Sitzungszustand

- Gruppierung von Anfragen zu Sitzungen (*Sessions*)
- Klasse `HttpServletRequest`
  Methode `HttpSession getSession (bool create)`
  liefert aktuelle Sitzungsobjekt
- Zustand lokal zur Anwendung (`ServletContext`)
- Implementierung (transparent)
  - URL-Rewriting:
    Anhängen eines Parameters `;jsessionid=123123`
  - Cookies `JSESSIONID`
  - SSL Sessions

Beispiel: NumberGuess

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class NumberGuess extends HttpServlet {
  public void doGet (HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException
{
    HttpSession session = request.getSession (true);
    response.setContentType ("text/html; charset=utf-8");
    PrintWriter out = response.getWriter ();
    out.println ("<html><head><title>Number Guess</title></head>");
    out.println ("<body><h3>Number Guess</h3>");
    String guessString = request.getParameter ("guess");
```

Beispiel: NumberGuess

```
    if (guessString == null) {
        long n = Math.round (Math.random () * 100);
        session.setAttribute ("SN", new Long (n));
        out.println ("I am thinking of a number from 1-100");
    } else {
        long guess = Long.parseLong (guessString);
        long n = ((Long)session.getAttribute ("SN")).longValue ();
        if (guess == n) {
            out.print ("You got it!");
        } else if (guess > n) {
            out.println ("Try lower");
        } else {
          out.println ("Try higher");
        }
    }
    String uri = request.getRequestURI ();
    out.println ("<form action =\"" + response.encodeURL (uri) + "\" method=\"get\">");
    out.println ("<input type=\"text\" name=\"guess\"/>");
    out.println ("<input type=\"submit\" value=\"Make A Guess\"/>");
    out.println ("</form></body></html>");
  }
}
```

Wichtige Methoden der Klasse `HttpSession`

- `Enumeration getAttributeNames ()`
- `Object getAttribute (String name)`
- `void setAttribute (String name, Object value)`
- `int getMaxInactiveInterval ()`
- `void setMaxInactiveInterval (int seconds)`
- `long getLastAccessedTime ()`
- `boolean isNew ()`
- `void invalidate ()`

Klientenzustand mit Cookies

- `void HttpServletResponse.addCookie (Cookie cookie)`
- `Cookie[] HttpServletRequest.getCookies ()`
- Klasse `Cookie`
  - Konstruktor `Cookie (String name, String value)`
  - `String Cookie.getName ()`
  - `void Cookie.getName (String name)`
  - `String Cookie.getValue ()`
  - `void Cookie.setValue (String value)`

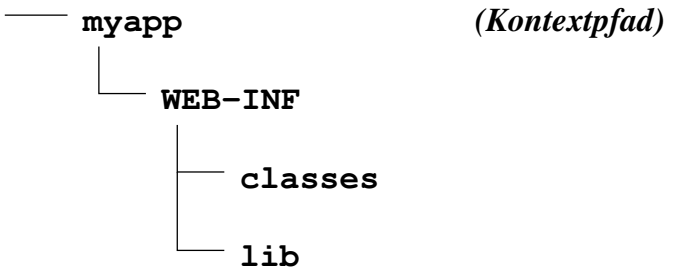
Probleme mit Servlets

- Kontrollfluss ist nicht offensichtlich (siehe `NumberGuess`)
- Referenzen zwischen Servlets durch Strings (`action` Attribute)
- Kommunikation zwischen Formular und Servlet durch Strings

9.4.6 Webanwendungen

- können bestehen aus
  - einer Menge von Servlets
  - XHTML Dokumenten
  - Stylesheets
  - Bildern, etc.
- abgelegt in standardisierter Verzeichnisstruktur

Verzeichnisstruktur einer Webanwendung



- `myapp/` (und alle Unterverzeichnisse ausser `WEB-INF`)
  statische Ressourcen (XHTML Seiten, Bilder, Stylesheets)
- `myapp/WEB-INF/`
  Deployment Descriptor in Datei `web.xml` (s.u.)
- `myapp/WEB-INF/classes/`
  `class` Dateien von Servlets und Hilfsklassen
  in Verzeichnisstruktur entsprechend den Paketnamen
- `myapp/WEB-INF/lib/` `jar` Dateien, die in den CLASSPATH der Anwendung eingebaut werden
- oder gebündelt in *Web Archive* (`war` Datei)

Deployment Descriptor

- Erforderlich in jeder Web Anwendung
- Konfigurationsdaten
  - Abbildung von URI-Pfaden auf Ressourcen der Anwendung
  - Initialisierungsparameter
    verfügbar über `String`
    `GenericServlet.getInitParameter (String name)`
  - Fehlerbehandlung
  - Zugriffsbeschränkungen und Authentisierung
  - Registrierung von Listeners und Filters
- deklarativ, getrennt vom Kode

Beispiel für Deployment Descriptor

