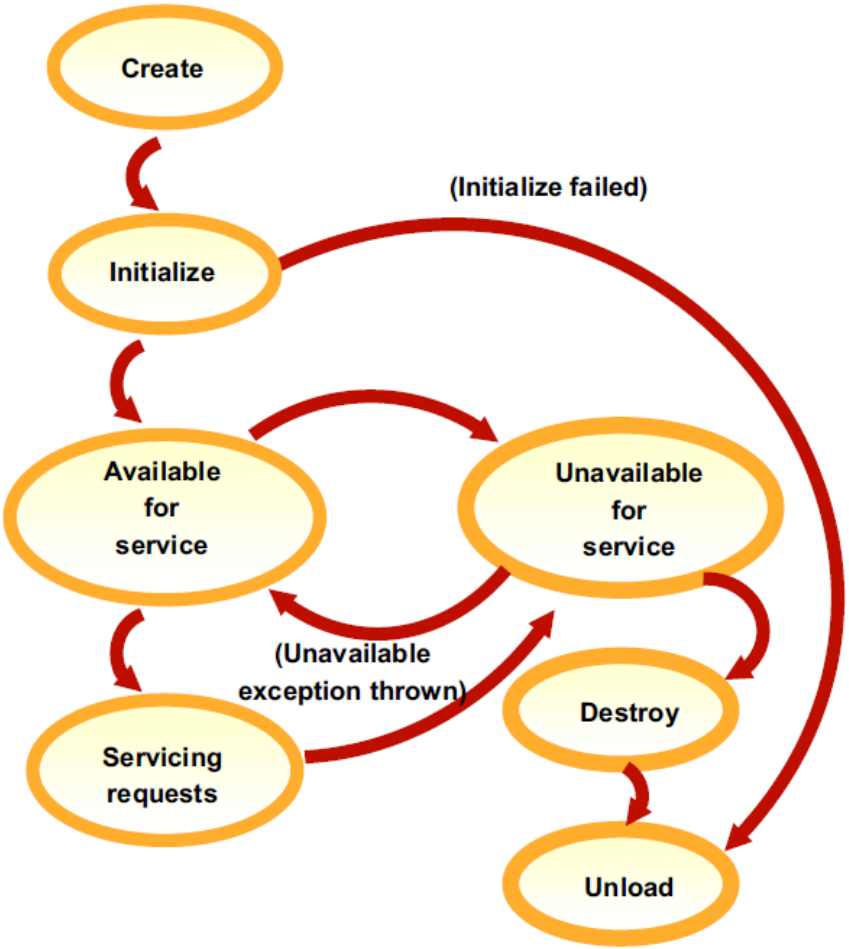
Module 4

Servlet API

Course Objectives

- After completing this module, you should be able to:
  - Explain the lifecycle methods of a Java servlet
  - Describe how servlets:
    - Process HTTP request parameters (for both GET and POST requests)
    - Set the HTTP status code
    - Build the dynamic content for the response

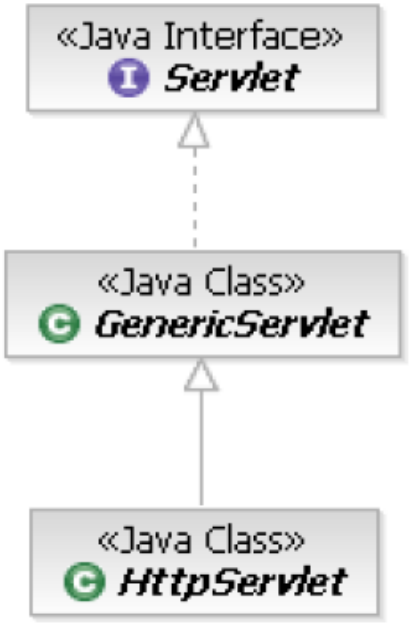
Java Servlet Lifecycle



The Java Servlet API

- The Servlet API is a set of Java classes that defines a standard interface between a Web client and a Web server

- The Servlet API includes two packages:
  - **javax.servlet**
  - **javax.servlet.http**



ServletConfig and Initialization parameters

- `ServletConfig` is used by the Web container to pass information to the servlet during initialization
- Accessed via a GenericServlet method
  - `getServletConfig`
- **ServletConfig**:
  - Contains initialization parameters as a set of name-value pairs

```
public String getInitParameter(String name)
public Enumeration getInitParameterNames
```

  - Maintains a reference to the ServletContext object that gives the servlet information about the server

```
public ServletContext getServletContext
```

