FH Bielefeld
University of
Applied Sciences

Campus Minden

Webbasierte Anwendungen
SS 2015

Grundlagen Servertechnologien

Dozentin: Grit Behrens
mailto:grit.behrens@fh-bielefeld.de

Studiengang Informatik Fachbereich Technik

Lehrinhaltsübersicht der Vorlesungen zu WBA

1. Einführung in WBA
2. Wiederholung: Grundlagen des WWW, HTML und HTTP
3. Clientseitige Implementierungstechnologien: Javascript, DOM, Ajax, (Java-Applet)
4. **Serverseitige Implementierungstechnologien:** JSP, Java-Servlet
5. Anbindung von Datenbanken
6. WEB-Frameworks: JSF

Einführung in serverseitige Implementierungstechnologien

1. HTTP – Leistungsmerkmale
2. Session – Management

Einführung in serverseitige Implementierungstechnologien

*Unter Nutzung serverseitiger Technologien implementiert der Web-Entwickler Programme, die unter **Kontrolle eines Web - Servers ausgeführt** werden und deren Ergebnis, zumeist **dynamisch generierte HTML** oder XML- Dokumente, per **HTTP- Response** an den aufrufenden Client gesendet werden.*

häufig eingesetzte Technologien:

- Java- Servlets und JSP
- PHP
- ASP (Active Server Pages)

Einführung in serverseitige Implementierungstechnologien

1. **HTTP – Leistungsmerkmale**
    1. Authentifizierung
    2. Server – Redirection
    3. HTTP – Caching
2. Session – Management

HTTP- Leistungsmerkmale

Auf der Grundlage von HTTP- Headern und Statuscodes sind viele Leistungsmerkmale implementiert, z.B.

- ***Authentifizierung***
- ***Redirection***
- ***Caching***

1. Authentifizierung:

Ziel der Authentifizierung ist die **Identifizierung des Benutzers**.

Varianten zur Übertragung der Authentifizierungsinformationen:

- HTTP Basic
- HTTP Digest
- Formularbasierte Authentifizierung
- HTTPS Client - Authentifizierung

Einführung in serverseitige Implementierungstechnologien

1. **HTTP – Leistungsmerkmale**
    1. **Authentifizierung**
    2. Server – Redirection
    3. HTTP – Caching
2. Session – Management

HTTP: Authentifizierung

HTTP – Basic

- seit HTTP 1.0 verfügbar
- Benutzername und Passwort werden unverschlüsselt übertragen
- wenig sichere Datenübertragung

HTTP – Digest Authentifizierung

- seit HTTP 1.1 eingeführt als verbesserte Alternative zu HTTP-Basic
- Passwort nicht an den Server übertragen
- ***Vorgang (symmetrische Verschlüsselungstechnik):***
  - Server generiert Zufallswert
  - clientseitige Verknüpfung des Zufallswertes mit dem Passwort
  - Übertragung der Verknüpfung an den Server

HTTP: Authentifizierung

Formularbasierte Authentifizierung

- Übertragung durch Parameterstrings (HTTP - GET-Befehl) oder Datenpakete (HTTP - POST- Befehl)
- Bei Nutzung des HTTPS (HTTP over SSL) erfolgt gesamter Datentransfer verschlüsselt

HTTPS – Client - Authentifizierung

- beruht ebenfalls auf HTTPS
- Nutzer benötigt ein digitales Zertifikat von einer *Certifying Authoritiy (CA)*
- CA ‘s sind kommerzielle Organisationen
- Digitale Zertifikate enthalten Stammdaten des Zertifikateigners sowie einen öffentlichen Schlüssel
- Zertifikat wird an den Server übertragen und identifiziert den Client eindeutig

HTTPS: Authentifizierung

Praktische SSL-Implementierung (Secure Socket Layer):

Stellt Kombination des symmetrischen und asymmetrischen Verschlüsselungsverfahrens dar mit folgender zugrundeliegender Idee:

1. Server überträgt sein **digitales Zertifikat** und **öffentlichen Schlüssel** an den Browser
2. Browser generiert einen **zufälligen Session-Key**
3. Die Nachricht wird **mit dem Session – Key symmetrisch kodiert**
4. Session – Key wird mit öffentlichem Schlüssel des Servers verschlüsselt (auch als **digitaler Briefumschlag** bezeichnet)
5. Ergebnis aus 3 und 4 werden an den Server übertragen.
6. **Server extrahiert Session – Key** unter Verwendung seines privaten Schlüssels
7. Server **entschlüsselt die Nachricht symmetrisch** unter Verwendung des Session – Keys.

