Web- und Multimedia Engineering

Bedeutung des WWW

- Durchdringung aller Anwendungsbereiche: Geschäftsanwendung + Unterhaltung
- plattformunabhängig
- komplexe verteilte Andwendungssysteme mit UI

W3C:

- Software/ Technologie Standards im WWW
- zB. XHTML --Dokumente + CSS + Java Script + eingebettete Medien, Applet

Architektur

- Client – Server – Prinzip:
  - Cient(Webbrowser) ruft Daten über Protokoll(HTTP, HTTPS, RTP …) und Uniform Ressource Identifier (URI) beim Server ab

Trends

- Integration in bestehende Anwendungssysteme (Bsp. DB ...)
- Kollaborative vernetzte Anwendungen mit Desktop/ Multimedia-UI (Bsp. Community...)
- kontextsensitive und adaptive Softwaresysteme (Bsp: Personalisierung durch Sprache …)
- breites Spektrum an Endgeräten (Übergreifende Gestaltung)
- Verlagerung komplexer Geschäftsprozesse ins WWW (Bsp: Beratungssysteme, Internet of Things)

Software-Technologie: Geschäftsanwendung

- Lösungskonzepte
  - Mehrschichtenarchitekturen (Austauschbarkeit von Schichten)
  - Funktionale Modularisierung (Wiederverwendbarkeit)
  - „Separation of Concerns“ (Trennung technischer u. Fachlicher Aspekte für Wiederverwendbarkeit)
  - Standardisierung von Prozessen und Modulen (markt für Teillösungen …)

Cloud-Computing:

Herausforderungen:

- Gigantischer Speicherplatz
- Energieverbrauch
- Datensicherheit
- Engineering der Anwendungen

Kategorien von Webandwendungen:

Kategorie 1: Statische Webanwendungen

- Statische HTML-Dateien ggf. + CSS
- Vorteil:
  - niedrige Antwortzeiten
  - Einfachheit
- Nachteil:
  - manuelle Aktualisierung
  - Inkonsistenzen von Links

Kategorie 2: Datenbankbasierte Web-Anwendung

- Nutzung von SQL-,XML-, und Mediendatenbanken zur Verwaltung von Daten
- Vorteil:
  - Trennung Präsentation und Inhalt
  - Dezentrale Administration durch mehrere Personen
  - Gut für häufig aktualisierende Webseiten und große Datenmengen
- Nachteil:
  - Zusätzliche Abbildungsvorschriften durch Templates
  - angemessene Präsentation von Daten nötig
- Technologien: JSP, PHP, ASP, Servlets

Kategorie 3: Applikationsorientierte Web-Anwendungen

- Vorteil:
  - hohe Anzahl der Benutzer
  - Skalierbarkeit, Stabilität, Performance

    - Transaktionsmanagement, Sicherheit
    - Lastausgleich und Caching
- Nachteil:
    - Komplexität der Anwendungslogik
    - Stabilitäts- und Performance-Probleme
- Lösung: verteilen der Logik auf mehreren Applikationsservern
- Einsatz:
    - komplexe Geschäftslogik
    - Transaktionen, hohe Serverlast

Kategorie 4: Dienste-Architekturen

- Nutzung standardisierter Komponenten aus denen die Anwendung aufgebaut ist
- Technologien:
    - protal-Techniken: Container, Portlets
    - Web-Services: WSDL, SOAP, UDDI
- Nutzung:
    - Intern: modulare Architektur und Strukturierung der SW
    - Extern: Nutzung von Services unabhängiger Anbieter, Entwicklung Anwendungsfelder
- Beispiel: Nutzung von Protal-orientierten Anwendungen mit Portlets

Clientseitige Web-Technologien

XHTML-Client

- klassischer Webserver
- Client kümmert sich nur um Darstellung und die Übermittlung von UI-Ereignissen
- Server bereitet HTML auf Basis der Anwendungslogik auf
- Technologien: XHTML, CSS, Javascript

Asynchronous Javascript and XML (AJAX) - Technologie

- Client kümmert sich um Präsentationslogik einschließlich Übersetzung von UI-Ereignissen auf Anwendungslogik → Webseite muss nicht immer komplett neu geladen werden
- Server liefert geänderte /gewünschte Teilbereiche der aktuellen UI

- Serveranfragen asynchron im Hintergrund, warten meist nicht notwendig
- nicht zwingend asynchron, kaum XML → viel Javascript
- Dynamische Anpassung des (X)HTMLs über DOM
- XMLHttpRequest:
  - kann HTTP Anfrage an Server schicken und auswerten
  - Zur Darstellung wird Javascript verwendet
  - Ablauf:
    - Erzeugen XMLHttpRequest
    - Verbindung zur Seite herstellen + Status abfragen
    - Ausgabe der Rückgabe: feststellen des Status + Aktualisieren der Webseite
- Bestandteile einer AJAX-Anwendung:
  - Event-Handler: Javascript bei Client wird bei Aktion aufgerufen, verarbeitet Daten, instanziiert XMLHttpRequest
  - Callback-Funktion: Javascript bei Client, bekommt AJAX-Antwort, verarbeitet Daten
  - Server-Funktion: Service auf Webserver, BSP: CGI, PHP, JSL
- Nachteile AJAX:
  - Umständliche Erstellung der Anwendung
  - mühsehlige Neuimplementierung wiederkehrender Aufgaben
  - erfordert substanzielle Programmierkenntnisse
  - Inkompatibilität von Browsern und Plattformen
  - Unzureichende Wartungs- und Testmöglichkeiten

Plug-In

- Eigenständiger Anwendungsteil
- Server stellt z.B. über RPC Anwendungslogik und Datenbank bereitet
- Fast ausschließlich https-basierte Kommunikation
- zukünftig: serviceorientierte Architektur + Web-Services

