Web-basierte Anwendungen

Studiengänge AI (4140) & WI (4120)

HTML5

Neue Möglichkeiten – eine kleine Übersicht

HTML5: Übersicht

- **Neue Elemente**
  - Strukturelemente
    `article, aside, bdi, details, figure, figcaption, footer, header, hgroup, mark, nav, rp, rt, ruby, section, summary, time, wbr`
  - GUI-artige Elemente
    `command, datalist, keygen, meter, output, progress`
  - Native Multimedia-Unterstützung
    `audio, source, track, video`

Demo

  - Externe interaktive Inhalte, Plugins
    `embed`
  - 2D- und 3D-Grafik
    `canvas`

Demo

- **Neue Attribute**
  - Neue Typen für „input“
    `color, dates, email, number, range, search, tel, times, url`
  - Neue globale Attribute
    `data-*`
    `contenteditable, contextmenu, draggable, dropzone, hidden, spellcheck`
  - Event-Attribute: `on*`
    - Window events (3 + 16)
      `onload`
    - Form events (6 -1 + 5)
      `onsubmit`
    - Keyboard events (3 + 0)
      `onkeydown`
    - Mouse events (7 + 9)
      `onclick`
    - Media events (1 + 22)
      `onabort`

Demo-Quellen: z.B. https://www.w3schools.com/html5/tag_video.asp

HTML5: Übersicht

- **Neue Methodik**
  - Abkehr von SGML
    - **`<!DOCTYPE html>`** ohne FPI, DTD, Version
    - Regeln für die Behandlung von „tag soups“ (nur HTML-Darstellung)
    - SGML-Syntaxregeln werden nicht mehr streng verfolgt, sondern ersetzt
  - Zusammenführung der Spezifikationen von HTML und DOM
  - Großes Gewicht auf JavaScript (und CSS3)
  - Direkte Einbettung von SVG und MathML in HTML ohne NS-Konzept
    - XHTML: Wie gewohnt mit Namensräumen (NS)
  - WHATWG: „*Living Standard*“
    - keine Versionen, keine Termine, ständige Weiterentwicklung
  - Entwicklung z.Z. nur in einer Hand:
    - Ian Hickson, Google Inc. - ein „wohlwollender Diktator“?
  - W3C
    - Am 29.10.2014 erreicht HTML5 REC-Status
    - HTML 5.1 REC ed.2: 3.11.2017, HTML 5.2: CR vom 2.11.17

