Webanwendungen Grundlagen

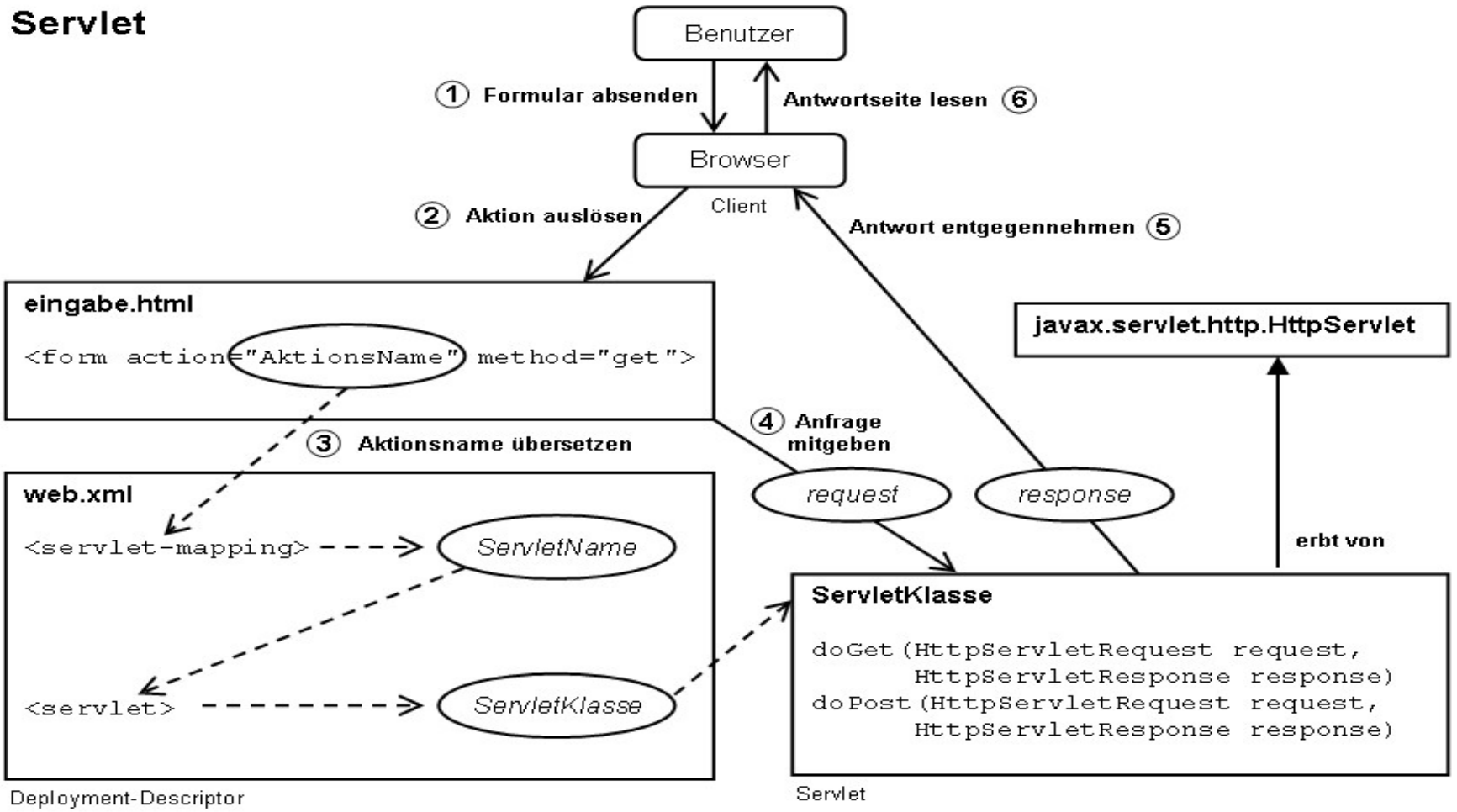
Session Tracking

Prof. DI Dr. Erich Gams
htl-wels.e.gams@eduhi.at

Übersicht ☛ Was lernen wir?

