Internet applications
Lecture 3

Introduction to Java Servlets™

Bartosz Walter

Bartek.Walter@man.poznan.pl

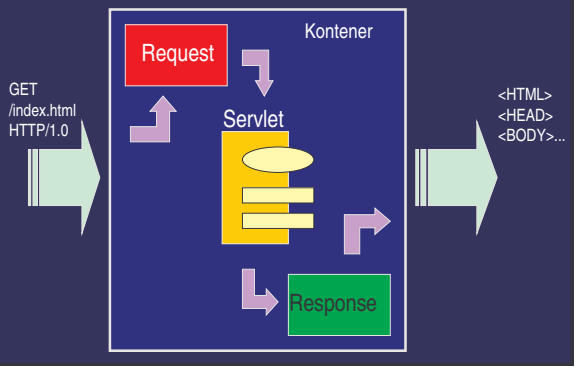
Agenda

1. CGI vs. Java Servlets™
2. Concept of request processing in a servlet container
3. Servlet life-cycle
4. Java Servlets™ API
5. Session management
6. Inter-servlet communication
7. Multipart requests
8. Filtering requests and responses
9. Freemarker – a templating engine for Java

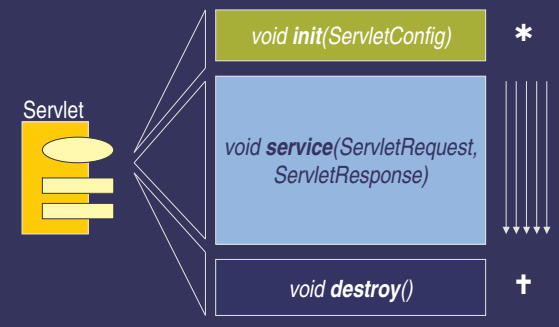
CGI vs. Java Servlets™

CGI	Java Servlets™
external to the server	built into the server
uses heavyweight processes	threaded
communicates through environment variables	communicates to the server through a dedicated protocol
low scalability	highly scalable
language independent	proprietary (Java)
standardized by W3C	standardized by JCP

Servlet container



Servlet life-cycle



`javax.servlet.HttpServlet`

Servlet example

```
public class MójServlet extends HttpServlet {
    public void init() {
    }

    public void doGet(HttpServletRequest rq, HttpServletResponse rs)
    throws ServletException, IOException {
        rs.setContentType(„text/html”);
        PrintWriter wr = rs.getWriter();
        wr.println(„<h1>Hello, World!</h1>”);
        String a = rq.getParameter(„a”);
        wr.println(„Parameter ‘A’ = „ + a);
        wr.flush();
    }

    public void destroy() {
    }
}
```

Configuring the servlet

Servlet name

Servlet parameters

Mapping URLs to servlets

web.xml

```
<servlet>
  <servlet-name>serwlet_A</servlet-name>
  <servlet-class>
    moj.pakiet.ServletA
  </servlet-class>
  <init-param>
    <param-name>parametr</param-name>
    <param-value>wartość</param-value>
  </init-param>
</servlet>

<servlet-mapping>
  <servlet-name>serwlet_A</servlet-name>
  <url-pattern>/A/*</url-pattern>
</servlet-mapping>
```

Java Servlets™ API: HttpServlet

```
interface javax.servlet.HttpServlet {
  void destroy()
  void init()

  ServletConfig getServletConfig()

  void service(HttpServletRequest, HttpServletResponse)
  void doGet(HttpServletRequest, HttpServletResponse)
  void doPost(HttpServletRequest, HttpServletResponse)
}
```

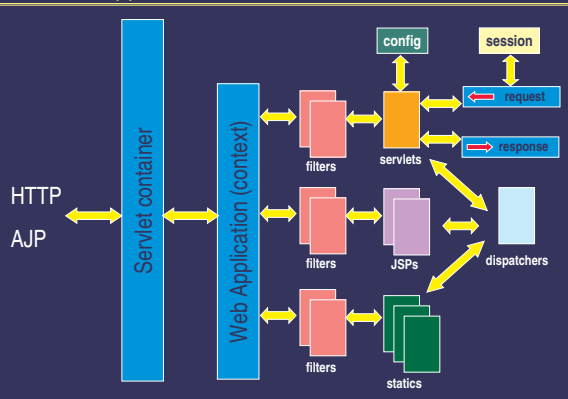
```
void init() {
  String param = config.getInitParameter(„param")
  //....
}
```

Java Servlets™ API: HttpServlet

```
interface javax.servlet.ServletConfig {
  String getInitParameter(String name)
  Enumeration getInitParameterNames()
  ServletContext getServletContext()
}
```

```
void init() {
  String param = config.getInitParameter(„param")
  //....
}
```

