

Java 4 WEB

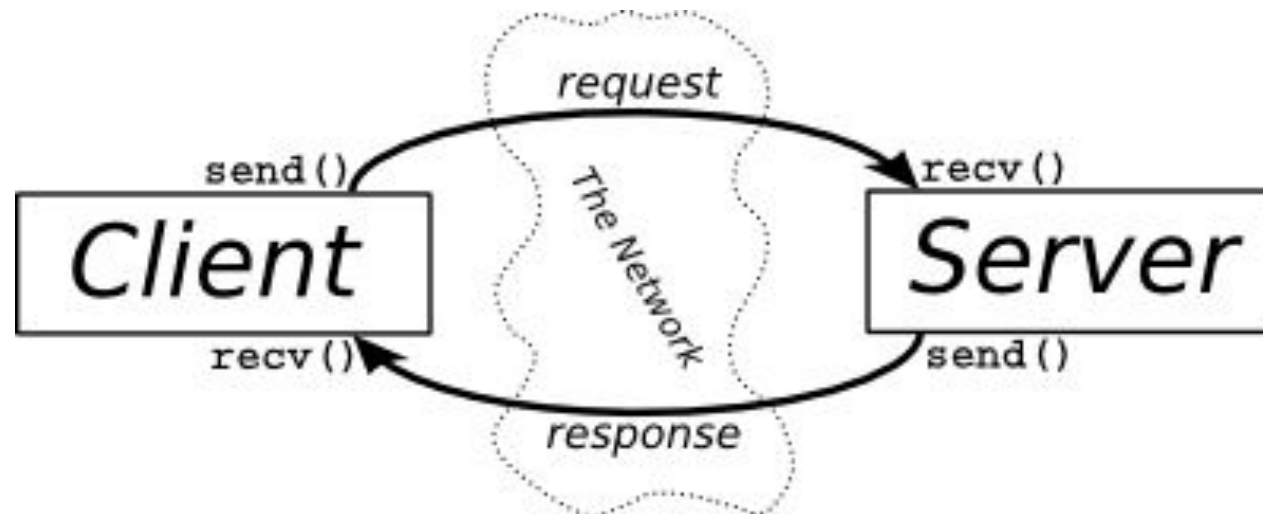
Lesson 12 – Servlet API

Lesson goals

- Client-Server
- Static vs dynamic content
- Java Servlet
- Tomcat

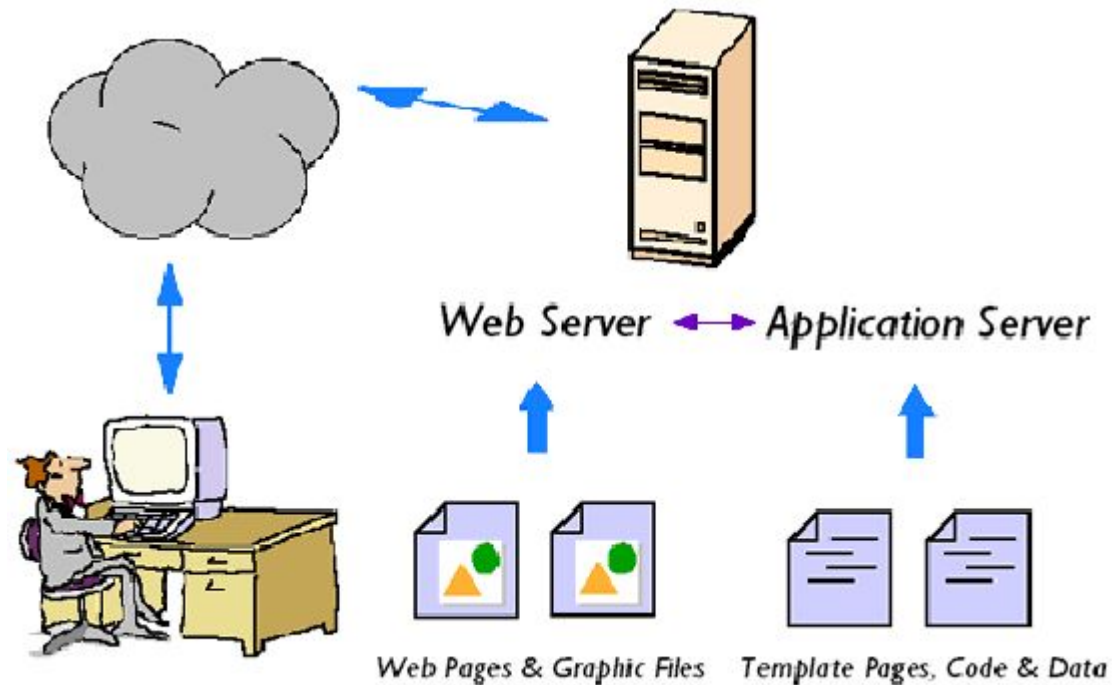
Client-Server architecture

- A person enters a URL string into the browser.
- The browser generates HTTP-request.
- The server receives a request, processes, and sends an HTTP response.
- The browser receives a response and displays it.



Client-Server architecture

- **Web-server for static content** (popular): Apache, nginx, GWS, IIS, ...
- **Software for dynamic content:** CGI programs & modules, PHP AS, MS IIS AS, Java EE, ...



CGI - Common Gateway Interface

Standard used to communicate of the web server with an external program for generating dynamic web content.

CGI-script - a program that works on the CGI-interface. Web-server is configured to redirect certain URL requests for CGI scripts.