Servlet definition in web.xml file

```
<servlet>
      <display-name>RegistrationServlet</display-name>
      <servlet-name>RegistrationServlet</servlet-name>
      <servlet-class>
            com.ibm.exam.servlet.RegistrationServlet
      </servlet-class>
      <init-param>
            <param-name>MaxTries</param-name>
            <param-value>4</param-value>
      </init-param>
      <init-param>
            <param-name>AutoSave</param-name>
            <param-value>false</param-value>
      </init-param>
      <load-on-startup>1</load-on-startup>
</servlet>
 <servlet-mapping>
      <servlet-name>RegistrationServlet</servlet-name>
      <url-pattern>Register</url-pattern>
</servlet-mapping>
```

Example servlet with init parameters

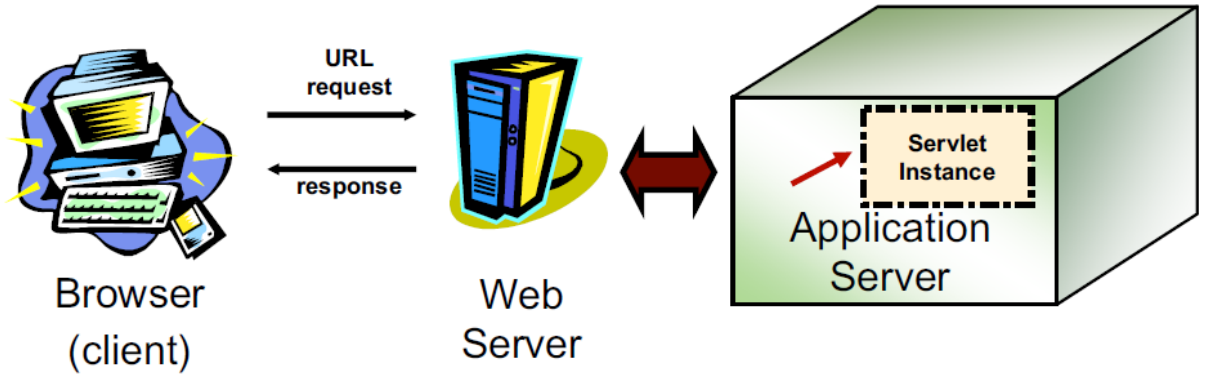
```
// servlet's init method
public void init throws ServletException{
   String maxTries;
   String autoSave;
   maxTries = getInitParameter("MaxTries");
   autoSave = getInitParameter("AutoSave");
   // process the parameters
   ...
}
 or
 // servlet's init method
public void init(ServletConfig config) throws
ServletException{
   String maxTries;
   String autoSave;
   maxTries = config.getInitParameter("MaxTries");
   autoSave = config.getInitParameter("AutoSave");
   // process the parameters
   ...
}
```

HttpServlet

- An HTTP-specific request handler
- Adds HTTP specific methods:
  - doGet: handle a GET request (URL)
  - doPost: handle a POST request (HTML form)
- Subclasses override the doGet, doPost, and other methods, and may override init and destroy
- Typically, doGet and doPost do the work, and are called by service

Requests and Responses

- The service, doGet, and doPost methods each have two parameters:
  - HttpServletRequest: provides access to request data (parameters), HttpSession information, and so forth
  - HttpServletResponse: provides services to allow the servlet to supply a response to the requesting client
- Most servlet programming amounts to reading a request and writing a response



HttpServletRequest

- Represents client's request
- Contains getters for parts of the request:
  - Request header, content type, length, method
  - Request URL as a String and request servlet path
  - Client security type
  - Request parameters
- A scope for object sharing between participants in the request

```
POST /Music/SearchServlet HTTP/1.1
Accept: */*
Referer: http://www.music.ibm.com/Music/musicSearch.html
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; …)
Host: localhost:9080
Content-Length: 50
Connection: Keep-Alive
Cache-Control: no-cache

song_title=Hello&song_artist=Jones&limit_number=20
```

Request Protocol

- The request object encapsulates all of the information from the client request. The following methods are available to access parameters:
  - getParameterNames
    - Returns an **Enumeration** of parameters on the HTML page
  - getParameterValues(String name)
    - Returns the value of a multivalued parameter
  - getParameter(String name)
    - Returns the value of a specific named parameter
  - getReader
    - Returns a **BufferedReader** to view input