- Servlets Lebenszyklus
- Webanwendungen
- Sessions in Java
- Sessions API
- Beispiele

WH Kommunikation



Lebenszyklus eines Servlets

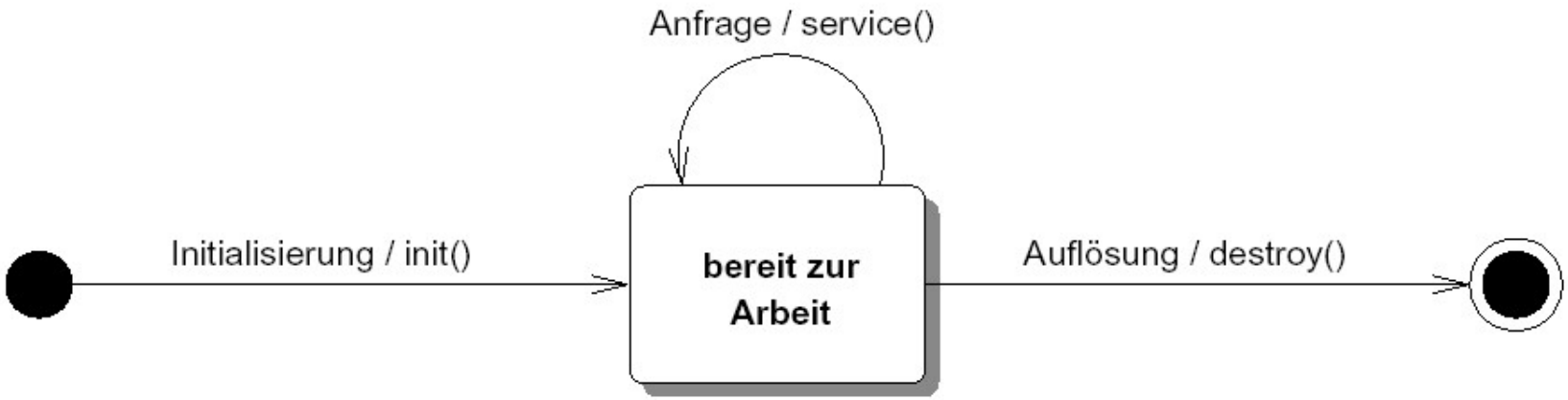


Abbildung 5.3: Lebenszyklus von Servlets

Lebenszyklus eines Servlets (2)

- Der Lebenszyklus eines Servlets im Detail:
  - Laden und instanziieren
    - Entweder beim Start des Servlet-Containers oder bei der ersten Anfrage
  - Initialisieren
    - Die *init*-Methode des Servlets wird aufgerufen
    - Hier kann das Servlet Initialisierungsaufgaben erledigen, z.B. eine Datenbankverbindung herstellen oder Konfigurationsdaten aus einer Datei einlesen

Lebenszyklus eines Servlets (3)

- Client-Anfragen bearbeiten
  - Die *service*-Methode des Servlets wird aufgerufen
  - Diese Methode überprüft den HTTP-Anfragetyp und leitet die Anfrage an die richtige Methode weiter, z.B. *doGet*, *doPost*
  - Die Methoden können auf ein Objekt von Typ *ServletRequest* zugreifen, das Zugriff auf alle Daten der Anfrage ermöglicht
  - Analog haben die Methoden Zugriff auf ein Objekt vom Typ *ServletResponse*, mit dessen Hilfe die Antwort an den Client geschickt wird
  - Bei einem HTTP-Servlet sind die Objekte vom Typ *HttpServletRequest* bzw. *HttpServletResponse*
- Servlet-Klasse wieder entladen
  - Der Servlet Container entscheidet, wann die Servlet-Instanz wieder aus dem Speicher entfernt wird
  - Vorher wird die Methode *destroy* aufgerufen