HTML5: Übersicht

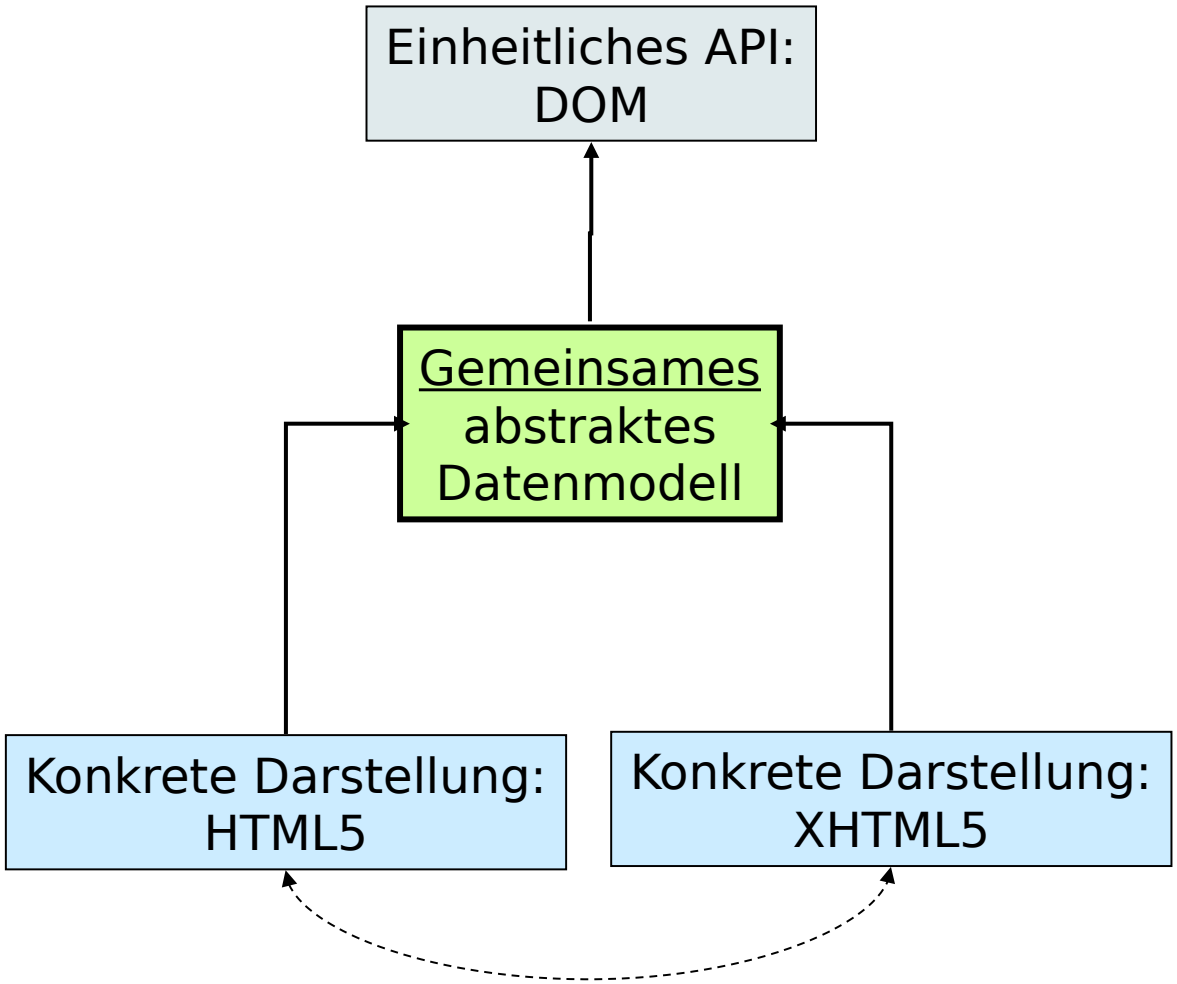
- **Anmerkungen zu den neuen Strukturelementen**
  - Die von HTML5 vorgesehene Strukturierung von **„article“** wird kontrovers diskutiert
    - Sehr spezieller Fall („Blog“)
    - Nicht gerade intuitiver Aufbau, vgl. iX-Artikel
  - **„nav“, „aside“ etc. bieten wenig grundsätzlich Neues**
    - Bisherige Layouts erreichen mittels „div“ und CSS ähnliche Wirkung!
    - Möglich: Neue CSS-Voreinstellungen für „nav“ etc. in heutigen Browsern.
    - Gut: Klare Benennung auf abstrakter Ebene, Vereinheitlichung
  - „rp“, „rt“, „ruby“: Für Randnotizen, insb. in Fernost genutzt
    - Entspricht dem Ruby-Modul von XHTML 2.0
  - **„wbr“: Zur „Empfehlung“ eines Zeilenumbruchs**
    - Kann z.B. Silben eines langen Worts umschließen
  - **„figure“, „figcaption“: Endlich richtige Umgebungen für Abbildungen!**
  - **„time**“: Zur **strukturierten** Erfassung von **Datums- und Zeitangaben**

HTML5: Übersicht

- Nicht mehr unterstützte Elemente
  - **`acronym, applet, basefont, big, center, dir, font, frame, frameset, noframes, strike, tt, u, xmp`**

- Weitere Eigenschaften/Ziele – keine Themen für dieses Grundlagen-Modul
  - Unterstützung von Mobile Web APIs
    - z.B. *„Geolocation“*
  - *(Offline) Web Storage*
    - *Application cache, local storage*
  - *Microdata*
  - *Communication:*
    - *Web sockets, Server-sent events, cross-document messaging, channel messaging*
  - *Web workers*
    - Hintergrund-Skripte, „Langläufer“

HTML5



Weitgehend, aber nicht völlig äquivalent!

