**Cookies: -** ermöglichen das **Speichern von Key/Value Paaren**

- werden **von der Domäne** **gesetzt** ([www.htlwrn.ac.at](http://www.htlwrn.ac.at)). Bei Anfragen an Domäne **Cookies mitgesendet**.

- Webserver kann aufgrund der mitgesendeten Cookies den **Benutzer wiedererkennen**  
- haben ein **Ablaufdatum**, nachdem der Browser sie löscht

**Design-Ziele verteilter Systeme:** - Ressourcen teilen - Nebenläufigkeit - Transparenz –   
Offenheit (Services werden gemäß Standards angeboten) – Skalierbarkeit - Fehlertoleranz

**Separation of Concerns: Datenkapselung (Information hiding):  
-** Einteilung in Client, Server, Service - Zugriff nur über Schnittstellen **-** Definieren von Schichten (Layering) - Direkter Zugriff auf Implementierung nicht möglich **-** Implementierung aufteilen auf Komponenten  
  
**Schichtenarchitektur, 2-Schichten-Architektur ,Schichten im Java-Web-Backend  
  
DAO Layer: -** steht für Data Access Object - trennt Application Layer von der Datenbank  
- enthält Code für den Datenbank Zugriff

**Datenkapselung über Interfaces:  
Direkter Zugriff** auf Implementierungsklassen wird **vermieden**. **Zugriff erfolgt über Interfaces**.  
**Information Hiding:** Interne Schnittstellen bleiben für Aufrufer der Schnittstelle verborgen.  
**Vorteil:** Die DAO Implementierung kann ausgetauscht werden, ohne sich auf die Serviceschicht auszuwirken.

**Application Layer: -** Enthält **Geschäftslogik -** unabhängig davon, wo und wie die Daten gespeichert werden

**-** unabhängig davon, wie die Daten dem Anwender präsentiert werden

**Web Controller: -** Behandelt die **Kommunikation zwischen Web-Client und Java-Backend   
-** Spezifische Schicht im Falle einer Web-App **-** Gängige Variante ist REST-API

**Middleware und Frameworks**

Typische Geschäftsanwendungen bestehen zu 70-80% aus Applikationsinfrastruktur, die sich von Anwendung   
zu Anwendung kaum unterscheidet. Nur 20-30% Geschäftslogik die spezifisch ist für die jeweilige Anwendung.

**Middleware Services: -** Bietet einfache Kommunikationsmöglichkeiten (z.B. Remote Procedure Call) **-** Unterstützung für Nebenläufigkeit - Erleichtert Skalierbarkeit  **-** Services für Datenbankzugriff  **-** Unterstützung für Replikation **-** Naming Services zum Auffinden der Server  **-** Security: Authentifizierung, Berechtigungen, …

**Logging Frameworks: -** Vorgänge einer laufenden Software sollen für spätere Auswertung protokolliert werden

**-** Mehrere Log-level (Error, Warning, Info, Debug, Trace)

**Dependency Injection: -** Entfernt starre Abhängigkeiten im Programmcode **-** Abhängigkeiten werden vom

Framework zur Laufzeit erkannt und aufgelöst **-** Es können sowohl einfache Datentypen als auch Objekte

“injiziert” werden **-** Über Konfigurationsdatei können die übergebenen Werte und Objekte definiert werden

**Spring Boot: -** Dependency Injection vieler Komponenten **-** Vereinfacht Konfiguration von Applikationen

- Spring Boot sucht automatisch nach Klassen zur Auflösung von Abhängigkeiten  
**Weitere Spring Frameworks:** Spring Data, Spring Security, Spring LDAP, Spring Batch  
  
**Objektrelationale Mapper:** Frameworks zur Ablage von Objekten einer objektorientierten Programmiersprache   
in eine relationale Datenbank. Beispiel: Java Persistence API (JPA)

**GET-Anfrage: -** Hauptbestandteil ist die URL **-** Parameter können mittels „?“ angehängt werden

**GET-Antwort: -** Header enthält Statuscode (z.B. „200 OK“) und Metadaten (z.B. Kodierung mit UTF-8)

- Body enthält die angeforderte Ressource

**HTTP Statuscodes: Jede HTTP-Anfrage** wird vom Server mit einem **HTTP-Statuscode** beantwortet **-** Informiert, ob Anfrage erfolgreich war **-** Im Fehlerfall gibt Code Fehlergrund an **-** Im Fehlerfall ist der Body meistens leer

**HTTP Statuscode Klassen: -** 1xx: Information **-** 2xx: Erfolgreiche Operation **-** 3xx: Umleitung

- 4xx: Client Fehler - 5xx: Server Fehler

**Markup Languages**

**Extensible Markup Language (XML):** XML ist eine Sprache zur Darstellung hierarchisch strukturierter Daten.

**XML-Schema:**

- Exaktere Beschreibung der Struktur von XML-Dokumenten  
- Spezifiziert, welcher Datentyp der Inhalt eines Tags haben darf  
- Konkretes XML-Schema wird als XML-Schema Definition (XSD) bezeichnet  
**XML-Namespace: -** ermöglicht **mischen von Tags aus verschiedenen XML-Schemata.**  
Zu Beginn werden Schema einem Präfix zugeordnet. Tags werden dann mit Präfix annotiert,   
um ihm dem jeweiligen Schema zuzuordnen.  
**SOAP:** - XML-gestütztes Netzwerkprotokoll zum Datenaustausch - Ermöglich Remote Procedure Calls (RPC)  
- Anfragen und Antworten mittels XML-Dokumenten  
**XPATH:** ist eine Abfragesprache für XML-Dokumente. Damit können alle Knoten im Element,   
die zur Abfrage passen ermittelt werden.

**REST Prinzipien:  
- Zustandslosigkeit =** Jede Anfrage eines Clients ist in sich geschlossenAnfrage enthält alles was Server für die Verarbeitung benötigtbegünstig Skalierbarkeit eines WebservicesVoneinander unabhängige Anfragen auf mehrere Server verteilt werden