Example HTML form

```
<P>Use this form to search for the music you want.
<FORM METHOD="POST" ACTION="/Music/SearchServlet">
   <P>Please enter your search criteria:
   <P>Song title:
   <INPUT NAME="song_title" TYPE="TEXT" SIZE="12"
MAXLENGTH="20">
   <P>Song artist:
   <INPUT NAME="song_artist" TYPE="TEXT" SIZE="15"
MAXLENGTH="25">
   <P>Thank you!
   <INPUT TYPE="SUBMIT">
   <INPUT TYPE="RESET">
</FORM>
```

Reading a post

```
public class SearchServlet extends HttpServlet {
 public void doPost(HttpServletRequest req,
                    HttpServletResponse res)
      throws ServletException, IOException {
      ...
   Enumeration enum = req.getParameterNames;
      while (enum.hasMoreElements)  {
            String name = (String) enum.nextElement;
            String value = req.getParameter(name);
            //… do something with each pair...
      }
   ...
 }
}
```

HttpServletResponse

- Represents communication channel back to client
- Sets the content type and status code
- Sets content headers (cookies, caching, and so forth)
- Allows the servlet to return dynamic content or error information
- Allows the servlet to redirect the client to another URL

Response Control

- **setContentType(String type)**
  - Set the content type for this response
  - Type is a MIME type
- **getWriter**
  - Returns a reference to the PrintWriter
- **getOutputStream**
  - Returns a reference to the ServletOutputStream
  - Used to create binary documents

Setting the status code

- Your response must contain a status code
- Status codes for HTTP 1.1
  - 1xx: Informational – Request received, continuing process (the client needs to respond in some way)
  - 2xx: Success – The action was successful HttpServletResponse.SC_OK
  - 3xx: Redirection – Further action must be taken in order to complete the request See also the HttpServletResponse.sendRedirect method
  - 4xx: Client error – The request contains bad syntax or some other client error
  - 5xx: Server error – The request is valid, but the server failed to fulfill the request
- HttpServletResponse has constants beginning with SC_ for the standard status codes

Simple Servlet

```
public class MyServlet extends HttpServlet {
   public void doGet(HttpServletRequest req,
                     HttpServletResponse res)
          throws ServletException, IOException {
      // get stream to output HTML on!
      res.setStatus(HttpServletResponse.SC_OK);
      res.setContentType("text/html");
      PrintWriter out = res.getWriter;
      // send out a simple banner
      out.println("<HTML><BODY><H1>Today is "+(new Date));
      out.println("</H1></BODY></HTML>");
   }
}
```

Processing an HTTP request

- General flow of processing an HTTP request
  - Process input (forms) data
  - Process input headers
  - Initiate and do the business work
  - Set status code
  - Set response headers
  - Output document
- Note that you must set the status code and headers before any of the output document is actually sent back to the client
  - If the PrintWriter is buffered, the status code and response headers have to be set before the buffer is flushed
- If you are going to redirect the HTTP request to another URL, you must do this before any of the document has been returned to the client

Processing input data

- Form data or query data is sent from the Web browser to the Web server
  - For a GET, form data is passed with request string, that is,
  - http://www.ilscs01.ibm.com/MyServlet?name=jane+Doe
  - For a POST, data passed as a part of the request body
- Fields and values are separated by & and =
- Query data is encoded and must be decoded before it can be processed:
  - The text: name=Margaret Vogel&age=7 1/2 years is encoded as: name=Margaret+Vogel&age=7+1%2F2+years
  - Values can be omitted or duplicated: height=49in&name=&height=125cm
- The Servlet APIs do the work of decoding all this for you

Processing input data example

```
public void processRequest(HttpServletRequest request,
                  HttpServletResponse response) throws... {
 // Called from doGet and doPost
 // Name should have only one value and height is
 // expected to have multiple values
 String name = request.getParameter("name");
 if (name==null) {
       name = "unknown";
 } else if (name.length== 0) {
       name = "missing";
 }
 String height[] = request.getParameterValues("height");
 if (height == null) {
       height = new String[] {"unknown"};
 }
 for (int i = 0; i < height.length; i++){
       if (height[i].length== 0) height[i] = "missing";
 }
 // now process the request headers, set status and response
 // headers and build the output document ...
}
```

Checkpoint

1. How does a Java Servlet use the HttpServletRequest and HttpServletResponse classes?
2. What methods are involved in the lifecycle of a Java Servlet?
3. What must your servlet do to process input parameters from a GET and POST request?
4. How do you set the status for a response?
5. How do you build the output document?

Module Summary

- This unit covered the following topics:
  - Explain the lifecycle methods of a Java servlet
  - Describe how servlets:
    - Process HTTP request parameters (for both GET and POST requests)
    - Set the HTTP status code
    - Build the dynamic content for the response