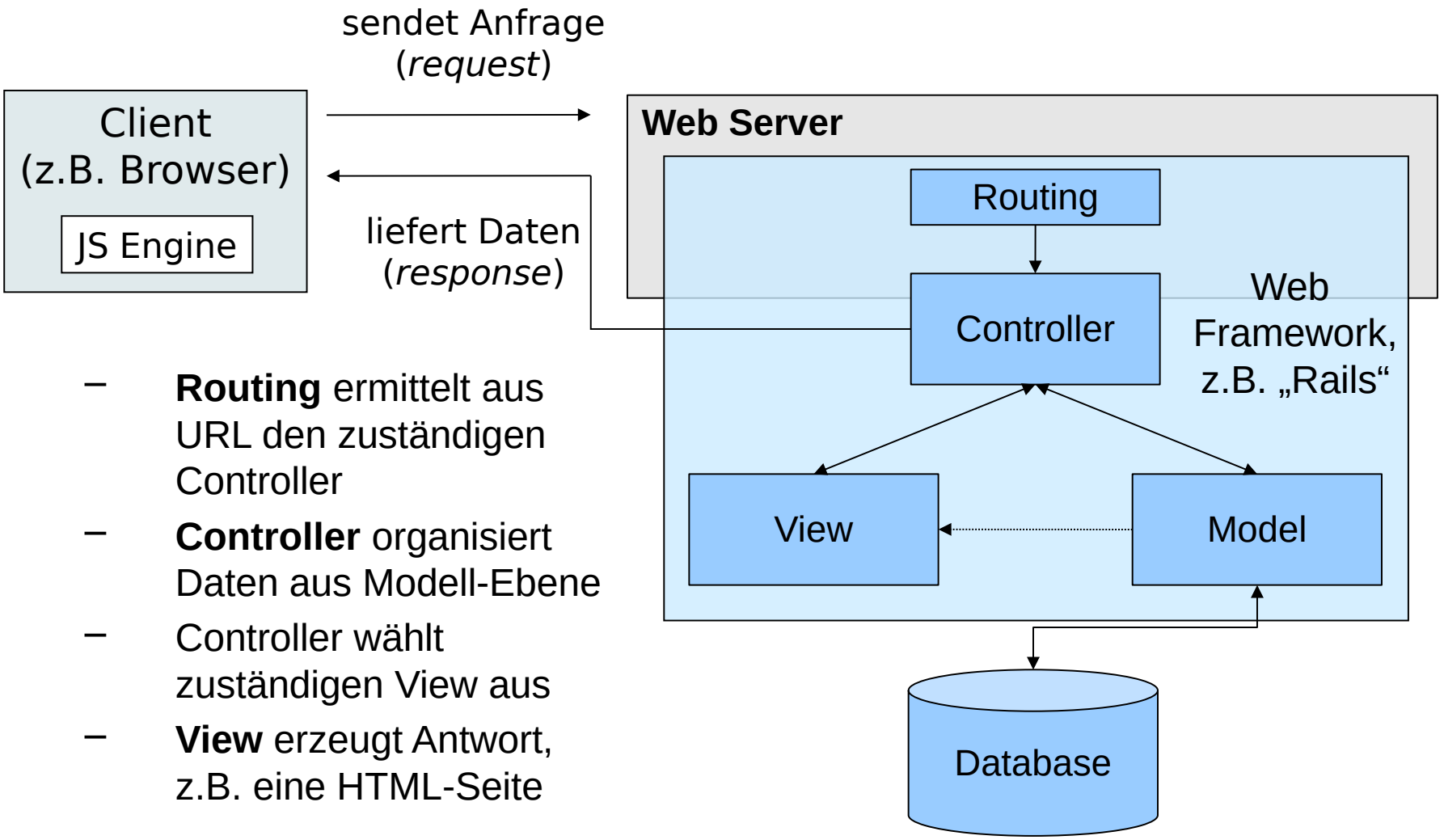
HTML-Formulare

(X)HTML-Formulare

- Warum hier besonders behandeln bzw. erarbeiten?
  - Formulare spielen eine entscheidende Rolle bei der Interaktion mit den Anwendern:
    - Erst die Interaktion unterscheidet Web-basierte Anwendungen von reinen Hypertext-Netzen
    - Der Informationsfluss zwischen Server und Client sollte detailliert verstanden werden. Formulare bilden auf Clientseite eine entscheidende Komponente.
    - *User input* bietet großes **Potenzial für Sicherheitslücken** – erst detailliertes Verständnis ermöglicht systematische Abhilfe
  - Frameworks wie Rails kapseln Formulare zwar, setzen für fortgeschrittene Nutzung aber Kenntnis der HTML-Hintergründe voraus.
    - Spezielle HTML-Optionen lassen sich etwa direkt „durchreichen“
    - Rails-Namenskonventionen wirken sich bis auf Formularinhalte aus und lassen sich erst voll verstehen, wenn man die Möglichkeiten und Grenzen von F. kennt
    - Noch nicht von Rails unterstützte HTML5-Eigenschaften erfordern „Eigenbau“
  - Formulare bilden daher auch eine Grundlage der nächsten Praktikumseinheit…

