Servlet Komponenten:
Servlet, Filter, Listener

Themen heute

- Komponente Servlet
  - Repetition Servlet anhand UB1
  - Lösung liegt auf dem AD
- **Servlet Context** und **Servlet Container**
  - Einführung
- Komponente **Filter**
  - Praktische Übung zum Filter
- Komponente **Listener**
  - Praktische Übung zu Listener und ServletContext

Besprechung Übung 1

- Request auf
  http://localhost:8080/flashcard-basic/questionnaires/1

- Lösungsschritte
  1. Dispatching des Requests
  2. Questionnaire ID aus dem Request extrahieren
  3. Questionnaire mit ID aus Repository lesen
  4. HTML Response generieren

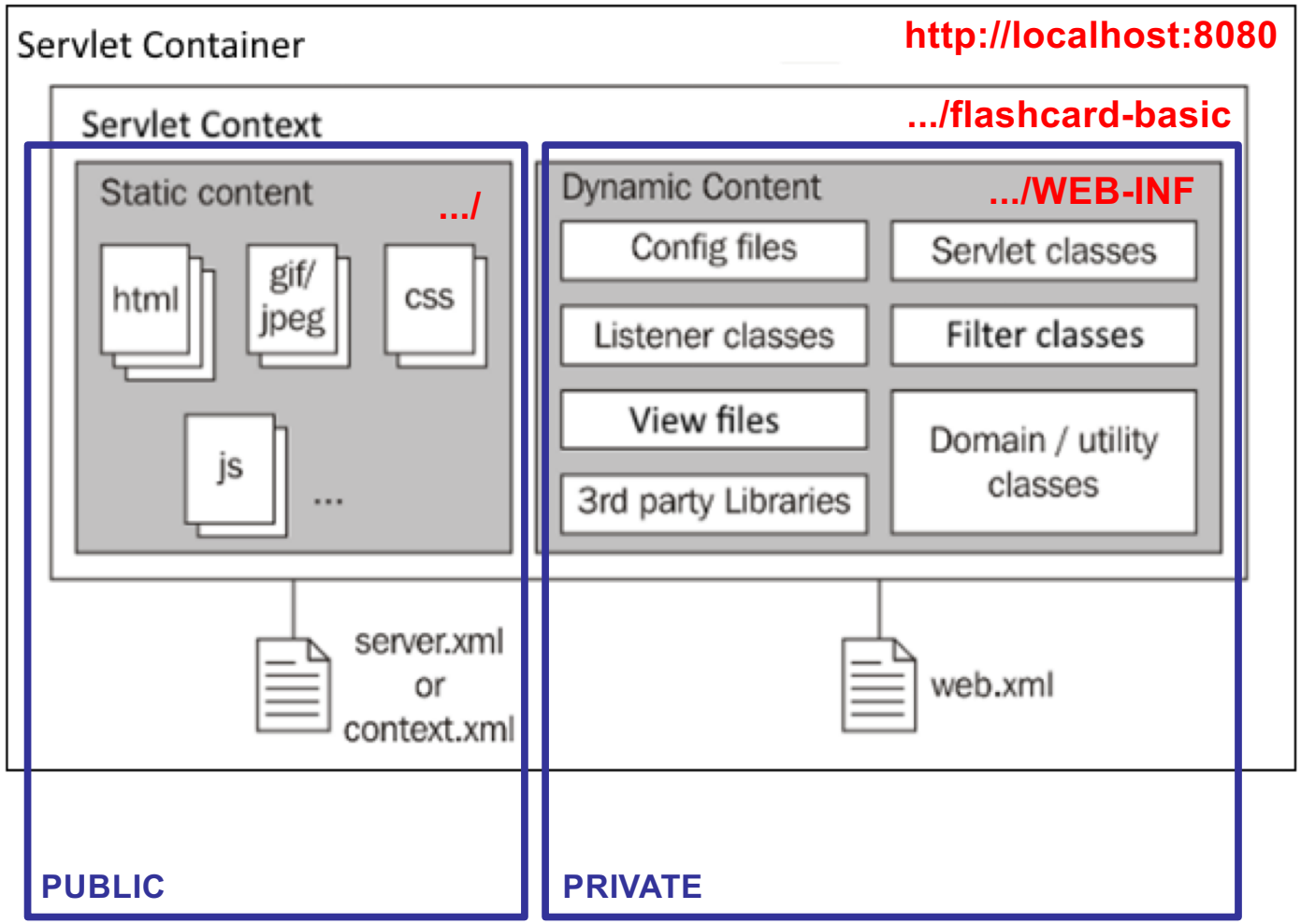
Servlet Context

- Verwaltet Werte, die der gesamten WebApp zur Verfügung stehen.

- Zugriff auf die Initialisierungsparameter der WebApp
  - servletContext.getInitParameter("something")

- Verwaltung von Attributen während der Laufzeit
  - servletContext.setAttribute("myAttr", myValue)
  - servlet.Context.getAttribute("myAttr")

Servlet Container

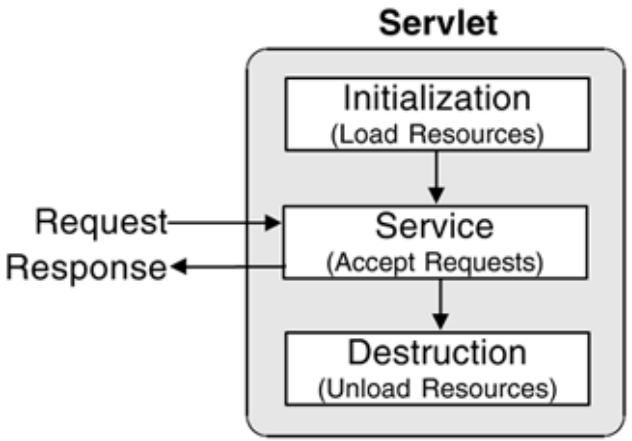


Servlet Container: Zugriff auf Ressourcen

- statische Ressourcen
  - Bilder, Dokumente, ...
  - **über URL direkt zugreifbar (PUBLIC)**
- dynamische Ressourcen
  - die Webapplikation mit den class-Files, den jar-Bibliotheken, ...
  - **über URL direkt NICHT zugreifbar (PRIVATE)**
    → Webapplikation verwaltet die Ressourcen
- Konfiguration der Webapplikation für Zugriff auf statische und dynamische Ressourcen gleichzeitig:
  - URL für dynamische Ressourcen über Servlet Mapping z.B. auf "/app/*" anpassen!

Servlet

- Welches sind ihrer Meinung nach die wichtigsten Punkte, die man bei einem Servlet kennen sollte?
  - Aufgaben eines Servlets
    - Servlet setzt einen Use Case um
    - Servlet ist Endpoint eines Requests
  - Programmierung eines Servlets
    - Parameter "Request" vs. "Response" bei "doXXX()" Methoden
    - Reihenfolge "ContentType" und Einsatz von "getWriter()"
  - Laufzeitverhalten eines Servlets
    - Servlet ist ein Singleton
    - Konfiguration
      - web.xml vs. Annotation