- Einmalig im digitalen Briefumschlag übertragener ***Session – Key*** wird später weiter benutzt für symmetrische Verschlüsselung aller zwischen Browser und Server ausgetauschten Nachrichten

HTTPS: Authentifizierung

Beispielanwendung für HTTP-Basic (steht für alle relevanten Servertechologien zur Verfügung):

1) **GET-Request:**

```
GET /secure_document.html HTTP/1.0
Accept: image/gif, image/jpeg, */*
Accept-charset: iso-8859-1, *, utf-8
Accept-encoding: gzip
Accept-language: en
User-Agent: Mozilla/4.51 [en] (WINNT; I)
```

2) **Server-Antwort mit Status 401 Unauthorized**

```
HTTP/1.1 401 Unauthorized
Date: Mon, 13 Jan 2003 08:35:41 GMT
Server: Apache/1.3.24 (Win32) PHP/4.3.0
WWW-Authenticate: basic realm="geschuetzterBereich"
```

2) **Anwendung öffnet im Browser dann ein Dialogfenster mit der Bitte um Eingabe von Benutzername und Passwort**

3) **GET-Request aus 1 wird noch einmal gesendet mit zusätzlichem Authorization-Header des Clients:**

```
Authorization: Basic aGVpa286d29laHI
```

Bemerkung: Username und Passwort werden im Authorization Header Base64 – kodiert (aber unverschlüsselt) versendet

Einführung in serverseitige Implementierungstechnologien

1. **HTTP – Leistungsmerkmale**
   1. Authentifizierung
   2. **Server – Redirection**
   3. HTTP – Caching
2. Session – Management

HTTP: Server Redirection

Response auf einen HTTP – Request enthält nicht die erfragte Resource, sondern die geänderte URL, unter der die Ressource jetzt abrufbar ist.

Beispielanwendung :

1. **GET-Request:**

```
GET /formular.php?name=behrens&password=unverschluesselt HTTP/1.0
Accept: image/gif, image/jpeg, */*
Accept-charset: iso-8859-1, *, utf-8
Accept-encoding: gzip
Accept-language: en
User-Agent: Mozilla/4.51 [en] (WINNT; I)
```

2. **Server-Antwort mit Status 302 Found**

```
HTTP/1.1 302 Found
Date: Mon, 13 Jan 2003 16:48:49 GMT
Server: Apache/1.3.24 (Win32) PHP/4.3.0
Location: Login.html
```

Bemerkung: Status 302 besagt, dass die Ressource jetzt unter neuer URL gemäß Location - Header abrufbar ist.

3. **Anwendung setzt im Browser Request wie in 1 noch einmal ab mit modifizierter URL:**

```
GET /login.html?name=behrens&password=unverschluesselt HTTP/1.0
```

Einführung in serverseitige Implementierungstechnologien

1. **HTTP – Leistungsmerkmale**
    1. Authentifizierung
    2. Server – Redirection
    3. **HTTP – Caching**
2. Session – Management

HTTP : Caching

Durch einen Client erfragte Ressourcen können in Caches zwischengespeichert werden:

- **Browser – Cache**
- **Proxy – Cache**

Ressourcen, die nur selten oder nur in bestimmten Abständen geändert werden, sollten gecached werden.