Serverseitige Technologien

Serverseitige HTML-Generierung

- Offline Vorberechnung der HTML-Seiten
- Aufruf einer HTML generierenden Anwendungen
  - CGI-Skripte: Server ruft Skript auf, Skript nutzt ggf. Datenbank, gibt HTML

zurück

- Einbettung HTML-generierender Skriptteile
  - Server-Side Scripting: Unterscheidung Compiler einmalige Generierung, Interpreter zur Laufzeit
- „XML-Weg“
  - Problem: Brüche zwischen einzelnen Schichten in Datenmodellen
  - Ziel: Harmonisierung mit XML
  - XML zur Speicherung, Anwendungslogik durch XSLT-Transformation von XML-Daten, Präsentation über XSL-Transformation (server- als auch clientseitig XSLT und XML nutzbar)

Dokumentbeschreibungssprachen

HTML – HyperText Markup Language

- früher: Anwendung von SGML(Standard Generalized markup language): mächtig, aber sehr komplex
- HTML5: ist generalisierte Sprache wie SGML, Nutzung von web-spezifischen Parsern, statt SGML-Parsern
- besteht aus: Dokumenttypdefinition, Wurzelelement <html> und <head> + <body>
- Logische Formatierung → konkrete Darstellung ist browseranhängig
- Einbindung multimedialer Inhalte über <object>

XML – Extensible Markup Language

- XML-Dateien
  - Darstellung hierarschisch strukturierter Daten in Form von Textdateien
  - dienen dem plattform- und Implementationsunabhängigen Datenaustausch
  - Parser lädt DTD und prüft XML-Dokument gegen
  - Logischer Aufbau eines Dokuments:
    - Prolog: optional, Anweisungen für Programme die das Dokument verarbeiten
    - Wurzelelement: enthält Daten des Dokuments als Elemente, Atrribute und Texte
- standardisierte Metagrammatik
  - mit der sich Regeln für die Definition von Grammatiken (Bsp: XHTML, SMIL …) definieren lassen
  - Namensräume in XML möglich: Elemente aus selben Namensraum haben selben Präfix → verweisen nicht auf reale Ressourcen, nur zur Identifizierung

- Parser:
  - Programm oder Programmteile, die XML-Dateien auslesen, auf Gültigkeit überprüfen und interpretieren

- ***Wohlgeformtheit von XML:***
  - mindestens ein Element
  - genau ein Wurzelelement
  - Elemente beachten die Groß- und Kleinschreibung
  - alle Elemente haben ein übergeordnetes Element in dem sie beginnen auch enden
  - alle geöffneten Elemente werden geschlossen
  - alle Entities sind deklariert
  - ein Element darf nicht merere Attribute mit dem selben Namen besitzen
  - Attributeigenschaften müssen in Anführungszeichen stehen
  - informell: eine wohlgeformtes XML Dokument ist syntaktisch korrekt und es lässt sich eine prinzipielle Grammatik dafür angeben

- ***Gültigkeit von XML:***
  - es gibt eine zugehörige Grammatik und es entspricht dieser
  - ist wohlgeformt

DTD – Document Type Definition:

- Schemasprache zur Einschränkung von XML-basierten Sprachen
- legt Struktur fest: Reihenfolge, Art der Attribute, Verschachtelung der Elemente
- Oft wird im XML-Dokument in Kopf auf DTD verwiesen
- Teil der XML-Spezifikation, aber selbst kein XML
- Document-Type-Declaration: (DOCTYPE)
  - Angabe der DTD am Anfang des Dokuments
  - hat die Form <!DOCTYPE name SYSTEM/PUBLIC inhalt>
- Markup-Deklarationen: in DTD
  - definiert Element-Typen, Attributlisten, Entities, Notationen, Textblöcken
  - Strukturelemente werden über Attributzuordnungen definiert:
    - Element:
      - hat die Form <!ELEMENT name inhaltsmodell>
      - definieren des elements und dessen Inhalts

        - Attribut
            - Liste der möglichen Attribute eines Elements
            - hat die Form <!ATTLIST elementname attributliste>
        - Entity
            - hat die Form <!ENTITY name „inhalt“>
            - Bei Parameter-Entities folgt ein %
            - banannte Abkürzung für Zeichenkette oder ganzes Dokument das in XML-Dokumenten mit dieser DTD genutzt werden kann
        - CDATA:
            - Textblock
        - PCDATA:
            - Textblock der weitere Anweisungen an Parser enthalten kann

- ***Nachteile von DTD:***
    - Kardinalitäten lassen sich nicht exakt definieren
    - kaum Wiederverwendbarkeit von Inhaltsmodellen
    - für Elementinhalt ist nur Datentyp #PCDATA verfügbar, auch für Attributinhalte nur sehr wenige Datentypen verfügbar
    - Namensräume können nicht in DTDs benutzt werden → Probleme Import
    - Syntax von XML und DTDs ist unterschiedlich
    - bei Definition für implizite Gültigkeitsregeln ist keine Nebenbedingung für Elemente möglich

XML Schema (XSD):

- Schemasprache zur Einschränkung von XML-basierten Sprachen
- Struktur wird in Ofrm eines XML-Dokuments beschrieben
- große Anzahl an Datentypen unterstützt
- beschreibt Datentypen, Instanzen und Gruppen solcher Instanzen
- ***einfache Datentypen:***
    - keine Attribute/Unterelemente
    - vordefiniert
    - Nutzung für Elemente und Attribute
    - eigene Typen durch Nutzung vordefinierte Typen definierbar