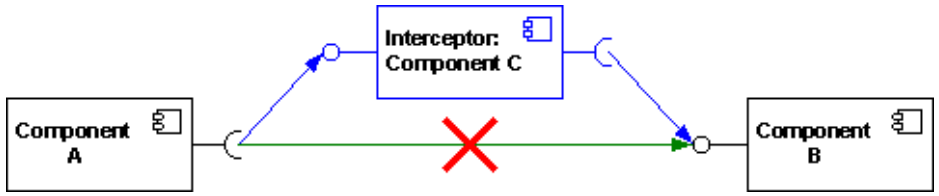


Filter

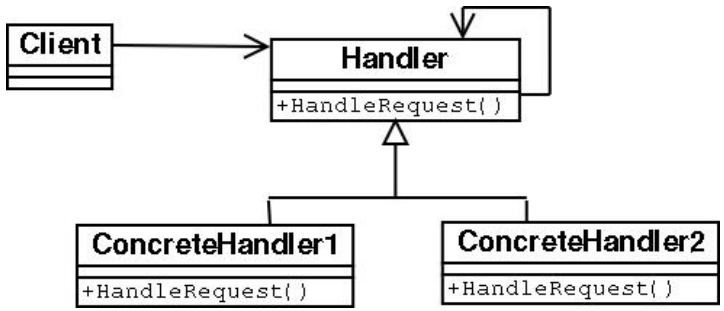
- Zugriff auf Request und/oder Response Pfad bei einem Servlet Aufruf, zum Beispiel für
  - Authentication
  - Auditing
  - Image conversion
  - Data compression
- Ausführungsreihenfolge
  - siehe Filter Mapping
- Modifikation von Request/Response
  - ResponseWrapper/RequestWrapper (siehe Übung 2)

Design Patterns hinter dem Konzept "Filter"

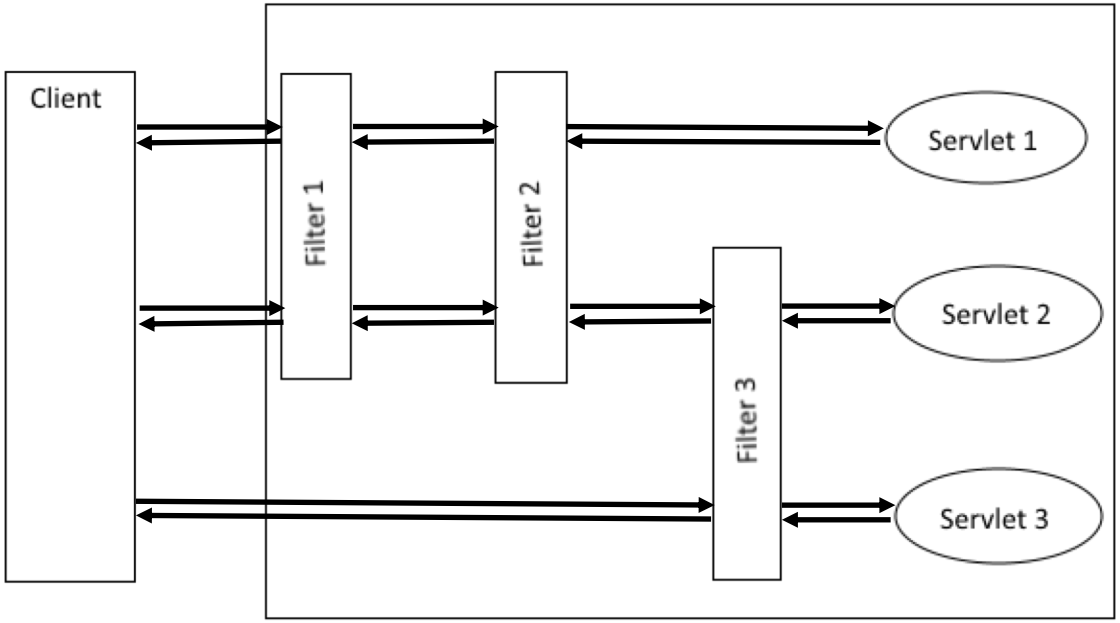
- **Interceptor Pattern**
  - Interceptors sollen technische Dienste bereitstellen, die nicht direkt mit der Anwendungslogik zu tun haben.



- **Chain of Responsibility Pattern**
  - Der Auslöser und der Verarbeiter einer Nachricht werden entkoppelt. Es wird dabei eine Kette von Objekten durchlaufen, welche die Nachricht verarbeiten können. Die Nachricht wird solange weitergereicht, bis ein Objekt diese verarbeitet hat oder das Ende der Kette erreicht ist.