Java Web Application



Java Servlets™ API: ServletContext

```
interface javax.servlet.ServletContext {
  Object getAttribute(String name);
  void setAttribute(String name, Object value);
  void removeAttribute(String name);
  Enumeration getAttributeNames();

  RequestDispatcher getNamedDispatcher(String name);
  RequestDispatcher getRequestDispatcher(String url);
}
```

Java Servlets™ API: HttpServletRequest

```
public interface javax.servlet.http.HttpServletRequest {
  String getContextPath();
  Cookie[] getCookies();
  String getHeader(String);
  Enumeration getHeaderNames();
  String getMethod();
  String getPathInfo();
  String getQueryData();

  String getParameter(String);
  Enumeration getParameterNames();
  Enumeration getParameterValues(String);
  Map getParameterMap();
  HttpSession getSession(boolean createNew);

continued on next page...
```

Java Servlets™ API: HttpServletRequest (cont.)

```
  ...continued
  String getContentType();
  int getContentLength();

  String getAttribute(String name);
  void setAttribute(String name, Object value);
  void resetAttribute(String name);

  String getCharacterEncoding();
  void setCharacterEncoding(String);
  Enumeration getLocales();
  InputStream getInputStream();
  BufferedReader getReader();
  RequestDispatcher getRequestDispatcher(String url);
  boolean isSecure();
}
```

Java Servlets™ API: HttpServletResponse

```
interface javax.servlet.http.HttpServletResponse {
  String getCharacterEncoding();
  Locale[] getLocale();

  ServletOutputStream getOutputStream();
  Writer getWriter();

  void setContentLength(int);
  void setContentType(String);
  void setLocale(Locale);
  void addCookie(Cookie);
  void addHeader(String name, String value);
  void encodeURL(String);
  void sendError(int);
  void sendRedirect(String);
  continued on next page...
```

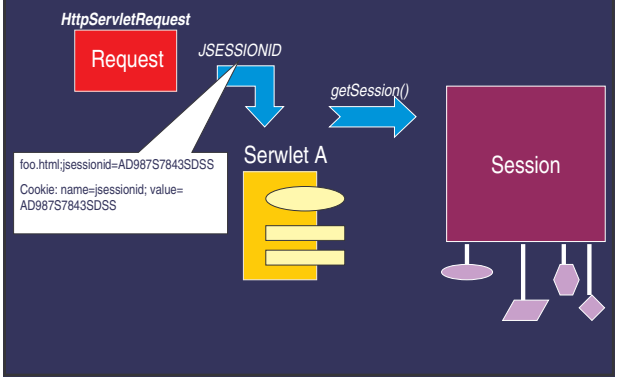
Java Servlets™ API: HttpServletResponse (cont.)

```
                    ...continued from previous page
  void setHeader(String name, String value);
  void setStatus(int);
```

Java Servlets™ API: HttpSession

```
interface javax.servlet.http.HttpSession {
  Object getAttribute(String);
  void setAttribute(String name, Object value);
  void removeAttribute(String);
  Enumeration getAttributeNames();

  long getCreationTime();
  String getId();
  void invalidate();
  boolean isNew();
}
```

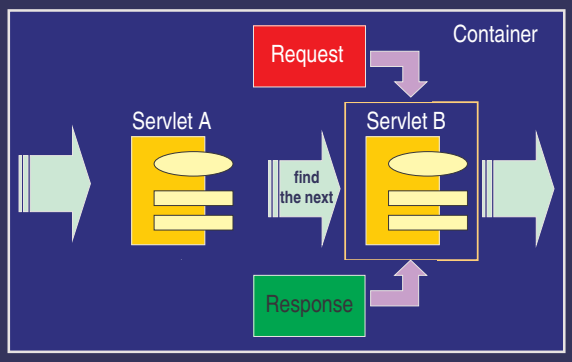
Session management



Storing data

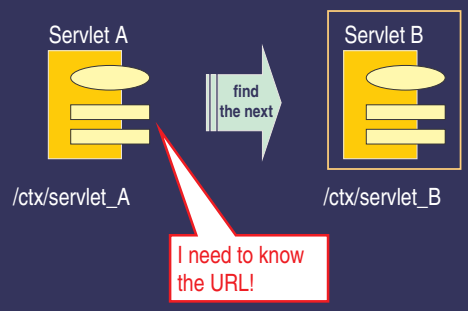
- context
  - context.setAttribute(key, value);
  - visible for all application resources
  - durable for application life-span
- session
  - session.setAttribute(key, value);
  - visible session participants (requests sent from a single browser)
  - durable for session life-span
- request
  - request.setAttribute(key, value);
  - visible for request handling objects
  - durable for request life-span