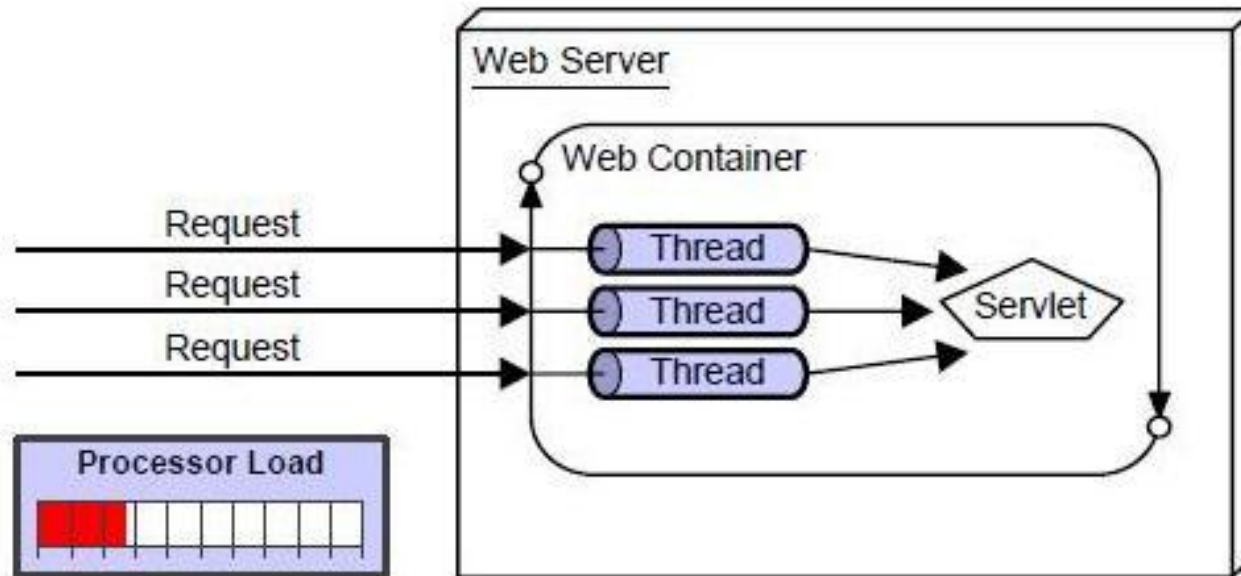
CGI - Common Gateway Interface

Shortcomings of CGI technology:

- Start an OS process for each request
- A lot of low-level coding
- No sharing resources across requests
- No session state

Http Servlet

- `service(HttpServletRequest, HttpServletResponse)`
- `doGet(HttpServletRequest, HttpServletResponse)`
- `doPost(HttpServletRequest, HttpServletResponse)`



Java Servlet

```
@WebServlet(name = "helloServlet", urlPatterns = {"/hello"})  
  
public class HelloServlet extends HttpServlet {  
    public void doGet(  
        HttpServletRequest request,  
        HttpServletResponse response  
    ) throws IOException {  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();  
        out.println("<h1>Hello World</h1>");  
    }  
}
```