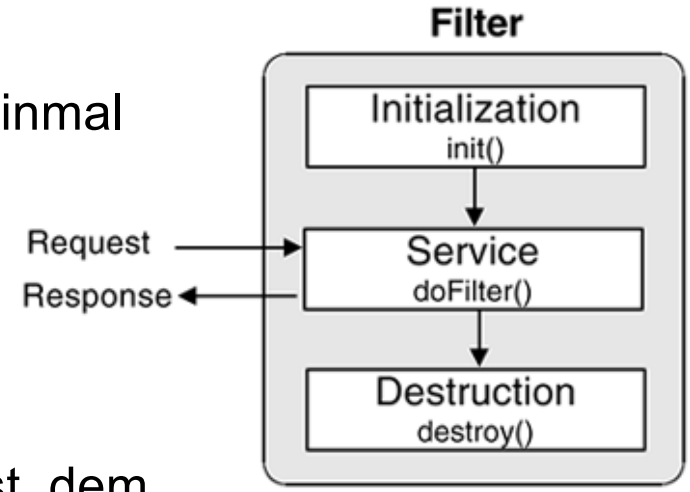
Filter Mapping



- Filter-Reihenfolge bestimmt über
  - Mapping-Eintrag im web.xml
  - ... und bei Annotationen?
    - Reihenfolge kann NICHT über Annotation beschrieben werden.
      → **Eintrag "filter-mapping" wird notwendig bleiben**

javax.servlet.Filter

- init(javax.servlet.FilterConfig config)
  - Bei der Initialisierung des Filters, wird init() einmal aufgerufen



- **doFilter(ServletRequest request, ServletResponse response, FilterChain filterChain)**
  - Der Servlet-Container ruft bei jedem Request, dem ein Filter zugeordnet ist, die Methode doFilter() auf.

- destroy()
  - Wenn ein Filter aus dem Servlet-Container entfernt wird, wird die Methode destroy() aufgerufen und der Filter kann allfällige Aufräumarbeiten erledigen.

Praktische Übung "Filter"

- Arbeitsblatt 3, Aufgabe 1
  - Lesen sie die Theorie zu Filter (Seite 4)
  - Bemerkungen:
    - doFilter(ServletRequest request, ...)
      Der Parameter "request" ist in einer Webapplikation vom Typ "**HttpServletRequest**". Dieser Typ bietet verschiedene Methoden, die im Type "ServletRequest" nicht sichtbar sind
      → Cast auf "HttpServletRequest" ist sinnvoll
    - **NICHT VERGESSEN**
      - chain.doFilter()
      - Filter Mapping

- Besprechung

Listener

- Reaktion auf verschiedene Ereignisse des Servlet-Containers möglich:
  - Servlet Context
  - Servlet Context Attribute
  - Http Session Attribute
  - Http Session Binding

Praktische Übung "Listener"

- Arbeitsblatt 3, Aufgabe 2
  - Lesen sie die Theorie zu "Listener" (Seite 5)
  - Bemerkungen:
    - Verwenden sie die Klasse "ServletContextListener"
    - Über "ServletContextEvent" (Input-Parameter der init-Methode) haben sie mit der Methode "getServletContext()" Zugriff auf den ServletContext, der auch im Servlet selber sichtbar ist.
    - **NICHT VERGESSEN**
      - Eintrag in web.xml oder Annotation
      - Anpassung in "BasicServlet" vornehmen

- Besprechung

Übung 2 im Selbststudium: I18NFilter

- **Ergänzungen**
  - **WICHTIG:**
    Es ist auf **dem Outbound nicht möglich den Original Response direkt zu manipulieren**, da mit dem Abschluss der service()-Methode im Servlet der Servlet-Container implizit ein flush() & close() auf den OutputStream aufruft.

  - Somit ist eine Modifikation ausgeschlossen - und deshalb kommen die folgenden Wrapper Klassen in Spiel:
    - HttpServletResponseWrapper
      Provides a convenient implementation of the HttpServletResponse interface that can be subclassed by developers wishing to adapt the **response from a Servlet.**
    - HttpServletRequestWrapper
      Provides a convenient implementation of the HttpServletRequest interface that can be subclassed by developers wishing to adapt the **request to a Servlet**.