- ***komplexe Datentypen:***

        - können Unterelemente und Attribute besitzen
        - Sequenzen, Alternativen und Unterelemente in beliebiger Reihenfolge und Kardinalität kann definiert werden
    - Ersetzungsgruppen(Substitution groups):
        - Erweiterungsmechanismus auf Elementebene
        - ermöglicht existierende Schemas nachträglich zu erweitern
        - Definition von Ersetzungen möglich → Elemente müssen von einander Ableitung sein
    - Abstrakte Elemente und Typen:
        - ermöglicht Erzwingen von Ersetzung
        - kann direkt im Instanzdokument nicht genutzt werden
        - erfordert Ersetzungsgruppe
    - Vererbung bei komplexen Typen
        - durch Einschränkung, Erweiterung, Umdefinition

Navigation in XML Dokumenten

- ***Xlink:***
    - definiert Links über spezielle Attribute die für jedes XML-Element definiert werden können (Nutzung Namespaces)
    - definiert keine importierbare grammatik, eher generelles Konzept
    - ermöglicht:
        - Trennung zwischen Dokumentinhalt und Verknüpfung
        - Rollen für Verknüpfung
    - Unterteilung in:
        - Out-of-line-Links: zu entfernten Dokumenten
        - Inline-Links: zur Verweisquelle innerhalb eines Links
    - als auch in:
        - einfache Links:
        - erweiterte Links: 1:n- und bidirektionale-Vernüpfungen, Verweise auf Verweise
- ***Xpath:***
    - dient der Adressierung beliebiger Knoten und Knotenmengen innerhalb desselben oder eines anderen Dokuments
    - Grundlage für Xpointer und XSLT
    - operiert auf logischer Struktur
    - adressiert einzelne Knoten als auch Knotenmengen
    - XPATH- Ausdruck wird in Kontext ausgewertet

- Kontext kann vom Verarbeitungsprozessor innerhalb eines mehrstufigen Ausdrucks oder beim Auswerten verändert werden
- Xpath-Achsen: Achsen sind Richtungen in die man sich von Knoten aus begeben kann, meist: Verarbeitungsrichtung, Bsp: child, ancestor ...
- Adressierung von Achsen erfolgt immer durch aktuellen Kontextknoten

- ***Xpointer:***
  - kann Punkte und bereiche adressieren
  - Nutzung von range, Knoten oder point
  - kann mit URL verbunden werden, da Xpath ausdrücke nicht direkt in URI verwendet werden können

Dokumenttransformation: XSLT + XSL FO

- **XSL:**
  - dient zur Anpassung und Ergänzung von Präsentationseigenschaften
- ***XSL Transformations:***
  - Transformation des Quelldokuments, z.B. Extraktion, Sortierung, Nummerierung der anzuzeigenden Informationen (nicht zwingend umkehrbar)
  - XSLT-Prozessor: arbeitet mit XPath zusammen um Element zu finden für die Regel gilt (gehen von oben Baum durch und suchen Regel, rekursiv)
  - XSLT Stylesheet beschreibt wie XML in ein Anderes oder ein nicht-XML-Dokument transformiert wird
  - Steuerung der Transformation durch unabhängige regeln in Form von Templates definiert
  - Reihenfolge der Regeln ohne Bedeutung
  - Pattern: Regel nach der eine template Rule für einen Knoten ausgewählt wird → TR stimmt dann match
- ***XSL Formating Objects:***
  - dient zur Festlegung des eigentlichen Layouts des Zieldokuments (Druck, Web, ….)
  - basiert auf DSSSL und CSS
  - implementiert ein ähnliches bereichsmoddel CSS, bietet jedoch Erweiterungen wie Paginierung
  - formatiert Ergebnisbaum einer vorhergehenden XSLT-Transformation
  - XSL FO übernimmt Grundlegende CSS-Eigenschaften und fügt eigene hinzu
  - formatiert mithilfe von Bereichen:
    - Block Areas: sind gestapelt, nicht überlappend

        - Inline Areas: nebeneinander angeordnet

                → Kinder unterteilen sich wieder
    - Nutzung von spezialisierten Bereichen:
        - Glyph Areas: nur Inline Areas ohne Kinder
        - Line Areas: nur aus Inline Areas
    - <u>Verarbeitung eines XSL-FO Dokuments:</u>
        1. XSL-FO-Dokument wird eingelesen und in Knotenbaum überführt
        2. Knotenbaum wird in Formatobjektbaum überführt
        3. Verfeinerung des FO-Baums
        4. Umwandlung des verfeinerten FO-Baumes in Bereichsbaum
        5. Darstellung des Bereichbaums

XML Programmierschnittstellen XML-APIs

- ermöglicht Anwendung auf Informationen in XML-Dokumenten zuzugreifen
- übernimmt:
    - Einlesen und überprüfen der Gültigkeit des XML-Dokuments
    - Überführung in hauptspeicherrepräsentation
    - Navigation und Manipulation der Information
    - Ergebnisbehandlung
    - Zugriff auf Inhaltsmodelle und Stylesheets
    - Ausgabe der Inhalte
    - Unterstützung des Autorenprozesses
- benutzt DOM!!!