Java Servlet 4.0

- **HTTP /2**
 - Request/Response multiplexing
 - Stream Prioritization
 - Server Push
 - Upgrade from HTTP 1.1

Servlet Lifecycle

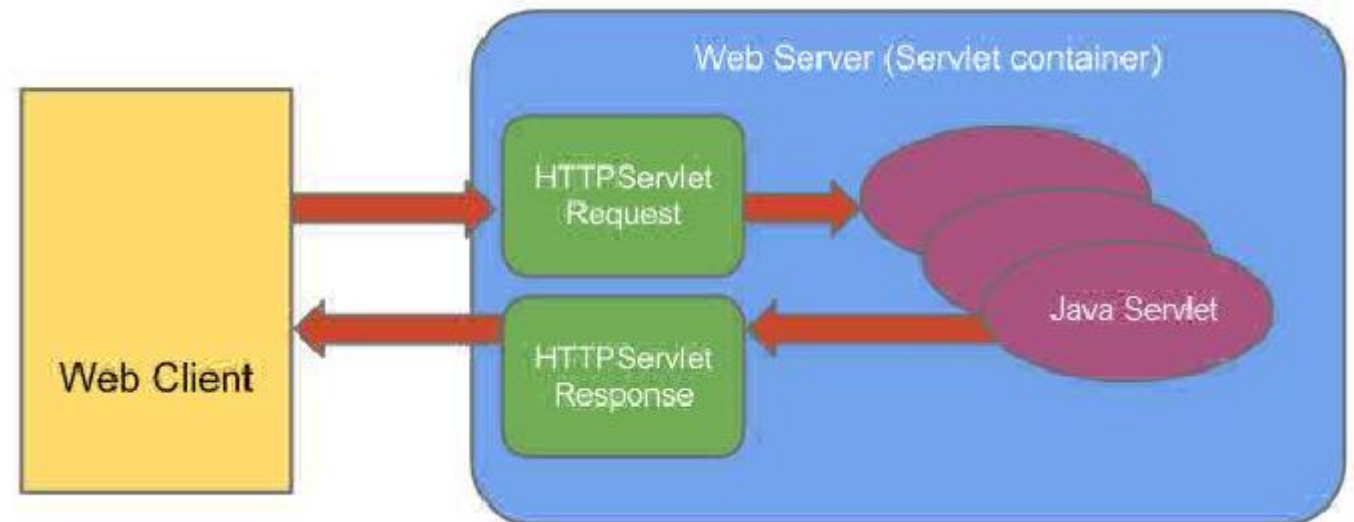
- Loads the servlet class
- Creates an instance of the servlet class
- Initializes the servlet instance by calling the `init()` method
- Does all work
- When the container needs to remove the servlet, it calls the servlet's `destroy()` method

HttpServletRequest

Container creates HttpServletRequest object and passes it as argument to Servlet's service methods (doGet, doPost, etc.)

HttpServletRequest provides client request info to Servlet.

Passed to the service(...) method, doGet (...), doPost (...), do ... ().



HttpServletRequest

Methods:

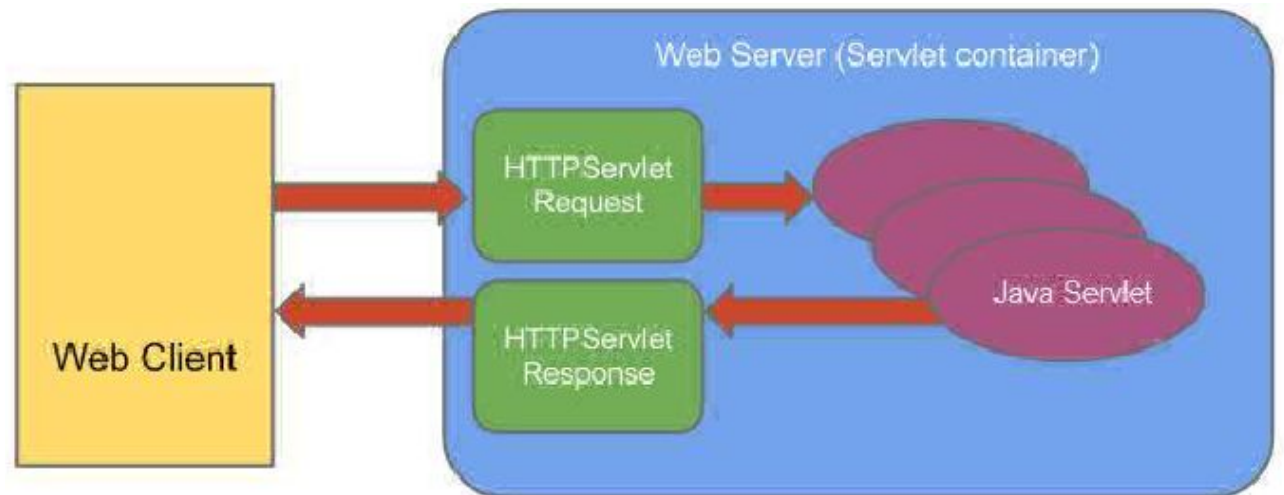
- `getHeader (String)`
- `getParameter (String)`
- `getInputStream ()`
- `getSession ()`
- `getServletContext ()`
- etc