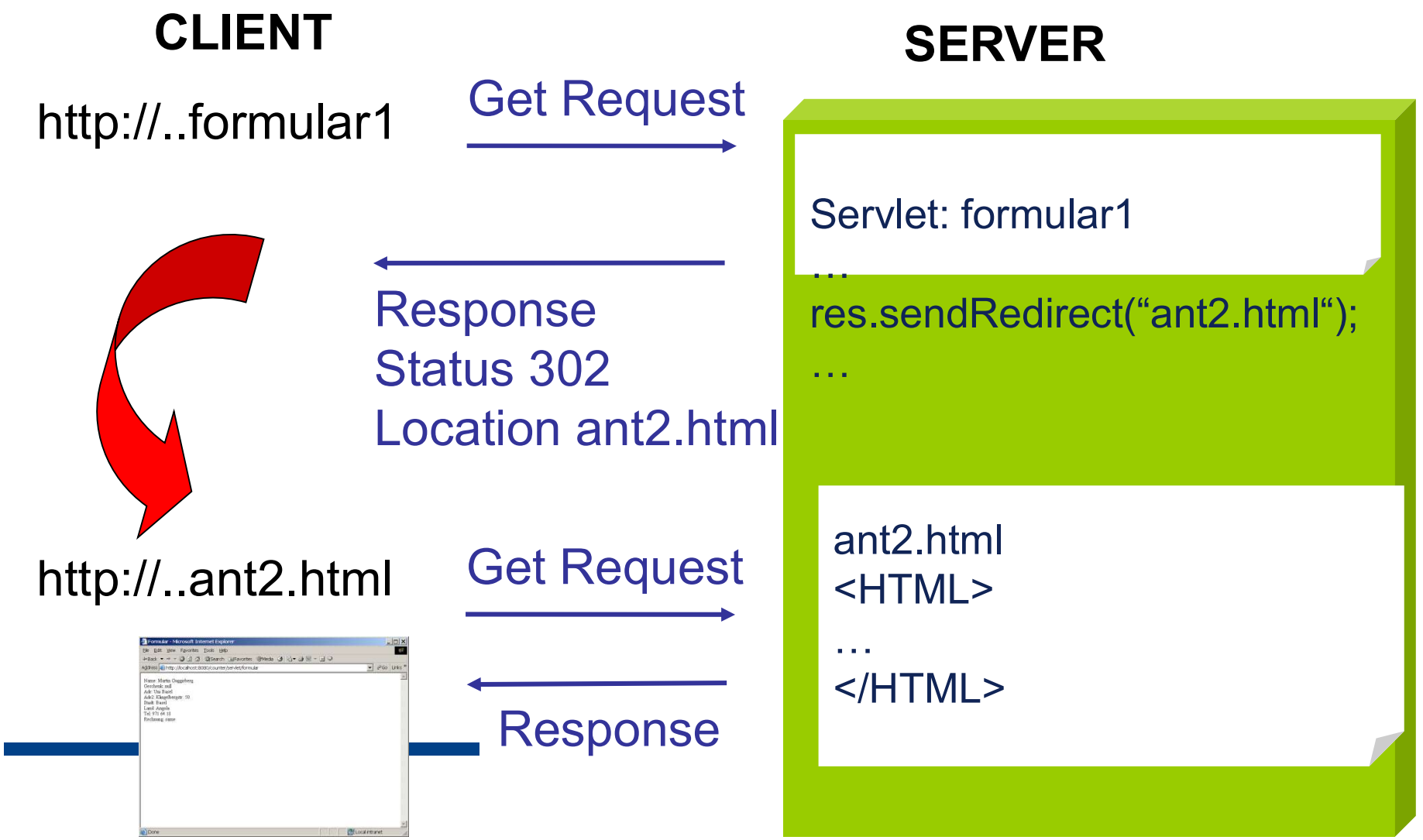
Redirect

```
@WebServlet("/wrong-destination")
public class WrongDestination extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
            String userAgent = request.getHeader("User-Agent");

            if ((userAgent != null) && (userAgent.contains("MSIE")) {
                    response.sendRedirect("http://home.mozilla.com");
            } else {
            response.sendRedirect("http://www.microsoft.com");
            }
    }
}
```

Redirect


CLIENT
SERVER
http://..formular1
Get Request
Servlet: formular1
…
res.sendRedirect(“ant2.html“);
…
Response
Status 302
Location ant2.html
http://..ant2.html
Get Request
ant2.html
<HTML>
…
</HTML>
Response

HTTP

- HTTP ist ein "zustandsloses" Protokoll, d.h. jedes Mal, wenn ein Client eine Webseite aufruft, öffnet der Client eine separate Verbindung zum Webserver.
- Der Server speichert automatisch keine Aufzeichnungen über vorherige Client-Requests.