Exkurs: DOM

- standardisierte plattform und sprachunabhängige API für die Modifikation von XML/HTML-Dokumentinhalten
- definiert Standardobjekte und Standardmodell das festlegt wie Objekte kombiniert werden können
- definiert Überführung von HTML und XML in Objektstruktur
- definiert Schnittstellen und Objekte zur Repräsentation und Manipulation des Dokuments
- Core Modul für HTML oder XML zwingend erforderlich
- Defizite:
    -
- **<u>Level:</u>**

- ***DOM1:*** Core und HTML
  - baut knotenhierarchie
  - definiert Schnittstellen für HTML
  - Datentyp Node(Knoten, Document Object): definiert methoden zum Hinzufügen und Löschen von Knoten
  - DOMException: Ausgelöst wenn Operationen scheitern
- ***DOM2:*** Events, View, Style, Traversal, Range
  - Interface zur Baumstruktur, Namespaces
  - DOM Traversal bietet Schnittstelle um DOM zu durchlaufen
  - neue DOM-Module:
    - Events: generisches Ereignismodell, beschreibt Methoden eines Ereignisses mouseover die nach oben zur Wurzel propagiert werden um Target und dann Listener zu erreichen
    - Views: verscheidene Ansichten des Dokuments
    - Style: Zugriff auf Stylesheets, CSS-Schnittstelle
  - Ziele:
    - Registrierung von Ereignishandler
    - Beschreibung Ereignisfluss
    - Kontektinformationen zu Ereignis liefern
  - Knoten vom Typ:
    - DocumentFragment (leichtgewichtige Dokument-Knoten, Ausschnitt eines Dokuments repräsentiert)
    - Entityreference (reference auf Entität)
    - Element Knoten (HTML/XML-Element)
  - Event capture:
    - Ereignislistener fangen bestimmte Ereignistypen
    - von Wurzel abwärts
  - Event bubbling:
    - neben üblicher Eventweiterleitung auch Event weiterleiten an alle Listener der Elternelemente
  - Event cancelation
    - Standardevents abbrechbar je nach Ausführung des Lsiteners
- ***DOM3:*** Load and Save, Validation, XPATH

XHTML – Extensible Hypertext Markup Language:

- Nutzt Metagrammatik XML und XML
- Modularisiert

- Erweiterbarkeit: Anbieter von Web--Clients können XHTML um eigene Module erweitern
- Unterscheide zu HTML:
  - Strengere Konventionen
  - Unterscheidung zwischen Groß- und Kleinschreibung
  - Tags ohne Abschlusstag anders
  - jedes Attribut braucht Wert

CSS – Cascading Style Sheets:

- Layout-Eigenschaften werden durch separate Beschreibungssprache definiert
- ermöglicht Trennung von Struktur und Layout
- Vorrangregeln:
  - Autoren-Stylesheet
  - Benutzer-Stylesheet
  - Browser-Einstellungen
- Definition Selektor:
- Arten von Selektoren:
- Kaskaden:
-

Web-Programmiersprachen

Javascript

- Zugriff auf HTML Struktur via DOM (Domain Object Modell)

Java Applets

- GUI-Gestaltung mit UI-Bibliotheken
- Applets erlaubt Kommunikation mit Server
- werden Durch Java-Klassen implementiert die vom Server geladen werden
- sind externe Komponenten und werden in HTML-Dokumente eingebettet
- sind interaktive GUI-Elemente
- werden in sicherer Browser-Umgebung ausgeführt
- ***Einschränkungen zur Sicherheit:***
  - keine Bibliotheken benutzen
  - keine Dateien des Hosts lesen o. Schreiben

    - ausschließlich Netzwerkverbindung zu Host von dem Dateien stammen
    - keine lokale Anwendung starten
    - nur wenige Systemeigenschaften ermitteln
- ***Applet-Server-Kommunikation:***
    - über HTTP, Sockets oder RMI
    - Browser lädt applet wege applet-tag-> Request initiiert, Response von Apllet interpretiert
- ***Vorteile:***
    - Implementierung einfach
    - keine besondere Anforderungen an serverseitige Applikation
- ***Nachteile:***
    - HTTP-Kommunikation ist langsam
    - Informationen müssen HTTP-kodiert sein
    - Response muss Applet-spezifisch semantisch kodiert sein

Serverseitige Programmierung

Architektur von Webanwendungen

Definition Architektur:

- abhängig von Anforderungen, Zielen Erfahrung, Randbedingung
- Qualität durch:
    - Entwurfsprozess + Architekten + Erfahrung im Anwendungsbereich
    - Wiederverwendung

Definition Web-Architektur:

- beschreibt die Struktur (Komponenten, Schnittstellen, Beziehungen …) einer verteilten Webanwendung
- betrachtet statische als auch dynamische Aspekte, ist zugleich Ablaufplan und Bauplan
- Ziele:
    - Struktur macht Systeme verständlich und beherrschbar
    - Erleichtern der Kommunikation von Systemaspekten

Sichten auf eine Webanwendung:

- Konzeptuelle Sicht: identifiziert Entitäten der Anwendungsdomäne und deren Beziehungen

- Laufzeitsicht: beschreibt Komponenten zur Laufzeit des Systems
- Prozesssicht: Abläufe zur Laufzeit des Systems (Synchronisation und Nebenläufigkeit)
- Implementierungssicht: beschreibt Softwareartefakte (Subsytseme, Komponenten, Quellcode)

Definition Muster

- beschreibt die Lösung für wiederkehrender Entwurfsprobleme (← Entwurfskontext)
- beschreibt Komponenten und deren Verantwortlichkeiten, Beziehungen, Zusammenwirken
- ermöglicht Wiederverwendung

Definition Framework:

- ermöglicht Wiederverwendung von Mustern
- Softwaresystem mit generischer Funktionalität
- gibt Architektur + Basisfunktionalität für Anwendungsbereich vor

Schichten-Architektur:

- 1-Tier-Architektur: Präsentation Teil des Servers
- 2-Tier-Architektur:
  - Aufteilung und Funktionsblöcke: Präsentation bei Client und Anwendungs- und Ressourcenverteilung
  - dazwischen API-Kommunikation
  - auch Client/Server-Architektur genannt
- N-Tier-Architektur:
  - Nutzung von z.B. Firewall oder Proxy
  - Applikationsserver als Integrationsplattform für viele heterogene, externe Anwendungen
  - Nachteil: komplexe Middleware zur Kommunikation notwendig