Vorteile von Caching:

- ***schnellere Bereitstellung*** von Ressourcen
- ***Reduzierung der notwendigen Bandbreite*** zur Datenübertragung

-> Verbesserung der Nutzbarkeit (Usability) der Webanwendung

Regeln für effizientes Caching einer Serveraplikation:

- überwiegenden Teil ***statischer Ressourcen cachen***
- bei auftretenden Änderungen an den Ressourcen sollte das ***Verfallsdatum möglichst zeitnah*** liegen
- die ***selben Ressourcen immer über die selbe URL*** referenzieren
- ***Nutzung von SSL reduzieren***, da SSL grundsätzlich nicht gecached wird.
- dynamisch erstellte Ressourcen nur dann cachen, wenn sich Inhalt selten ändert

Bemerkung: genaue Konfiguration hängt vom jeweiligen Server ab.

HTTP : Caching

HTTP-Response `Header Expire:`

Wird dieser Response Header an den Client gesendet, wird dieser vom Browser und von dazwischenliegenden Proxies berücksichtigt.

Beispiel:

```
Expires : Sat, 20 Sep 2009 22:29:00 GMT
```

Pragma-Header `(Pragma: no cache):`

nicht immer nützlich, da im HTTP nur für Request - Header spezifiziert, nicht für Response – Header

`Cache – Control` – Header :

- seit HTTP 1.1
- erlaubt differenzierte Festlegungen, wie Caches mit einer Ressource umgehen sollen
- die wichtigsten Werte:

HTTP : Caching

Die wichtigsten Werte des `Cache-Control`-Headers:

Wert	Beschreibung
`no-cache`	Cache muss vor Lieferung einer gecachten Ressource einen *Validierungsrequest* an den Server stellen.
`no-store`	Cache darf die Ressource nicht zwischenspeichern (z.B. vertrauliche Dokumente)
`max-age`	Legt Verfallszeitpunkt fest, mit Angabe der Zeit in Sekunden

Validierungsrequest mit `if-modified-since` Request - Header ist eine bedingte Anfrage an den Server. Der Server liefert die Ressource nur dann, wenn sie aufgrund Ihres Zeitstempels nicht mehr gültig ist.

Einführung in serverseitige Implementierungstechnologien

1. HTTP – Leistungsmerkmale
    1. Authentifizierung
    2. Server – Redirection
    3. HTTP – Caching
2. **Session – Management**

Session- Management

***Eine Session** beschreibt einen Dialog, der sich über **mehrere Requests und Responses** erstreckt.*

***Session Management** befasst sich mit dem **Sammeln und Speichern von Informationen** innerhalb einer Session.*

Techniken zur Implementierung von Sessions:

- Hidden Fields
- URL Rewriting
- Cookies

}

- Speicherung von Zustandsinformationen auf dem Server

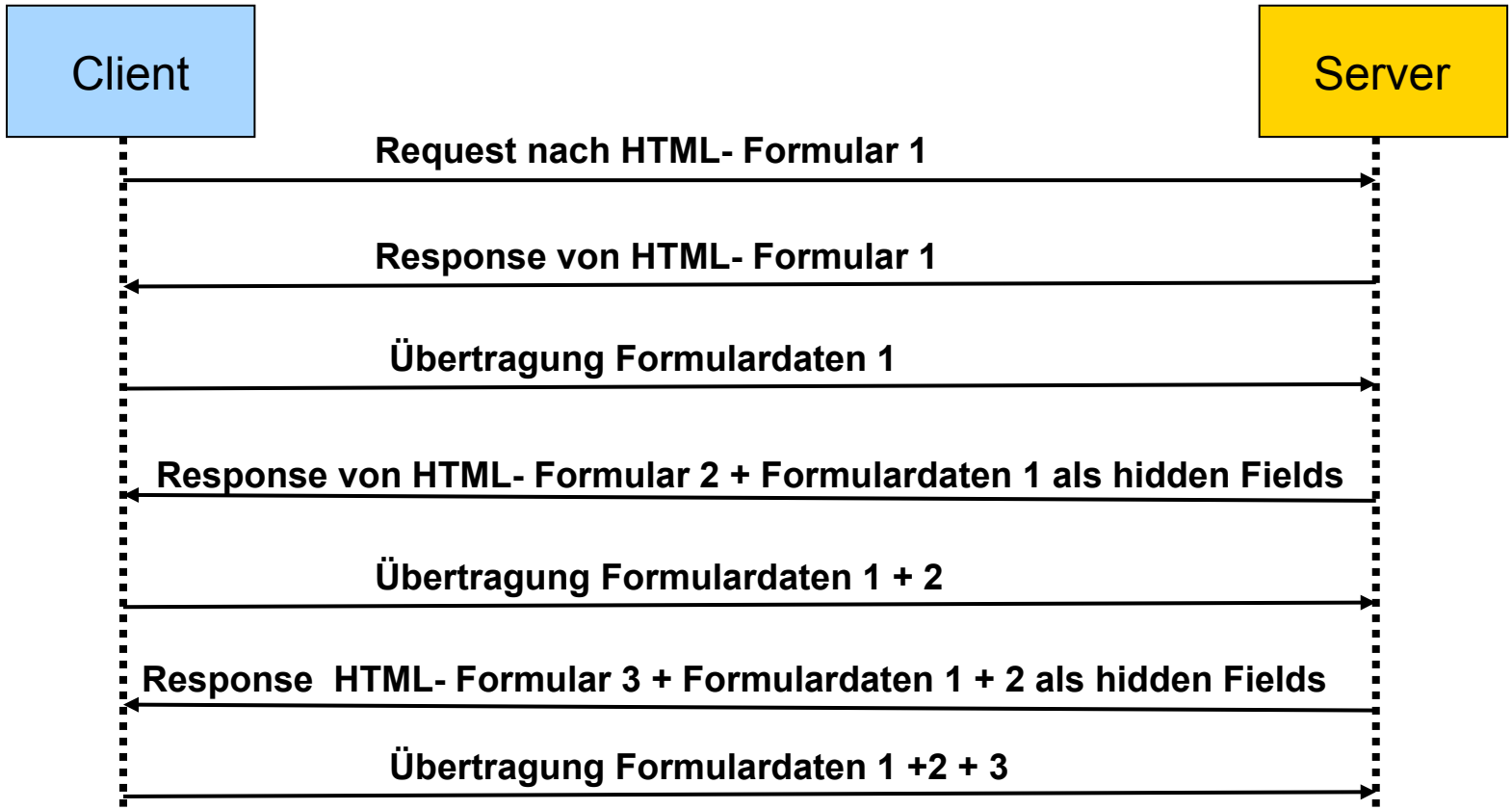
Einführung in serverseitige Implementierungstechnologien

1. HTTP – Leistungsmerkmale
    1. Authentifizierung
    2. Server – Redirection
    3. HTTP – Caching
2. **Session – Management**
    1. **Hidden Fields**
    2. URL Rewriting
    3. Cookies
    4. Speicherung von Zustandsinformationen auf dem Server

Hidden Fields

Hidden Fields sind Elemente eines HTML-Formulars, die `name = "wert"`-Paare an den Server übertragen und die für den Benutzer nicht sichtbar sind.

Beispieldialog für „Füllen eines Warenkorbes“ mit *hidden fields* schematisch:



Hidden Fields

Codefragmente Beispieldialog „Füllen eines Warenkorbes“:

Ausschnitt aus Formular 2 mit dem hidden field `card`

```
<form method = “get“ action = “shop“>
   <!--… -->

   <input type = “hidden“ name = “card“ value = “Artikel_1“>
</form>
```

Ausschnitt aus Formular 3 mit Artikel 1 und 2 als hidden fields `card`

```
<form method = “get“ action = “shop“>
   <!--… -->

   <input type = “hidden“ name = “card“ value = “Artikel_1“>
   <input type = “hidden“ name = “card“ value = “Artikel_2“>
</form>
```

Hidden Fields

Vorteile der Nutzung von Hidden-Fields:

- **Generelle Unterstützung**
- **Client bliebt im Dialog anonym**

Nachteile der Nutzung von Hidden-Fields:

- **Erzwungene Kontinuität aufeinander folgender Formulare**
- **Bei Unterbrechung des Dialogs Verlust der bereits übertragenen Daten**

Einführung in serverseitige Implementierungstechnologien

1. HTTP – Leistungsmerkmale
    1. Authentifizierung
    2. Server – Redirection
    3. HTTP – Caching
2. **Session – Management**
    1. Hidden Fields
    2. **URL Rewriting**
    3. Cookies
    4. Speicherung von Zustandsinformationen auf dem Server

URL Rewriting

Session- Informationen werden beim URL Rewriting als Teil der URLs an den Server übertragen.