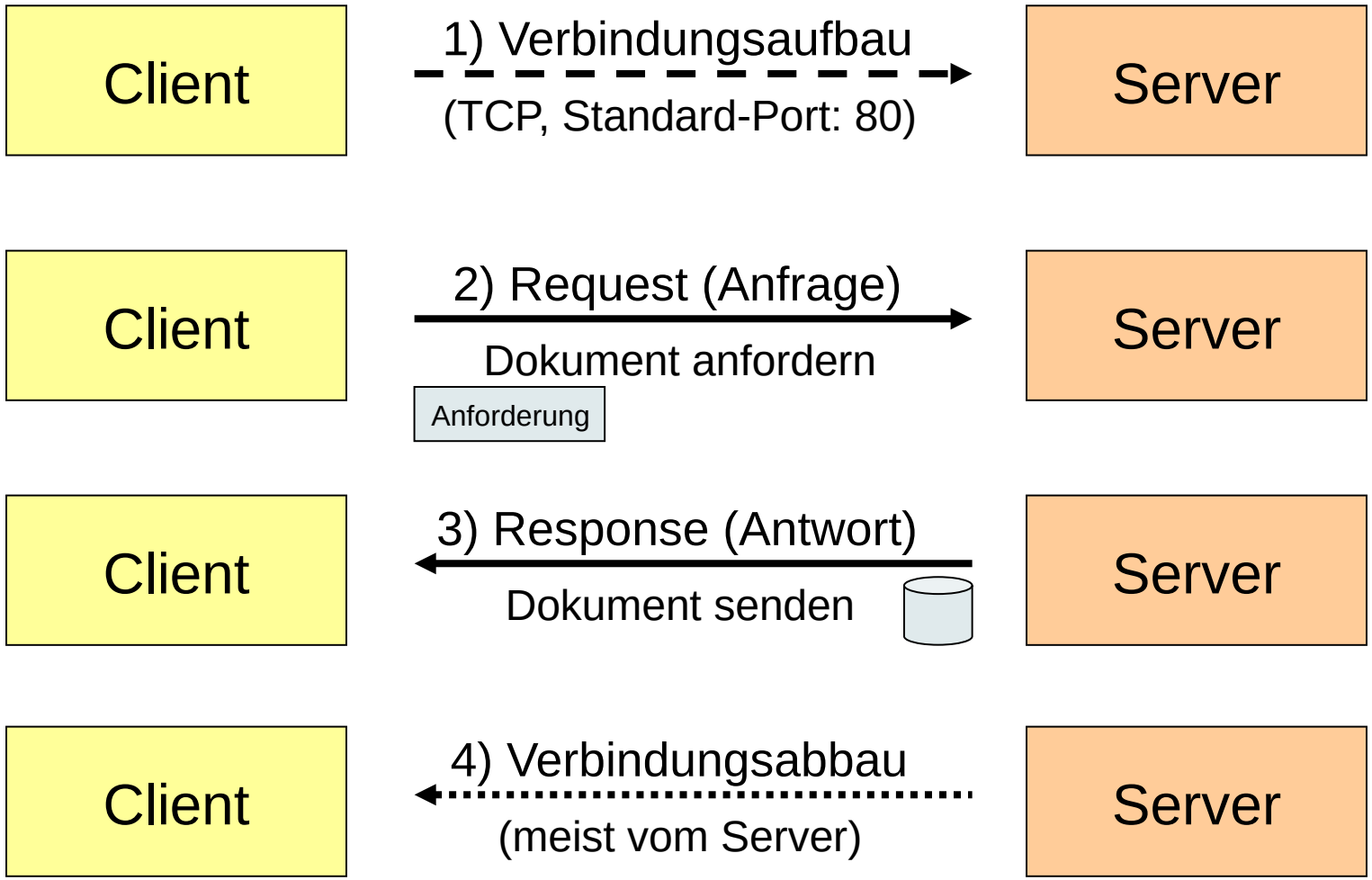
Erinnerung

- Serverseitige Web-Anwendung, hier: MVC-Entwurfsmuster



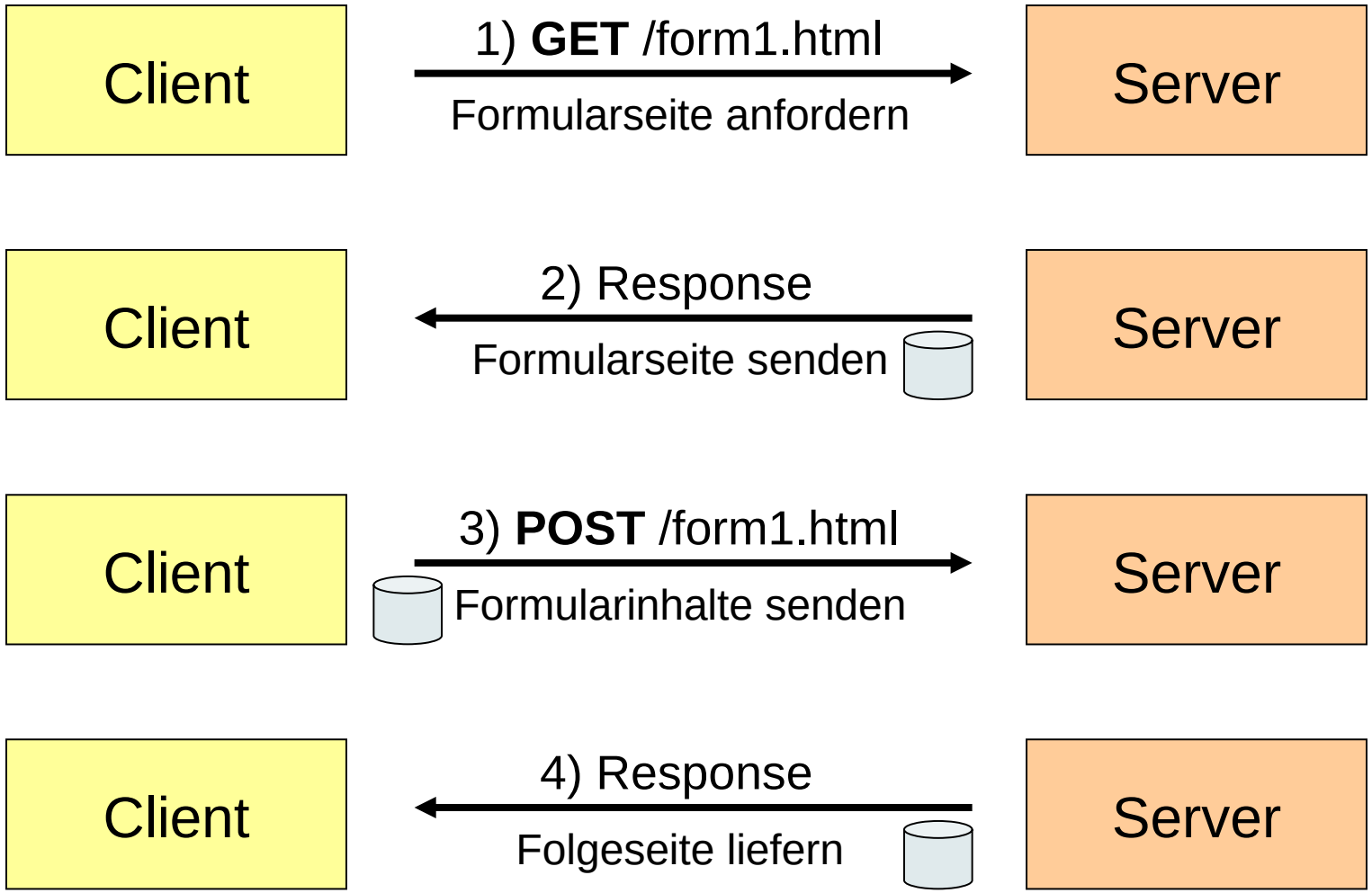
- **Routing** ermittelt aus URL den zuständigen Controller
- **Controller** organisiert Daten aus Modell-Ebene
- Controller wählt zuständigen View aus
- **View** erzeugt Antwort, z.B. eine HTML-Seite