Protokolle

HTTP

- verläuft in 4 Schritten:
  1. Client baut Http-Request zum Server auf
  2. Client sendet HTTP-Request zum Server
  3. Server sendet HTTP-Response zum Client
  4. Server baut Verbindung zum Client wieder ab

- zustandslos, ermöglicht bidirektionale Übertragung
- Adresse muss eindeutig identifizierbar sein
- HTTP-Adressierung:
- Nutzung von Caches und Proxies zur Entlastung und Verringerung des Datenverkehrs
- Format besteht aus:
  - Body-header: Informationen zu Body, immer dabei
  - Request-Header: vom Client, Informationen zum Request
  - Response-Header: vom Server, sendet Daten an Client
- Methoden:
  - OPTIONS: Informationen zu Kommunikationsoptionen
  - GET: fordert Ressource an, über URL
  - HEAD: nur Header von ressource wird zurückgegeben
  - POST: übertragen von Parametern
  - PUT: übertragen von Dateien
  - DELETE: löscht übergebene Ressource vom Server
  - TRACE: liefert Übertragungsweg und gesendete Nachrichten
- HTTP-Satuscode:
  - 1xx: Request angenommen und in Bearbeitung
  - 2xx: Erfolgsmeldung
  - 3xx: weitere Aktionen des Clients nötig für Bearbeitung
  - 4xx: Client-Error
  - 5xx: Server-Error
- Sessionmanagement manuell über Cookies oder Hiddenfields

RTP/RTSP: Real-Time Streaming Protokolle

- dient zum Streamen multimedialer Daten → Daten müssen Client synchronisiert erreichen
- variable Anzahl von Empfängern mit unterschiedlichen Qualitätsanforderungen unterstützt
- TCP ungeeignet, da: Zustandverwaltung nötig, Paketverlust unwichtig → Zeitbedingung
- ermöglicht Broad- und Multicast unter Verwendung eines Streamingsservers
- benutzt nur gebrauchte Bandbreite, bricht ab wenn BB zu gering
- Realtime Streaming besteht aus 4?! Streaming Protokollen:
  - ***RTP:*** eigentliche Datenübertragung Echtzeitdaten
  - ***RTP Control Protocol (RTCP):*** Teilnehmerverwaltung und QoS-

Parameter(Ermittlung Qualität, Regulierung Bandbreite)

- ***Ressoruce Reservation protocol (RSVP):*** Signalerzeugung und -verarbeitung, Kontrollprotokoll für Netzwerke
- ***Realtime Streaming Protocol (RTSP):*** Steuerung von Medienströmen, Anforderung und Übertragung Daten, Fernsteuerung Multimedia Server
- ***Session Description protocol (SDP):*** Beschreibung von Streaming Sitzungen

→ Transportprotokoll meist UDP

- ***Verlauf RTSP-Sitzung***
  1. Beschaffungsphase
  2. Verhandlungs- und Transport-Initialisierungsphase
  3. Wiedergabe- bzw. Aufnahmephase
  4. Abbruchphase
- Zustände Client: Init, Ready, Playing, Recording

Server Skript Sprachen / Technologien

- Einteilung in:
  - ***Typ 1: Selbstständige Anwendungen***
    - Laufen i-d-R- auf dem Server
    - z.B. CGI-Skripte
  - ***Typ 2: Eingebettete Skriptsprachen***
    - Logik und Gestaltung i.d.R. direkt im HTML-Code geschrieben
    - Client bekommt nur resultierenden HTML-Code
    - z.B. PHP, JSP, ASP, Perl

Common Gateway Interface(CGI):

- Schnittstelle zwischen Webserver und Serverseitiger Anwendung
- erzeugt dynamisch Content und gibt es an Webserver zum Senden, hat Zugriff auf DBMS
- CGI Interaktionsmodell:
  - werden im separaten Prozess ausgeführt
  - Dtaneübertragung Web-Server und CGI über stdin/out
  - über http://.../cgi-in/programmname ausgeführt

Java Servlets (JSL): → programmzentriert

- ausgeführt in Servlet-Containern auf Servlet-Engine (JVM)
- haben im gegensatz zu applets (clientseitig) keine graphische Benutzeroberfläche
- Trennung von Präsentation und Anwendungssteuerung

- JSL teilen den gemeinsamen Adressraum
- Programmcode der HTMl-Code erzeugt
- Schnittstelle für HTTP-Requests
- Container:
  - ermöglicht Multi-Threading → jede Clientanfrage in eigenem Thread
  - Lifecycle management der Komponenten
  - Methodenaufrufe bei Client-Anfragen
- Vorteile
  - Portabilität über Betriebsysteme etc.
  - Leistungsfähigkeit da Multithreading, Serialisierung, …
  - Effizienz: JSL bleibt geladen und damit Zustände erhalten
  - Sicherheit: Nutzung Java Security Manager
  - Einfachheit: JSL übersichtlich/einfach, Sessiontracking vereinfacht Entwicklung
- Lebenszyklus:
    1. Aufruf: JSL wird durch Container geladen
    2. bedient Requests, hält derweile State
    3. Beendigung durch Container, Entfernung aus Speicher