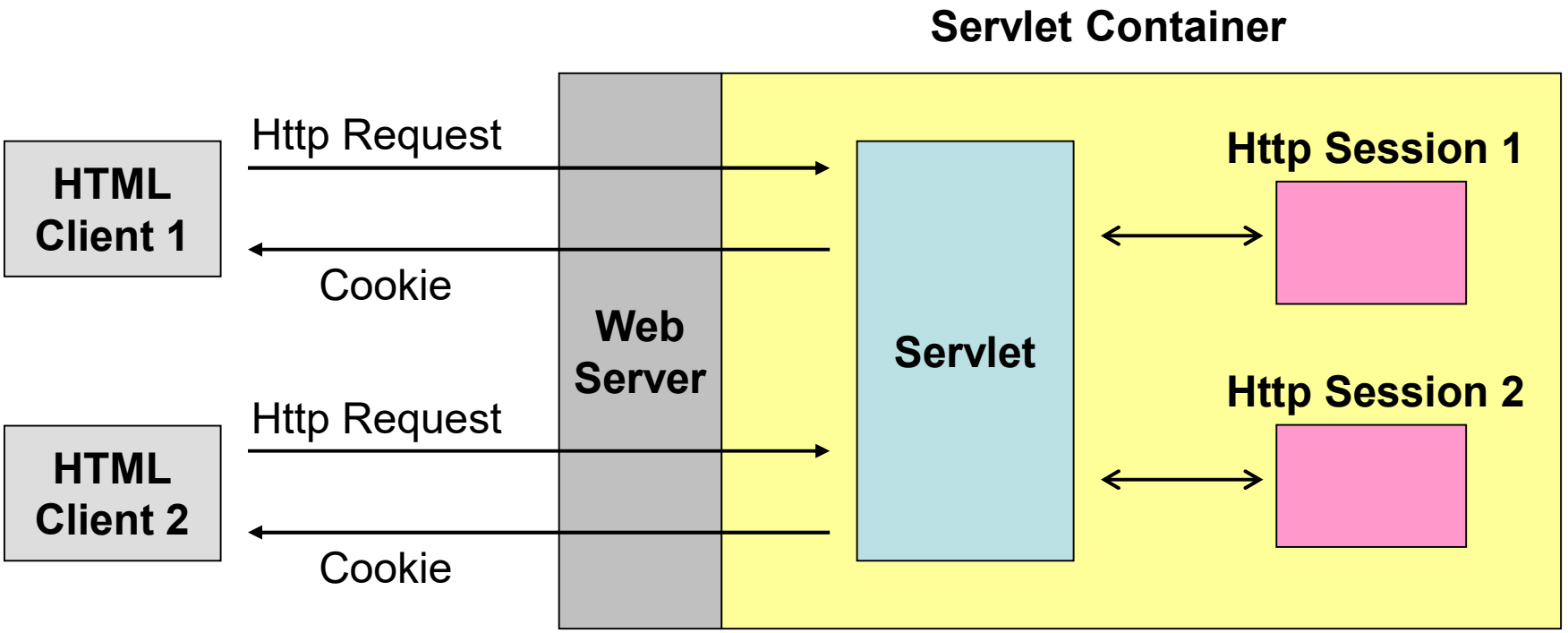
Webanwendungen

- Problem:
  - Viele Benutzer kommunizieren gleichzeitig mit dem Server.
  - Welche Requests gehören zu welchen Benutzern?

What is a Session?

- A session is a state associated with particular user that is maintained at the server side
- Sessions between the HTTP requests
- Sessions enable creating applications that depend on individual user data.

Session Tracking



Sessions in Servlets

- Servlets include a built-in Sessions API
  - Sessions are maintained automatically, with no additional coding.
  - The Web container associates a unique **`HttpSession`** object to each different client.
  - Different clients have different session objects at the server.
  - Requests from the same client have the same session object.
  - Sessions can store various data.

The Sessions API

- The sessions API allows
  - To get the **`HttpSession`** object from the `HTTPServletRequest` object
  - Extract data from the user's session object
  - Append data to the user's session object
  - Extract meta-information about the session object,
    - e.g. when was the session created

The Sessions API - Getting The Session Object

- To get the session object use the method

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

- If the user already has a session, the existing session is returned
- If no session still exists, a new one is created and returned
- If you want to know if this is a new session, call the `isNew()` method

Behind The Scenes

- When you call `getSession()` each user is automatically assigned a unique Session ID
- How does this Session ID get to the user?
  - **Option 1**:
    If the browser supports cookies, the servlet will automatically create a session cookie, and store the session ID within the cookie
    - In Tomcat, the cookie is called `JSESSIONID`