Vorgriff: HTTP: Ein reines C/S-Protokoll



Request/Response: Der Server wird nur nach Client-Requests aktiv!

Formulare: 2 Schritte



HTML-Formulare

http-Methoden im Formular-Kontext

- GET
  - Grundbedeutung: Client fordert Dokument vom Server an.
  - Lesezugriff – für das Senden von Formulardaten schlecht geeignet!
  - Leider für Schreibzwecke weit verbreitet, gar Voreinstellung bei Formularen
    - Daten in URL-Erweiterungen codiert („application/x-www-form-url-encoded“)
  - **Bei Formularen sinnvoll, die nur Daten für eine Auswahl der nächsten auszuliefernden HTML-Seite enthalten (W3C-Empfehlung)**
  - **Nicht bei Formularen sinnvoll, die serverseitig zu speichernde Daten enthalten!**

- POST
  - Übermitteln („Veröffentlichen“) eines neuen Dokuments
  - Schreibzugriff – gut für Formularinhalte (insb. komplexe)
  - Daten oft in HTTP „body“ enthalten („text/plain“, „text/xml“, etc.)
  - Perfekt angemessen für Formulare mit neuen Inhalten, die der Server speichern soll

HTML-Formulare

http-Methoden im Formular-Kontext

- PUT
  - Aktualisieren eines bereits vorhandenen Dokuments
  - Schreibzugriff – dennoch nur sinnvoll bei Überarbeitung von Inhalten
  - Im Prinzip geeignet zum Senden von Daten, die per Formular aktualisiert wurden
  - **Kommt in der Praxis nicht vor**

- DELETE
  - Löschen eines vorhandenen Dokuments
  - **Für Formulare nicht sinnvoll**
  - Von Browsern i.d.R. nicht verwendet

- Hinweis
  - Die REST-Architektur (→ Kapitel zu http) räumt mit derlei Unsitten auf, dazu später mehr

HTML-Formulare

- **Validierung von Formulardaten: Warum?**
  - Schutz der Anwendung vor unsinnigen Eingaben
  - Schutz der Anwendung von „*code injection*“-Angriffen
  - Schutz vor irrtümlichen Angaben
  - Hilfestellung & Komfort für den Anwender

- **Validierung von Formulardaten: Wege**
  - **Serverseitig**: Unverzichtbar (warum?), aber ein späteres Thema
  - **Clientseitig, mit JavaScript**
    + Sehr flexibel, jahrelange Erfahrung
    - Programmieraufwand; JS ist abschaltbar
  - **Clientseitig, HTML und Browser**
    ++ Sehr einfach (per Deklaration)
    - Erst mit HTML5 möglich – ~~noch zu geringe Browserunterstützung?~~
    - Individuell gestaltbar?
  - Beispiele: Pflichtfeld; E-Mail-, URL-, Tel.-, Zahlen-Feld, erlaubte Wertemengen- und Bereiche (min, max, „Regulärer Ausdruck“)

HTML-Formulare

- Dokumentation
  - Alles Erforderliche für gewöhnliche Formulare bis XHTML 1.0, incl. zahlreicher Beispiele, finden Sie im Formularkapitel von SelfHTML: https://wiki.selfhtml.org/wiki/HTML/Tutorials/Formulare
  - HTML5-Aspekte sind zum großen Teil bei W3Schools zu finden: https://www.w3schools.com/html/html_forms.asp
    - Alle Unterpunkte zu „HTML Forms“
  - Für den jeweils aktuellsten Stand zu HTML5 ist ein Blick in den Entwurf der HTML5-Spezifikationen erforderlich. Auch W3Schools ist hier nicht vollständig! Details in Kapitel 4.10 von:
    - https://www.w3.org/TR/html5/ (*Recommendation, 28.10.2014*),
    - https://www.w3.org/TR/html51/ (*Recommendation, 3.11.2017 (2. ed.)*),
    - https://www.w3.org/TR/html52/ (*Recommendation,* 28.01.2021)
    - https://html.spec.whatwg.org/ („Living standard”)