Inter-servlet communication



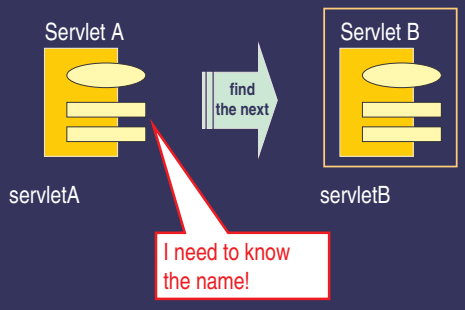
Resolving the servlet mappings

Method 1: URL-based assignment

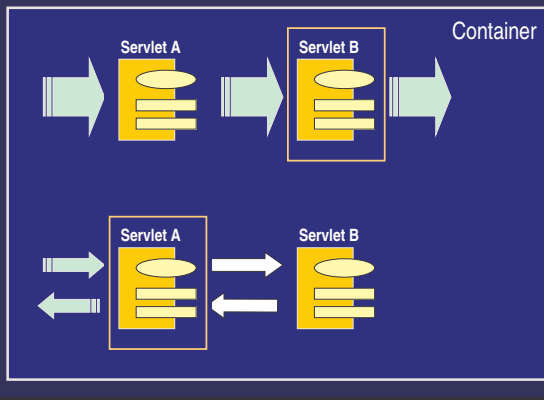


Resolving the servlet mappings

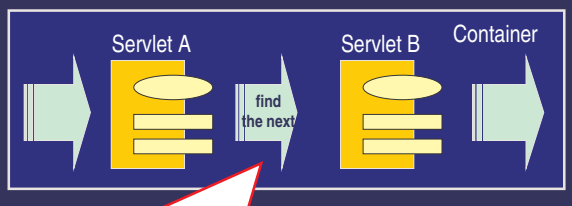
Method 2: servlet-name mapping



Inter-servlet communication

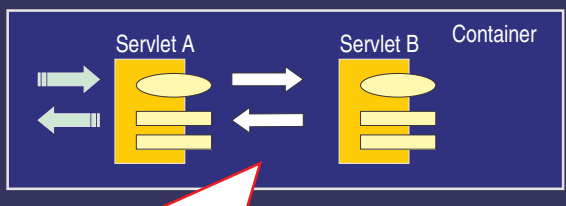


Inter-servlet communication: *forwarding*




```
RequestDispatcher rd =
      ctx.getRequestDispatcher("/serwlet_B");
rd.forward(request, response);
// ServletA is unavailable!
```

Inter-servlet communication: *including*




```
RequestDispatcher rd =
      ctx.getRequestDispatcher("/serwlet_B");
rd.include(request, response);
// ServletA is in control again
```

Pros & cons

Forwarding

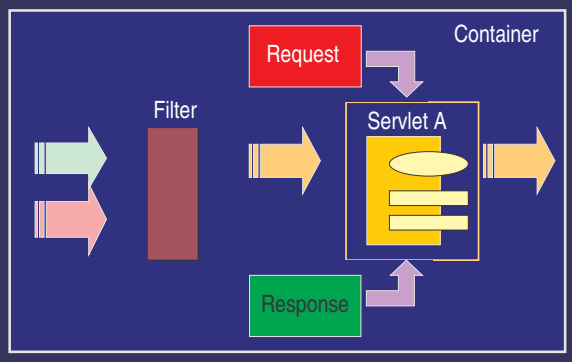
- ✓simple pipelining
- ✓quasi-filtering
- ✓erroneous response commitment

Including

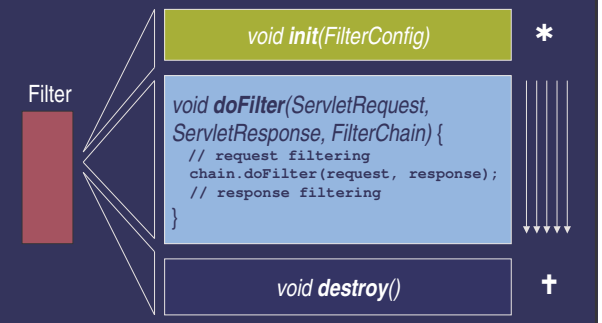
- ✓delegation-based processing
- ✓inflexible deployment

- ✓non-configurable
- ✓change in topology enforces recompilation

Filtering



Filter API



`javax.servlet.Filter`

Filter configuration

Filter name definition

Filter parameters

Assigning filter to URL or servlet.

web.xml

```
<filter>
  <filter-name>filter_1</filter-name>
  <filter-class>
    moj.pakiet.Filter1
  </filter-class>
  <init-param>
    <param-name>parametr</param-name>
    <param-value>wartość</param-value>
  </init-param>
</servlet>

<filter-mapping>
  <filter-name>filter_1</filter-name>
  <url-pattern>/A/*</url-pattern>
</filter-mapping>
```