- Realisierung in Form von GET-Parametern.
- Jeder Link muss dynamisch um aktuelle Session- Informationen ergänzt werden.

Beispiel:

```
http://www.shoppingExample.com/agb.de?cart=Artikel_1
```

Nachteile:
- Informationen werden offen übertragen -> geringe Datensicherheit
- Abspeichern inaktueller Favoriteneinträge im Browser

Einführung in serverseitige Implementierungstechnologien

1. HTTP – Leistungsmerkmale
    1. Authentifizierung
    2. Server – Redirection
    3. HTTP – Caching
2. **Session – Management**
    1. Hidden Fields
    2. URL Rewriting
    3. **Cookies**
    4. Speicherung von Zustandsinformationen auf dem Server

Cookies

Cookies stellen eine Erweiterung des HTTP-Protokolls dar.

- ***Sie sind durch einen Namen und einen Wert definiert:***

```
name=wert.
```

1) Server überträgt Session – Informationen an den Client durch Nutzen des ***HTTP-Response – Headers:***

```
Set-Cookie: name=wert
```

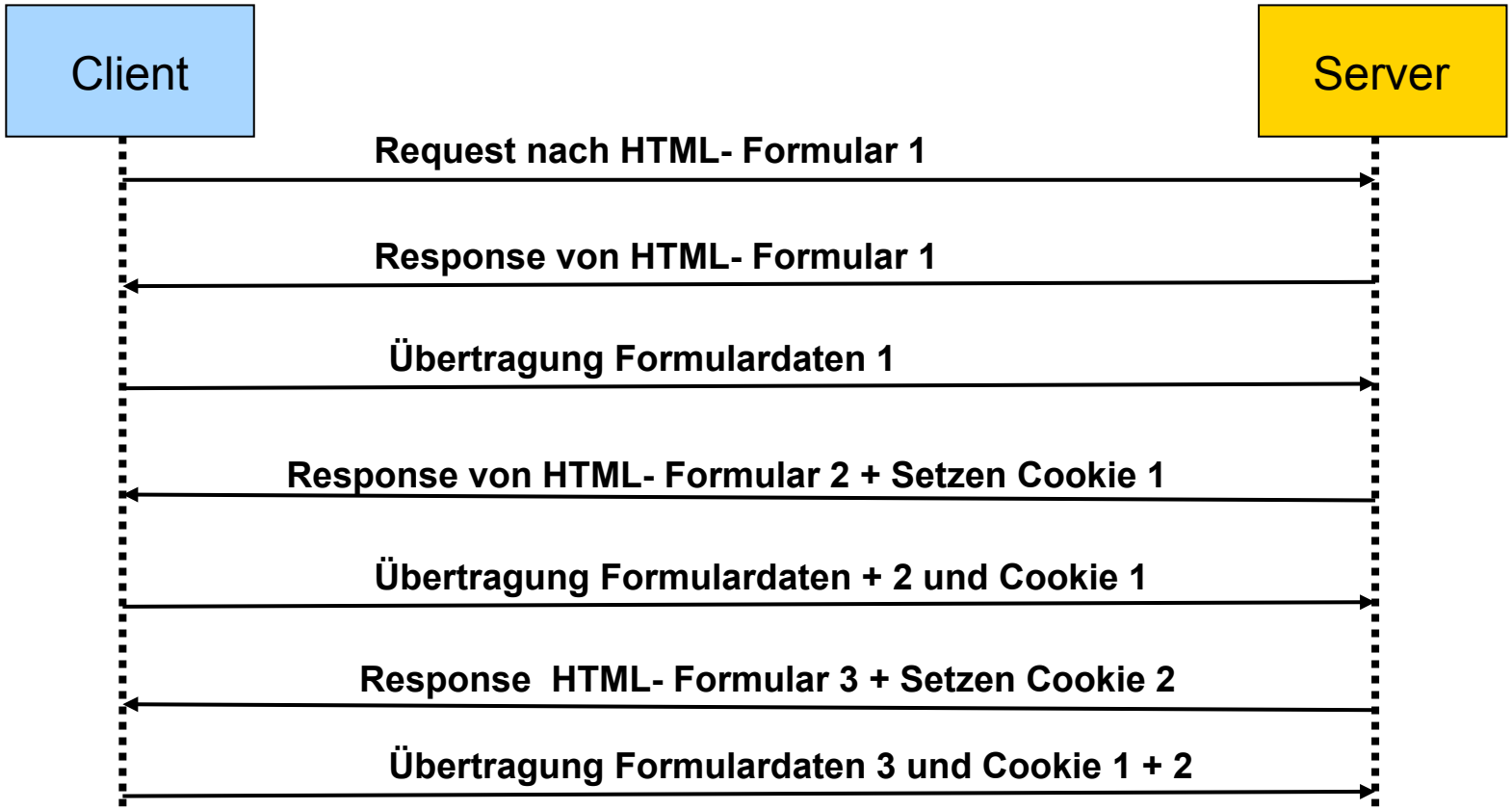
2) werden bei jedem HTTP- Request wieder an den Server übertragen im ***HTTP – Request – Header:***

```
Cookie:name=wert[;name2=wert2]
```