Behind The Scenes

- **Option 2**: URL-Rewriting:
  Client hängt die Session Daten an die URL an.
  - simpel und funktioniert bei allen Browsern unschön, da die Kennungen zu sehen sind, (werden in Bookmarks übernommen, Verlinkung von anderen Seiten)
  - Müssen dynamisch generiert werden
  - http://host/path/file.html;jsessionid=1234

Behind The Scenes

- **Option 2**: Hidden Form Fields: Idee:
  - <INPUT TYPE="HIDDEN" NAME="session" VALUE="...">
- Information müssen immer dynamisch mit eingebaut werden.
- Alle Seiten müssen Forms beinhalten

Extracting Data From The Session

- The session object works like a `HashMap`
  - Enables storing any type of Java object
  - Objects are stored by key (like in hash tables)
- Extracting existing object:

```
Integer accessCount =
        (Integer) session.getAttribute("accessCount");
```

- Getting a list of all "keys" associated with the session

```
Enumeration attributes = request.getAttributeNames();
```

Storing Data In The Session

- We can store data in the session object for using it later.

```
HttpSession session = request.getSession();
session.setAttribute("name", "SE 432");
```

- Objects in the session can be removed when not needed more.

```
session.removeAttribute("name");
```

Getting Additional Session Information

- Getting the unique session ID associated with this user, e.g. `gj9xswvw9p`

```
public String getId();
```

- Checking if the session was just created

```
public boolean isNew();
```

- Checking when the session was first created

```
public long getCreationTime();
```

- Checking when the session was last active

```
public long getLastAccessedTime();
```

Session Timeout

- We can get the maximal session validity interval (in seconds)

```
public int getMaxInactiveInterval();
```

  - After such interval of inactivity the session is automatically invalidated

```
public void setMaxInactiveInterval (int seconds);
```

- We can modify the maximal inactivity interval
  - A negative value specifies that the session should never time out

Terminating Sessions

- To terminate session manually use the method:

```
public void invalidate();
```

  - Typically done during the "user logout"
- The session can become invalid not only manually
  - Sessions can expire automatically due to inactivity.
  - session.setMaxInactiveInterval(120);
  - Im DD:

```
<session-config>
        <session-timeout>120</session-timeout>
</session-config>
```

Session Tracking

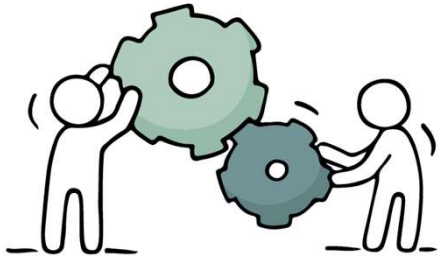
```
public void doPost (HttpServletRequest req, HttpServletResponse res)
                                        throws ServletException, IOException {
    . . .
    HttpSession session = req.getSession(true);
    . . .
    // schreibender Zugriff auf das Session-Objekt
    session.setAttribute("warenkorb", myWarenkorb);
    . . .
    // lesender Zugriff auf das Session-Objekt
    myWarenkorb = (Vector) session.getAttribute("warenkorb");
    . . .
    // Eintrag entfernen
    session.removeAttribute("warenkorb");
    }
```

Sichtbarkeitsobjekte

- Web-Anwendung
  → javax.servlet.ServletContext
- Client-Sitzung
  → javax.servlet.http.HttpSession
- Request
  → javax.servlet.ServletRequest

Methoden:

- java.util.Enumeration getAttributeNames()
- void setAttribute(String name, Object object)
- Object getAttribute(String name)
- void removeAttribute(String name)



Servlet Context

- Allows servlets that belong to the same application to share data and communicate
  - For instance, can be used to store details for a JDBC connection (details that are shared by different servlets)
- Allows setting and getting attributes
- Provides information about the server
  - Can be used for writing messages to a log file
  - getServletContext().getAttribute()
  - getServletContext().setAttribute()

Servlets

Arten von "Servlet-Variablen"

Art der Variable	Sichtbarkeit	Lebensdauer
Attribute eines **ServletContext**	Alle Servlets im ServletContext	Bis zum Ende der Web-Applikation
Attribute einer **Session**	Servlets im ServletContext, die Requests der Session behandeln	Bis zum Ablauf der Session
Instanzvariablen eines **Servlet**	Methoden des Servlet	Servlet.init() bis Servlet.destroy()
Attribute eines **ServletRequest**	Servlets, die den Request behandeln	Bis zum Ende der Request-Behandlung