Java Server Pages (JSL): → dokumentzentriert

- JSP-Engine kompiliert Pages beim ersten Aufruf in JSL
- für Präsentation verantwortlich
- ist Text (HTML, SVG, XML-templates) mit eingefügten Code
- Vorteil:
  - Verwendung und Gestaltung der Anwendung einfacher als bei JSL
- Lebenszyklus:
    1. Get-anfrage
    2. JSP-Engine liest JSP-Datei ein
    3. Servlet programm erzeugt (.java(
    4. kompilieren
    5. 5. ausführen
    6. Daten senden an Server
- <u>Unterscheidung in:</u>
  - Skriptlets: eingebettete Codefragmente mit ggf. Zugriff auf Standarddokumente
  - JSP-Ausdrücke:
    - Kurzform für Stringausgabe ins Zieldokument

        - wird zur Laufzeit evaluiert
    - JSP-Deklarationsanweisung:
        - ermöglicht Definition zusätzlicher (ggf. globalen) Attribute /Variablen und Methoden
    - JSP Kommentare:
        - <%--tada –%>
    - Interpreter-Anweisungen:
        - erlaubt Import von Java-packages
        - konfiguration von servlets
        - einbinden externer Dateien
- Java Beans: auch Business Objekt
    - Komponentenmodel von Java
    - besitzt parameterlosen Konstruktor und dazu getter & setter
    - Vorteile:
        - klare Trennung der verschiedenen Aspekte
        - klare Schnittstellen
        - flexible Umstellung je nach Client
- Verknüpfung JSP und JSL:
    - Bearbeitungs-Szenario:
        - JSL nimmt HTTP-Request entgegen
        - Weiterleitung an Business-Objekte → liefert ValueObjekte
        - VOs werden durch JSL in Context abgelegt, dann vom JSP-Dokument genutzt
    - macht Wiederverwendung von Servlet-Code

PHP: Hypertext Preprocessor (PHP):

- ***no new things :(***

Semantic Web

Motivation, Grundidee, Konzepte

- Probleme bei Websuche → Webseiten für Menschen gemacht
- Ziel ist Web für Menschen ud Maschinen → Maschinenverwendbarkeit
- Vorgehensweise:

1. gemeinsame Syntax für maschinenverarbeitbare Aussagen
2. gemeinsames Vokabulare etablieren
3. Wissensverarbeitung basierend auf Vokabular

RDF:

- Beschreibungssprache für Ressourcen im WWW
- möglichst Informationsaustausch ohne Bedeutungverlust
- Grundidee:
  - Ressourcen sind durch URI eindeutig bezeichnet und mit Eigenschaften und Eigenschaftswerten beschrieben

    → Aussagen bestehen aus Subjekt, Prädikat, Objekt
- Reification: ermöglicht Aussagen über Aussagen
- Prinzip:
  - Non-Unique Name Assumption
    - sind zwei Ressourcen nicht als verschieden definiert sind, können Sie trotzt unterschiedlicher Bezeichner das gleiche repräsentieren
    - jeder kann informationen bereitstellen und sie nennen wie er will
  - Open World Assumption
    - nicht definiertes Wissen ist unbekannt, nicht zwangsläufig falsch
- Friend-of-a-friend (FOAF):
    - Modellierung von Netzwerken durch Webseite knows Webseite

SPARQL Protocol And RDF Query Language (SPARQL):

- RDF Anfragesprache an SQL angelehnt
- Triplbasierte Graphpattern für WHERE-Bedingung

Ontologien, OWL

- **Definition Ontologie:** beschreibt einen Wissenbereich mit Hilfe einer standardisierenden Teminologie sowie Beziehungen und Ableitungsregeln die Interpretationen der Klassen (→ Hierarchiebeziehungen)
- **OWL:**
  - Standard-Ontologiesprache des W3C
  - 3 Mächtigkeitsstufen: Lite, DL, Full

Konkrete Anwendung im Web:

- **RDF-in-attributes (RDFa):**
  - entwickelt um RDF in XHTML einzubinden
  - Erweiterung bestehenden Webseiten um Metadaten

- Inhalte für Mensch und Maschine in einem Dokument
- verwendete existierende XHTML-Attribute

- **Linked Open Data (LOD):**
  - öffentliches Projekt zur Verknüpfung und Verlinkung der Daten → „Wertsteigerung“

Service-orientierte Programmierung

- **Problemstellung:**
  - Support statt nur Erstellen der Webseiten → ständige Anpassung
  - notwendig ist Isolation/ Kapselung der Module
  - Vorteil: Nutzung verschiedener Module in verschiedenen Sprachen o.a.

Service-oriented architecture (SOA):

- Konzept wie Anwendungen und Software-Infrastrukturen gestaltet, bereitgestellt und verwaltet werden
- wird als Programmiermodell als auch Architektur gesehen
- **SOA-Rollen:**
  - Service Provider: implementiert einen WS und macht diesen verfügbar
  - Service Requestor: Software z.B: Applet, JSL ….
  - Service Registry: Zentrales Repository das ein Directory implementiert
- **Interaktionen:**
  1. Publizieren
  2. Auffinden
  3. Binden
- W3C-Standards:
  - SOAP für Nachrichtenformat, WSDL zur Beschreibung des WS, UDDI für das Publizieren und Auffinden

WEB Services:

Definition Webservice:

- Software als abgeschlossene Einheit mit wohl geformter Funktionalität und öffentlicher Schnittstelle(XML) im Internet bereitgestellt
- selbst beschreibende Webanwendung
- Service besteht aus:
  - eindeutige URI

        - Service Interface: Schnittstelle
        - Service Contract: Spezifikation der Verantwortlichkeit, der Funktionalität, der Bedingung und Einschränkung
        - Service Implementierung: Technische Realisierung
    - Ziel:
        - Anwendungen sol über Internet generische Komponenten nutzen können, Auswahl und Aufruf erfolgt zur Laufzeit