Java Servlets™ API

```
interface javax.servlet.Filter
  void doFilter(request, response, FilterChain);
  void init(FilterConfig);
  void destroy();
```

Java Servlets™ API

```
interface FilterConfig
  String FilterName();
  String getInitParameter(String);
  String getInitParameterNames();
  ServletContext getServletContext();
```

Filter example

```
public class SetCharacterEncodingFilter implements Filter {
    protected FilterConfig filterConfig = null;

    public void doFilter(ServletRequest rq, ServletResponse rs,
       FilterChain chain)
    throws IOException, ServletException {
       String encoding = filterConfig.getInitParameter("encoding");
       request.setCharacterEncoding(encoding);
       chain.doFilter(request, response);
    }

    public void init(FilterConfig filterConfig)
    throws ServletException {
       this.filterConfig = filterConfig;
    }
}
```

Filter example

```
<filter>
  <filter-name>SetCharacterEncodingFilter</filter-name>
  <filter-class>filters.SetCharacterEncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
</servlet>

<filter-mapping>
  <filter-name>SetCharacterEncodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

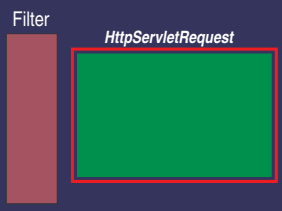
Filter example

```
public class SetCharacterEncodingFilter implements Filter {
    protected FilterConfig filterConfig = null;

    public void doFilter(ServletRequest rq, ServletResponse rs,
       FilterChain chain)
    throws IOException, ServletException {
       String encoding = filterConfig.getInitParameter("encoding");
       request.setCharacterEncoding(encoding);
       chain.doFilter(request, response);
    }

    public void init(FilterConfig filterConfig)
    throws ServletException {
       this.filterConfig = filterConfig;
    }
}
```

Filters and wrappers

Request and response are immutable objects!

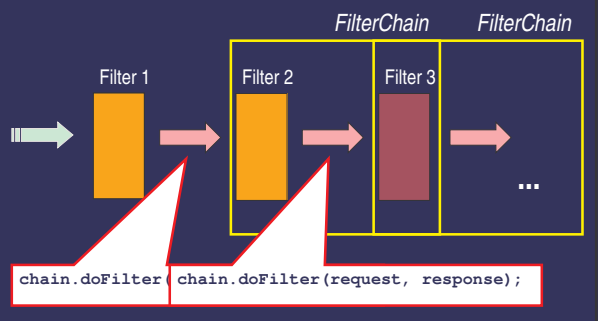
Response, once commited, cannot be modified!



`javax.servlet.HttpServletRequestWrapper`

`javax.servlet.HttpServletResponseWrapper`

Filter pipelining



Filter pipelines configurations

Filter_1 and filter_2 names definition

Filter_1 starts...

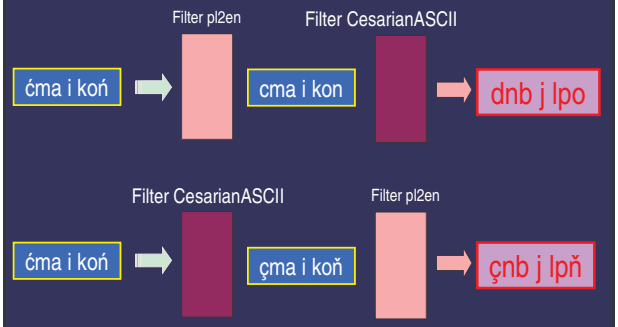
...and filter_2 follows.

web.xml

```
<filter>
  <filter-name>filter_1</filter-name>
  ...
</servlet>
<filter>
  <filter-name>filter_2</filter-name>
  ...
</servlet>

<filter-mapping>
  <filter-name>filter_1</filter-name>
  <filter-name>filter_2</filter-name>
  <url-pattern>/A/*</url-pattern>
</servlet-mapping>
```

Order of filters

What if the order changes?



Freemarker – a templating engine for Java™

✓Writing the content by the servlet is inflexible

http://freemarker.sourceforge.net/

```
void doGet(request, response) {
  HashModel root = new HashModel();
  root.put("name", someValue);

  Template tmpl = FileTemplateCache.getTemplate("/index.html");
  tmpl.process(root, response)
}
```

```
<HTML>...
${name}
</HTML>
```

Multipart requests

```
POST /processForm.html HTTP/1.0
Content-type: multipart/user-form
```

✓Java Servlets™ do not handle multipart requests

Solutions:

1. wrapper over the *HttpServletRequest*
2. parsing filter

`com.oreilly.servlet.*` at http://www.servlet.com/

Summary

- servlets – a powerful technology for writing web applications
- several useful libraries supporting the development
- Sun proposes interfaces, vendors implement the specification
- open implementations

Q & A