Kommunikation zwischen Servlets

In **doGet** oder **doPost**:

```
ServletContext context = getServletContext();
. . .
RequestDispatcher dispatcher =
        context.getRequestDispatcher("/myServlets.Servlet1");
. . .
//evtl.
req.setAttribute(name, value);
. . .
dispatcher.include(req, res);
// oder
dispatcher.forward(req, res);
```

Request Dispatcher

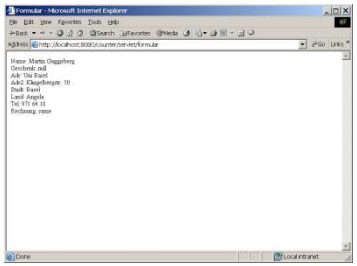
- forward – Die Kontrolle wird vollständig an ein anderes Servlet abgegeben (Es dürfen noch keine Daten gesendet worden sein)
- include – Lediglich das Resultat eines weiteren Servlets wird eingebettet ("wrapping")

Request Dispatcher

CLIENT

http://..formular1

Get Request



Response

SERVER

Servlet: formular1
…
res.sendRedirect(“ant2.html“);
…

ant2.html
<HTML>
…
</HTML>

Beispiel: `forward()` und `include()`

```
public class PresentationServlet extends HttpServlet {
   protected void doGet(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException {
     String item = req.getParameter("item");
     if(item == null) {
       req.setAttibute("exception", new Exception("Item not found"));
       getServletContext()
         .getRequestDispatcher("/servlet/ErrorServlet")
         .forward(req, res);
     } else {
       res.setContentType("text/html");
       PrintWriter out = res.getWriter();
       out.print("<HTML><HEAD><TITLE>Item " + item + "</TITLE>"+
           "</HEAD><BODY>Item " + item + ":<P>");
       getServletContext()
         .getRequestDispatcher("/servlet/ItemServlet?item="+item)
         .include(req, res);
       out.print("</BODY></HTML>"); } } }
```

Any requests?

Aufgabe 1

- Implementiere ein Servlet, welches diverse Informationen über den Request und den Container anzeigen soll.
  - Request Information: Req. URL, Http-Header, Req. Parameter, Client IP und Host, Datum und Zeit des Request, Request Methode (POST oder GET), QueryString bei Methode GET, Request Schema und Protokoll
  - Server und Servlet Informationen: Servername, Serverport, Servlet Pfad, Echter Pfad mit Context, Servlet Context Name, Container Name, Servlet Version.
- Das Servlet soll eine Startseite mit 2 Links beinhalten, welche die beiden oben beschriebenen Information abrufen.
- Rücklinks auf die Startseite sollen auch gesetzt werden!

Aufgabe 2 : Werbe-Banner

- Entwickeln Sie ein Servlet, das bei jedem Aufruf ein neues Bild aus einem Verzeichnis des Web-Servers zurückliefert.
- Implementiere auf diese Art und Weise einen einfachen Werbebanner-Mechanismus.
- **Zum Bsp.:** Am Web-Server liegen folgende Bilder:
  - img01.jpg
  - img02.jpg
  - img03.jpg
- Beispiel HTML-Seite:

```
<html><head><title>Banner Test</title></head>
<body><img src="/myWebapp/imgbanner?basedir=c:\temp\images"></body>
</html>
```

- Beim ersten Aufruf dieser HTML-Seite soll das Bild "img01.jpg" angezeigt werden, danach "img02.jpg", dann "img03.jpg" und dann wieder "img01.jpg" usw. Das Verzeichnis der Bilder wird über einen Request Parameter angegeben.
- Verwende die Session API um den „Zustand“ des Clients zu speichern, sodass für jeden Client die Reihenfolge eingehalten wird.
- Die Bilder sollen gleich groß im Footer dargestellt werden.

Aufgabe 3 : Black Jack

- Implementiere ein Black Jack Spiel, das der Benutzer gegen den Computer spielt.
- Profis stellen die Karten grafisch am Bildschirm dar.

Quellenverzeichnis

- https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/
- https://www.petrikainulainen.net/software-development/design/understanding-spring-web-application-architecture-the-classic-way/
- Infos:
- https://www.codejava.net/java-ee/servlet/webservlet-annotation-examples