HttpServletResponse

Container creates HttpServletResponse object and passes it to Servlet's service methods (doGet, doPost, etc.)

HttpServletResponse assists a servlet in sending a response to the client.

As well as HttpServletRequest is passed in service(...) and do...(...)



HttpServletResponse

Methods:

- addHeader (String, String)
- setStatus (int)
- sendError (int)
- sendRedirect (String)
- setHeader (String, String)
- getWriter ()
- etc

HttpSession

```
HttpSession session = request.getSession();
```

Main methods:

- invalidate ()
- getAttributeNames ()
- getAttribute (String)
- removeAttribute (String)
- setAttribute (String, Object)
- etc

ServletContext

Servlet uses ServletContext to communicate with its servlet container, access the servlet container.

```
ServletContext sc = request.getServletContext();
```

Main methods:

- `getAttributeNames ()`
- `getAttribute (String)`
- `removeAttribute (String)`
- `setAttribute (String, Object)`
- etc

Filter

```
@WebFilter(urlPatterns = {"/*"})  
public class HelloFilter implements Filter {  
    public void init(FilterConfig cfg) {  
    }  
  
    public void doFilter(  
        ServletRequest req, ServletResponse resp, FilterChain chain  
    ) throws ServletException, IOException {  
        chain.doFilter(req, resp);  
    }  
  
    public void destroy() {  
    }  
}
```

Error Handling

```
@WebServlet("/error")
public class ErrorHandler extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException
    {
        resp.setContentType("text/html; charset=utf-8");
        try (PrintWriter writer = resp.getWriter()) {
            writer.write("<html><head><title>Error description</title></head><body>");
            writer.write("<h2>Error description</h2>");
            writer.write("<ul>");
            Arrays.asList(
                ERROR_STATUS_CODE,
                ERROR_EXCEPTION_TYPE,
                ERROR_MESSAGE)
                .forEach(e ->
                    writer.write("<li>" + e + ":" + req.getAttribute(e) + " </li>"));
        };
        writer.write("</ul>");
        writer.write("</html></body>");
    }
}
```