```
<?xml version="1.0" encoding="utf-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee" version="2.4">
    <display-name>Echo Your Parameters</display-name>
    <servlet>
        <servlet-name>MyFirstServlet</servlet-name>
        <servlet-class>Echo</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>MyFirstServlet</servlet-name>
        <url-pattern>/echo/*</url-pattern>
    </servlet-mapping>
</web-app>
```

Erklärung zum Deployment Descriptor

- `display-name` Kurzname der Anwendung
- mehrere `servlet` und `servlet-mapping` Elemente möglich
- `servlet`: Name $\longrightarrow$ voll qualifizierte Klasse
  - Klasse darf mehrfach vorkommen
  - jeweils eine Instanz
- `servlet-mapping`: URI Muster $\longrightarrow$ Name
  - Muster relativ zum Kontextpfad
  - im Beispiel: jede URL mit Pfadpräfix `myapp/echo` startet `Echo`

Initialisierungsparameter

- Kontextparameter
  - gesamte Anwendung
  - `String ServletContext.getInitParameter (String)`
- Servletparameter
  - einzelne Servlets
  - `String GenericServlet.getInitParameter (String)`
- durch Servletparameter können mehrere Instanzen des gleichen Servlets unterschiedlich initialisiert werden

Beispiel Initialisierungsparameter

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee" version="2.4">
  <display-name>A Small Web Application</display-name>
  <context-param>
    <param-name>admin</param-name>
    <param-value>john.doe@widget.inc</param-value>
  </context-param>
  <servlet>
    <servlet-name>MyFirstServlet</servlet-name>
    <servlet-class>HelloWorld</servlet-class>
    <init-param>
      <param-name>verbose</param-name>
      <param-value>true</param-value>
    </init-param>
  </servlet> <servlet-mapping /> </web-app>
```

Beispiel Fehlerbehandlung

```
<web-app ...>
  ...
  <error-page>
    <error-code>404</error-code>
    <location>/not_found.html</location>
  </error-page>
  <error-page>
    <exception-type>inc.widget.NotImplementedException</exception-type
    <location>/error</location>
  </error-page>
  ...
</web-app>
```

9.4.7 Servlets in Tomcat

- Apache Jakarta Tomcat
- Referenzimplementierung des Servlet Framework
- Verzeichnisstruktur
  - `common/lib/servlet-api.jar`
    zum Übersetzen von Servlets
  - `bin/`
    Skripte zum Starten und Stoppen des Servers (`startup.sh`, `shutdown.sh`)
  - `conf/`
    Konfigurationsdateien des Servers
  - `webapps/`
    übergeordnetes Verzeichnis aller Webanwendungen

Tomcat Infos

- HTTP-Service über Port 8080
- `manager` Anwendung hilft beim Deployment und Update von Anwendungen ohne den Server zu stoppen.
  Konfiguration:
  - `conf/tomcat-users.xml`
  - Benötigt wird ein Benutzer mit Rolle `manager`:

```
<role rolename="manager"/>
<user username="jdoe"
      password="qwerty"
      roles="manager"/>
```

- Starten des Servers: `bin/startup.sh`
- Hinzufügen einer Webanwendung
  - Kopieren der Verzeichnisstruktur nach `webapps/myapp`
  - Aktivieren durch den `manager`
    `http://localhost:8080/manager/start?path=/myapp`
    dann Eingabe von Benutzername und Passwort
  - Auffrischen von `class` und `jar` Dateien
    `http://localhost:8080/manager/reload?path=/myapp`