<u>Einschränkungen für Cookies laut HTTP-Protokoll</u>:

- Größe eines Cookies ist auf 4 kB beschränkt.
- Client kann maximal 300 Cookies speichern.
- Client speichert pro Server nur max. 20 Cookies

Cookies


Client
Server
Request nach HTML- Formular 1
Response von HTML- Formular 1
Übertragung Formulardaten 1
Response von HTML- Formular 2 + Setzen Cookie 1
Übertragung Formulardaten + 2 und Cookie 1
Response HTML- Formular 3 + Setzen Cookie 2
Übertragung Formulardaten 3 und Cookie 1 + 2

Cookies

Vorteile der Nutzung von Cookies:

- nicht an HTML gebunden
- Unabhängig von Dialoghistorie
- Bleiben je nach Konfiguration sehr lange erhalten
  -> Session kann nach längerer Zeit wieder aufgenommen werden

Nachteile der Nutzung von Cookies:

- Können vom User unterbunden werden
- Beschränkung in Größe und Anzahl pro Server

Nachteile der clientseitigen Speicherung von Zustandsinformationen

- ***Wiederholter Transport*** **der Zustandsinformationen im Netz**
  -> *Performance* – Verluste
  -> beeinträchtigte *Sicherheit* der Daten

- **Nutzer kann Daten auf dem Client *einsehen und evtl. modifizieren***
  -> beeinträchtigte *Sicherheit der Serverapplikation*

-> Bei großen Webapplikationen werden Session – Informationen auf dem Server gespeichert.

Einführung in serverseitige Implementierungstechnologien

1. HTTP – Leistungsmerkmale
    1. Authentifizierung
    2. Server – Redirection
    3. HTTP – Caching
2. **Session – Management**
    1. Hidden Fields
    2. URL Rewriting
    3. Cookies
    4. **Speicherung von Zustandsinformationen auf dem Server**

Speicherung von Zustands-Informationen auf dem Server

Technologiebeispiel:

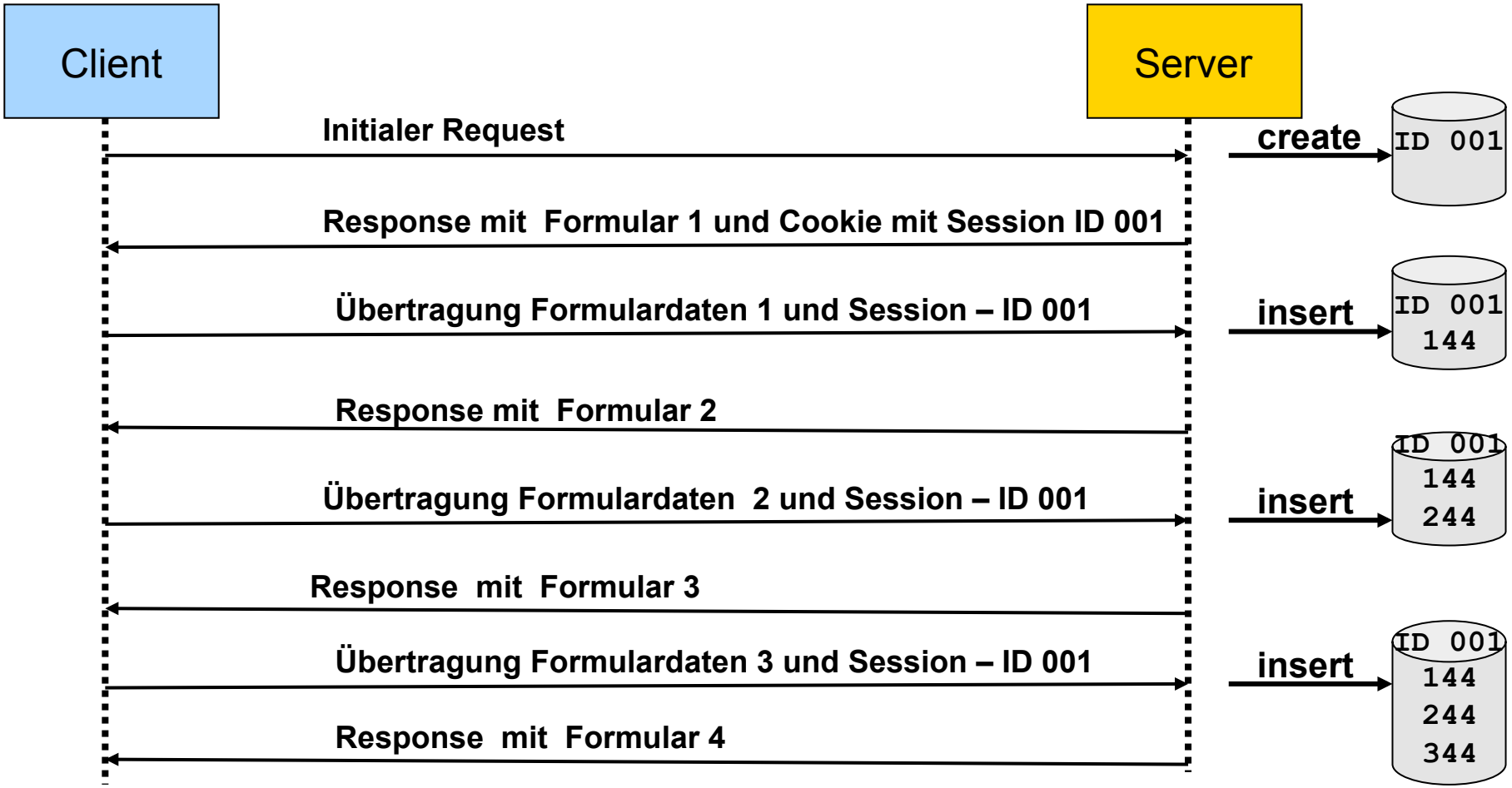
- Server erzeugt für jede Session eine Session – ID
- Speicherung von Zustandsinformationen unter eindeutiger Session – ID
- Session – ID wird einmalig an den Client übertragen mit Cookie, URL-Rewriting oder Hidden Fields
- Dauer einer Session wird begrenzt durch Programmlogik (z.B. Ende eines Bestellvorganges) oder serverseitigen Timeout oder beides

Speichermöglichkeiten der Session – Informationen:

- Hauptspeicher
- Datei
- Datenbank

Die serverseitigen Speichermöglichkeiten sind vielfältig und müssen den Bedürfnissen der Applikation angepasst werden.

Speicherung von Zustands-Informationen auf dem Server



Literatur für Grundlagen serverseitige Implementierungstechnologien

- **Heiko Wöhr „Webtechnologien“ , dpunkt.verlag, Heidelberg 2004**

Ausblick:

Serverseitige Implementierungstechnologien:

Java Server Pages