Error Handling

```
<?xml version="1.0" encoding="UTF-8" ?>
<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">

    <display-name>Web Project</display-name>

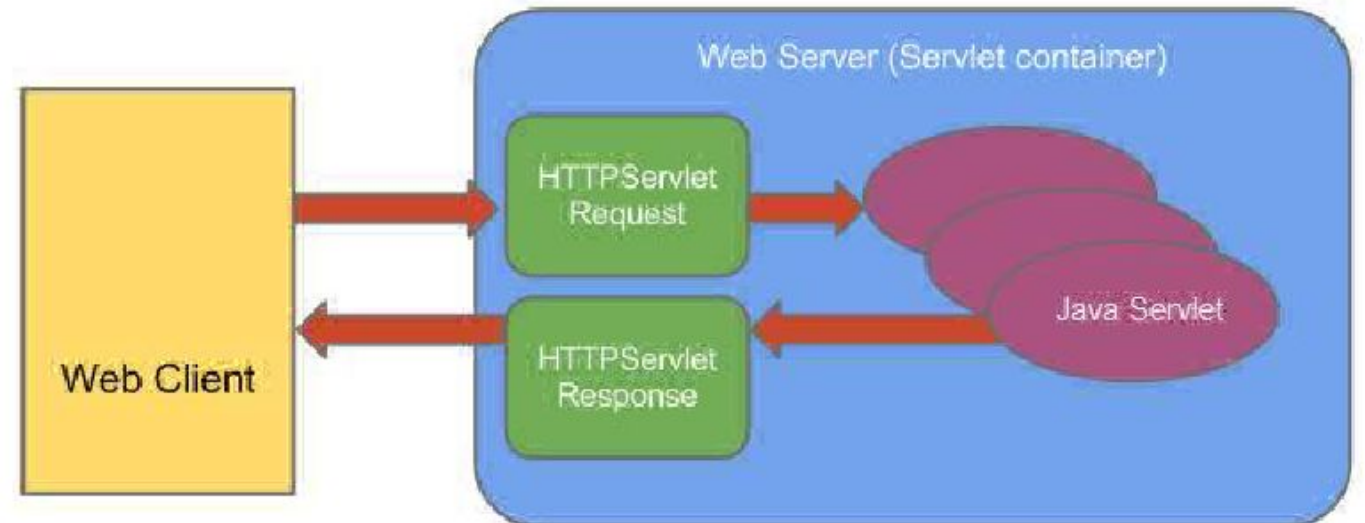
    <error-page>
        <exception-type>java.lang.RuntimeException</exception-type>
        <location>/error</location>
    </error-page>
</web-app>
```

Web (servlet) container

A Web server - component that interacts with Java Servlets.

Primary responsibilities are:

- Management of servlets and their life cycle
- Mapping the URL to a specific servlet.
- Check access rights



Java EE & Servlet (Web) Container














Most popular Servlet (Web) Containers

- Apache Tomcat
- Jetty
- JBoss Application Server (WildFly)
- ...