- Portlets: bilden auf Client-Seite einfach zu benutzende Oberfläche, auf Serverseite kapseln sie beliebige anwendungen die Darstellung an portlet weiterleitet
- Portale: Technik zur Integration verschiedener Services in eine Web-Präsentation, zusammenführen verschiedenster externer Komponenten, nutzen Protlets die für einheitlichen Zugriff sorgen

Definition Webservice-Architektur:

- Client-Server-Paradigma
- Schichtenmodell
    - Core-Layer: Kernschicht umfässt XML und SOAP
    - High-Layer: WSDL und BPEL

REST

Simple Object Access Protocol (SOAP):

- Legt Grubstruktur und Verarbeitungsvorschriften fest
- XML basiert
- unabhängig von Plattform und Programmiersprachen
- zustandsloses one-way Austauschformat (kein request-reply)
- kann auf jedes protokoll oberhalb von TCP/IP aufgesetzt werden
- Envelop-Format:
    - SOAP-Header: Indentifizierung der Inhalte (optional)
    - SOAP-Body: eigentliche mitzuteilende Information

Web Service Description Language (WSDL):

- Sprache zur Beschreibung von Geschäftprozessen

- beschreibt Webservice als Sammlung von Kommunikationspunkten, die Nachrichten austauschen können
- stellt Kontrakt(?) zwischen Service-Anbieter und Nutzer dar (Signatur, Transportprotokolle, Adresse des WS, Typ-Schemata für Datenaustausch)
- Schnittstellenbeschreibung

Universal Description, Discovery and Integration (UDDI):

- Client sucht nach geeignetem WS mit UDDi automatisch
- wichtiger Standard für Service Registry und Repositories

Business Process Execution Language (BPEL):

- XML basierte Sprache zur Beschriebung von Geschäftsprozessen über Anwendung /Unternehmensgrenzen hinweg
- Business-to-Business, nicht zur Mensch-Interaktion

Rich Internet Technologien (RIA)

- Daten und Logik Verteilung bei RIA anders als bei herkömmlichen Web-Anwendungen (N-Tier-Anwendung)
  - herkömmlich: Server handhabt Application, Database und tw. User Interface
  - RIA: Server handhabt Database u. Vlt. Teil der Application, Client hat UI und Application

Definition Web 2.0: „Wendepunkt fürs Web“

- Web als Plattform für Anwendung und Dienste
- Benutzer als Teilnehmer, nicht nur Konsument
- Inhalte als eigenständige Objekte (Seite nicht mehr grundlegend)
- Innovation im Aufbau durch die Verwendung von Komponenten
- einfache Geschäftsmodelle durch das verteilte, gemeinsame Nutzen von Inhalten und technischen Diensten
- Projekt immer im Beta-Stadium (Ende Softwarezyklus)
- Dient breitem Spektrum an Anwendung

Dimensionen des Web 2.0:

- Technische Dimensionen:
- Strukturelle Dimensionen:
  - abhängig von Technik

    - bei Ajax Seite nur Container für Objekte
- Soziale Dimensionen:
    - Nutzer als virtuelle Identität repräsentiert
    - Interaktion

Definition Rich-Client-Anwendung:

- Computer in einer verteilten Umgebung der verteilte und lokale Ressourcen nutzt
- liegt zwischen:
    - Thin Client: nutzt ausschließlich verteilte Ressourcen
    - Thick Client: die meisten Ressourcen liegen lokal vor
    - BSP: richer von links nach rechts: AJAX, XUL, Flash, Java Applets, Java Web Start
- Ziel: ähnliche Interaktionsmöglichkeiten auch bei Web-Anwendungen

Web Application Frameworks:

- Prinzipien:
    - Wiederverwendbarkeit von Code
    - Schnelle Entwicklung lauffähiger Anwendungen
    - Reduktion von Komplexität
- Eigenschaften:
    - Einsatz Model View Controller (MVC)
    - Nutzung Templates
    - Scraffolding: Erstellung von CRUD-Seiten (Create-Read-Update-Delete)
- Framework: bietet Unterstützung innerhalb einer bestimmten Sprache an
- Plattform: kann sprachübergreifend Deployment- und Laufzeitaufgaben übernehmen
- Toolkit: Sammlung von Werkzeugen und Komponenten
- Auswahl des Web-Application Frameworks:
    - Abhängig von: Anforderungen /Szenarien, Entwicklungszeit/ -aufwand, Client-Server-verteilung, Aufwand, vereitung und Community, Reife, Standardkonformität ...

Java Server Faces (JSF):

- Serverseitige Technologie zur entwicklung Java-basierter Web-Anwendungen
- besteht aus: API für UI + jSP Tag Bibliothek

- zur Verhinderung unerlaubter Zugriffe werden alle Aktionen mit dem benutzer von Einem Front-End-Faces-Servlet durchgeführt

JbossRichFaces (JRF):

- Baut auf JSF und Ajax4jsf
- dient der AJAX-Kopatibilität in Web-Anwendungen sowie verbesserter UI

Rich Ajax Platform (RAP):

- Einheitliches Entwicklungsmodell für Desktop- und Webandwendungen durch Renderingkits
- Verwendung von SWT(Widgets) und JFace(UI) und RAP Widget Toolkist(RWT)
- RWT reimplementierung von SWT-Komponenten
- HTML, Jacarscipt etc in RWT gebündelt
- Lifecycle:
  - readData: Einlesen der Daten vom Client
  - ProcessAction: Weiterleitung Event an Widget
  - Render: Änderung des Widgets an Client übertragen

Google Web Toolkit (GWT):