HTML-Formulare

- Einige zusammenfassende Beispiele und Ergänzungen:

```
<form action="http://www.example.org/cgi-bin/feedback.pl" method="get">
    <!-- hier folgen die Formularelemente -->
</form>
```

  - Attribut „**action**“ (einziges Pflichtattribut):
    - URI wird beim Absenden des Formulars aufgerufen (→ submit)
  - Attribut „**method**“:
    - Die HTTP-Methode zum Senden des Formularinhalts. Default ist „**get**“, alternativ: „**post**“.
    - Verwenden Sie „get“, wenn der Formularinhalt i.w. entscheidet, welchen Inhalt Ihre nächste Seite haben soll.
    - Bei komplizierten Formularinhalten wie Bestellungen, die Dokument-Charakter annehmen, sensiblen Daten wie Passwörtern, oder allg. bei umfangreichen Formularen, sowie für Datei-Uploads wählt man „post“.
    - Bem: Dies entspricht auch der ursprünglichen HTTP-Logik, s.o.
  - Weitere Attribute: „target“, „accept-charset“, „enctype“, „name“, …

HTML-Formulare

- Unterelement **input**

```
<form action="...">
    <p>Vorname:<br/>
      <input name="vorname" type="text" size="30" maxlength="30">
    </p>
    <p>Nachname:<br/>
      <input name="nachname" type="text" size="30" maxlength="40">
    </p>
</form>
```

  - Attribut „type“: Legt den Eingabetyp fest. Sehr vielgestaltig. Beispiele:
    - „**text**“, „**password**“: Einzeilige Text-Eingabe, ggf. verdeckt
    - „**checkbox**“, „**radio**“: Auswahl-Element
    - „**submit**“, „**reset**“, „**button**“: Button-artig, mit assoziierten Aktionen
    - „**image**“: Zeigt Bitmap-Grafik, liefert angeklickte x/y-Koordinaten
    - „**file**“: Für Datei-Auswahl auf Clientseite und Upload des Inhalts
    - „**hidden**“: Für Anwender unsichtbares Feld, z.B. für Session-Verwaltung
  - Attribut „name“:
    - Identifiziert dieses Formularfeld. Wichtig für die Anbindung an Rails!
  - Weitere Attribute: Viele, meist abhängig von „type“.
    - Viel Neues hierzu durch HTML5!

HTML-Formulare

- Unterelement **`input`**
  - Attribut „type“: Neue HTML5-Typen:
    - „**color**“: Komfortabler Color-Picker
    - „**date**“, „**datetime-local**“: Datums/Zeit-Picker
    - „**email**“, „**tel**“, „**url**“: Einzeilige Eingabe, mit Aufbau-Prüfung
    - „**number, range**“: Bequeme Zahleneingabe, Schieberegler
    - „**month, week, time**“: Weitere Picker für zeitliche Angaben
  - Browser-Abhängigkeit
    - Unterstützung der Browser variiert noch stark
    - Ohne Unterstützung: Reine einzeilige Texteingabe
      - Teils mit Format-Validierung
    - Mit Unterstützung: Von einfachen Schiebern bis aufwändigen ‚Pickern‘

HTML-Formulare

- Unterelement **`textarea`**

```
<form action="...">
    <p>Hier der Anfang der Geschichte:<br/>
      <textarea name="user_eingabe" cols="50" rows="10">
        Es war dunkel, feucht und neblig ...
      </textarea>
    </p>
</form>
```

  - Aufgabe: Mehrzeilige Texteingabe-Box
  - Attribute „cols“, „rows“:
    - Größe des Anzeigefeldes, in Anzahl Zeichen bzw. Zeilen
  - „Attribut „name“:
    - Identifiziert dieses Formularfeld. Wichtig für die Anbindung an Rails!
  - Weitere Attribute: „readonly“, „disabled“, …

HTML-Formulare

- Beispiele aus der Praktikumsaufgabe

Anmeldung

Nachname	
Vorname	
Geschlecht	Männlich ○ Weiblich ○ Divers ○
Geburtsdatum	05 . 04 . 1987
Familienstand	Ledig ○ Verheiratet ○ Geschieden ○ Verwitwet ○
verheiratet seit	- -
E-Mail-Adresse	
Homepage	
Telefonnummer	
Passwort	
Passwort-Wh.	