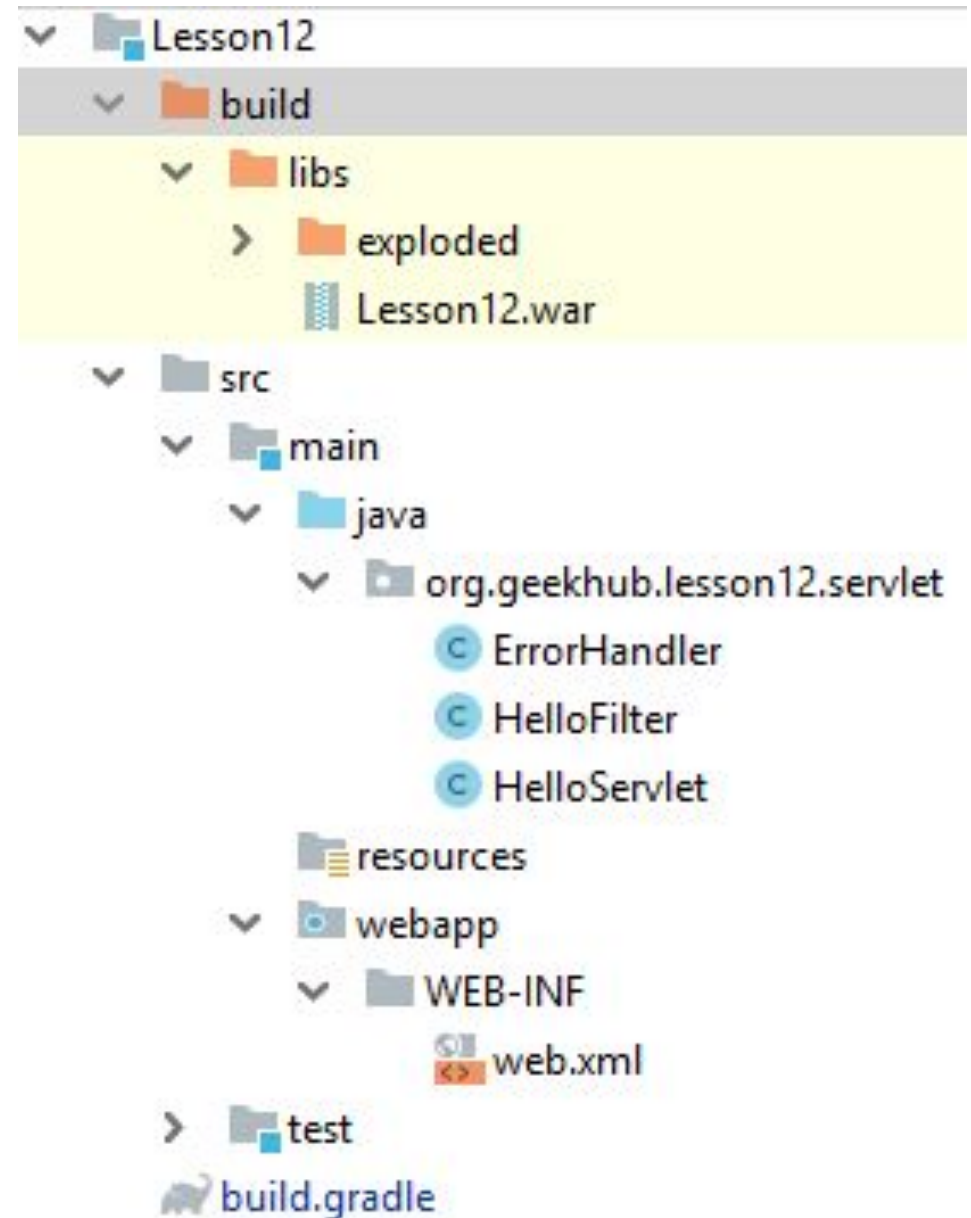
Apache Tomcat



| Downloads | |
|--|-------------------|
| Name | Date Modified |
|  apache-tomcat-8.0.15.zip | Nov 2, 2014, 1:54 |
| ▼  apache-tomcat-8.0.15 | Nov 2, 2014, 7:28 |
| ▶  logs | Nov 2, 2014, 7:25 |
| ▶  work | Nov 2, 2014, 7:25 |
|  LICENSE | Nov 2, 2014, 7:26 |
|  NOTICE | Nov 2, 2014, 7:26 |
|  RELEASE-NOTES | Nov 2, 2014, 7:26 |
|  RUNNING.txt | Nov 2, 2014, 7:26 |
| ▶  webapps | Nov 2, 2014, 7:26 |
| ▶  bin | Today, 11:30 PM |
| ▶  conf | Today, 11:30 PM |
| ▶  lib | Today, 11:30 PM |
| ▶  temp | Today, 11:30 PM |

WAR

Web application **A**Rchive - is a JAR file used to distribute a collection of JavaServer Pages, Java Servlets, Java classes, XML files, libraries, static web pages (HTML, JS, CSS, etc) and other resources...



WAR

```
plugins {  
    id "java"  
    id "war"  
}  
  
repositories {  
    mavenCentral()  
}  
  
dependencies {  
    testCompile libs.test  
    compileOnly 'javax.servlet:javax.servlet-api:4.0.0'  
}  
  
test {  
    useTestNG {}  
}
```


Literature

- [Building Java Web Applications](#)
- [Servlets \(by Baeldung\)](#)
- [Servlets \(by Tutorialspoint\)](#)
- [How to Install Apache Tomcat](#)
- [What is servlet container](#)
- [Servlets \(by JetBrains\)](#)

Homework Task 1

Implement a servlet with interface:

```
`/session?action=[add/update/remove/invalidate]&name=...&value=...`
```

1. The servlet generates a form with input fields "*action*", "*name*", "*value*" and button "*Submit*". Below print is a list of all attributes of the user session.

- *action* - depending on the value, the servlet adds, updates, or removes an attribute in/from a session;
- *name* - the name of the attribute;
- *value* - is the value of the attribute.

2. If any error happens – log and redirect user to custom error page

3. Add request blocking filter, when user browser is 'Microsoft Edge' – block request and show error page. Use

4. Add separate request logging filter. Log endpoints path and total execution time. Should measure all actions (even when request blocked)

5. Add separate request blocking filter. When today is weekend – deny all operations. Microsoft Edge users should never come here and be stopped by user-agent filter.

6. Attributes lists should not be shared between two browsers

Homework Task 1

Action:

Name:

Value:

string: tree
name: Harry

Action:

Name:

Value:

Action:

string: tree
name: Harry

Operation failed

Parameter name is not specified