);
```

Summary:



- With this we have come to an end of our session, where we discussed :
  - Servlet lifecycle
  - Key servlet methods
  - Servlet config
  - Servlet context
  - The request and response objects
  - Workflow of a web application

Appendix

Thank You

Persistent

Thank you