- OpenSource Java-Framework zur Entwicklung von AJAX-Anwendungen
- GWT-Compiler übersetzt Anwendung zu XHTML & Javascript → Trennung von javascript und dessen Einschränkungen
- Einbindung von Widgets
- wie bei gewohnter Programmioerung von OO-Mustern
- Unterstützung bei test(jUnit) und Debugging durch Entwicklungs und Web-Modus
- GWT garantiert gleiches verhalten von Java und generieter JS- Anwendung
- besteht aus: JRE Emulations Bibliothek + GWT Java-Bibliothek + Java-to-Javarscipt-Compiler

RAP vs GWT

- Gemeinsamkeiten:
  - Java-basiert → Debugging, testen, IDEs, Tools …
- Unterschiede:
  - RAP:
    - Java-Code komplett auf Server ausgeführt

        - ständige vollautomatische Synchronisation Client-Server
        - starke Netzwerklast Client-Server
    - GWT:
        - Java-code in Javascript umgewandelt
        - Clientseitige-Ausführung von Logik
        - geringere Kommunikation und Caching

The future of Web

HTML5

- Status ist „Candidate Recommendation“ (fast abgeschlossen)
- besser in: Tags, Barrierefreiheit, SEO, Kompatibilität (?), Offlinespeicher, Gerätezugang, Konnektivität
- Spezifikationen:
    - Vokabular aus alten und neuen Elementen
    - bringt Strukturmodell zur Schachtelung von Tags mit
    - XHTML und HTML haben gemeinsames DOM5
    - generalisierte Sprache wie SGML
    - bringt Standarddarstellung für jedes Elements mit
    - Microdata: maschinenlesbare Informationen in HTML-Dokumenten
    - Browserkontext:
        - erlaubt Nachrichtenübermittlung zwischen 2 unabh. BK
        - Vereinheitlichung des verhalten der Browser
    - Canvas2D: Schnittstelle zu 2D-Formen
    - HTML+RDFa: Einbettung von RDF
    - Einbinden von Audio und Video durch entsprechende Tags
- ***Ziele:***
    - Kompatibilität
    - Verwendbarkeit
        - neue Funktionen sollen bestehende Probleme lösen, aber bestehende Funktionen nicht neu erfunden werden
    - Sicherheit
    - Konsistenz
    - XML-teile aus XHTML Anwendungen sind erlaubt. HTML und XHTML besitzen eine gemeinsame DOM-Abbildung

    - Vereinfachung
        - geringe Komplexität + definiertes verhalten → interoperable Implementation
    - Universalität
    - Zugänglichkeit

CSS3

- Ziel: Modularisierung
- noch nicht standardisiert, aber gut unterstützt
- Neues:
    - 19 neue Selektionsmuster: Pesudoklassen, negation, Gecshwisterselektoren, Pseudoelemente, Attributselektoren
    -

-

URI / URL – Unterschiede:

Definition Ressource im WWW:

Arten von Caching:

Session-Implementierungstechniken:

- mit Hilfe von Cookies die per JS geschrieben und gelesen werden können
- mit Hilfe Erweiterungen der URL, nur bei Nutzung der Formulare
- PHP $_Session-Objekt

Unterschied CSS2 und CSS3:

Ziele von CSS3:

Unterschiede JSP und PHP:

XML-basierte Sprachen:

- XHTML, XAML, SVG, RSS, XML-Schema

Warum „cascading“ stylesheets?

Nennen der verschiedenen Selektorarten:

- einfach Selektoren
- Kombinatoren
- Pseudoklassen
- strukturelle Pseudoklassen
- Pseudoelemente

Klausur

Vorlesung 5:

Was versteht man inhaltlich unter SOA, was ist das Neue? Welche Konsequenz hat es für die Programmierung von Web-Anwendungen?

Was versteht man konkret unter einem Service bei SOA?

Erläutern Sie die Standards WSDL, SOAP und UDDI. Erläutern Sie diese an Hand eines Architekturbildes einer typischen dynamischen Web-Anwendung.

Was sind die Elemente von SOAP? Warum wird es benötigt?

Was sind die Elemente von WSDL, was beschreibt es wofür?

Wie sind die Daten / Informationen auf einem UDDI-Server klassifiziert bzw. strukturiert?

Wie kommen diese Informationen dahin, welche Schnittstellen gibt es?

Wie kann eine Web-Anwendung den UDDI-Server nutzen?

Was versteht man unter BPEL? Wofür ist es notwendig? Was bedeutet BPEL für den Programmierer eine Geschäftsanwendung?

Vorlesung 6:

Was ist eine typische Architektur für eine RIA?

– Was versteht man aus der Sicht eines Programmierers einer dynamischen Webanwendung unter Ajax?

- Wo kommt diese technologie bei der Programmierung in betracht und wie wird sie eingesetzt?
- Welche Methoden gehören zu einem XMLHttpRequest Objekt?
- Unterstellt sie haben eine dynamische Web-Anwendung vor 10 jahren mit XHTML, CSS, Javascript und JavaServlets programmiert und Sie wollen diese Anwendung nun mit Web 2.0 Techniken also mit Ajax überarbeiten. An welchen Stellen der Anwendung müssen Sie wie eingreifen?
    - Welche modernen Frameworks kennen Sie mit denen „Web 2.0“-/Ajax-Anwendungen entwickelt werden können? Was sind deren wichtigsten Merkmale?
    - Worin unterscheiden sie sich von der klassischen Ajax-Programmierung?
- Durch welche Techniken verwinfachen Web-Frameworks den Entwicklungprozess

von Webanwendungen?

- Welche Apsekte der Client-Server-Aufteilung eines programms müssen bei der Verwendung von RAP und GWT berücksichtigt werden?
- Worin unterscheiden sich Eclipse RAP und GWT in der Entwicklung und bei der Ausführung der Web-Anwendung?
- Nennen Sie die wichtigsten Merkmale von HTML5!