Absenden

Fragebogen

Schulabschluss	Keiner
Abgeschlossene Berufsausbildung	Nein ○ Ja ○ Ja, Meisterbrief ○
Beworben bei	Frankfurt UAS, Hochschule Darmstadt, Hochschule RheinMain, Uni Frankfurt
Entfernung zur Hochschule	6
Verkehrsmittel zur Hochschule	☑ Zu Fuß ☐ Per Fahrrad ☑ Per ÖPNV ☐ Mit PKW ☐ anders
Farbe des HSRM-Logos	
Zufriedenheit mit dem Studium (0-100%)	

Absenden

HTML-Formulare

- Unterelement **`select`**

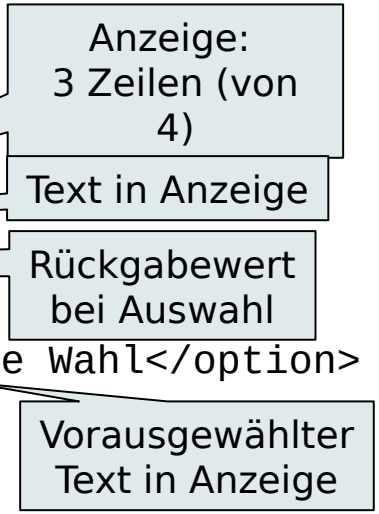
```
<form action="...">
    <p>Wählen Sie einen Eintrag aus:<br/>
      <select name="user_eingabe" size="3">
        <option>Erste Wahl</option>
        <option>Zweite Wahl</option>
        <option>Dritte Wahl</option>
        <option>Vierte Wahl</option>
      </select>
    </p>
</form>
```

  - Aufgabe: Auswahl-Liste
  - Attribute „size“:
    - Anzahl der angezeigten Optionen
  - „Attribut „name“:
    - Identifiziert dieses Formularfeld. Wichtig für die Anbindung an Rails!
  - Weitere Attribute: „readonly“, „disabled“, „multiple“ …

HTML-Formulare

- Unterelement **`select`**
  - **Trennung von Anzeige und Rückgabewert**
  - **Vorauswahl**

```
<form action="...">
    <p>Wählen Sie einen Eintrag aus:<br/>
      <select name="user_eingabe" size="3">
        <option value="1">Erste Wahl</option>
        <option value="2">Zweite Wahl</option>
        <option value="3" selected="selected">Dritte Wahl</option>
        <option value="4">Vierte Wahl</option>
      </select>
    </p>
</form>
```



HTML-Formulare

- Neue HTML5-Unterelemente
  - **datalist**
    - Eine Liste vordefinierter Optionen für ein input-Element
  - **keygen**
    - Generierung eines Schlüssel-Paars (public/private key)
  - **output**
    - Anzeige berechneter Werte aus Eingabefeldern (mit JavaScript-Hilfe)

(Hier nicht näher betrachtet)

HTML-Formulare

- **Rails-Helper**
  - Rails bietet zahlreiche Hilfsmethoden an, um komplizierte HTML-Konstrukte mitsamt der von Rails benötigten Voreinstellungen / Attribute erzeugen zu lassen
    - Das betrifft i.w. die Attribute **`id`** und **`name`**
  - Hilfsmethoden gibt es insbesondere für die verschiedenen Typen der Input-Felder eines Formulars
  - Inzwischen werden auch alle neuen HTML5-Formularfeldtypen unterstützt
    - Ausnahmen: type ∈ {reset, button, image}
  - **Nutzen Sie zur Bildung von Formularen möglichst Rails-Helper!**
    - Sie ersparen Ihnen viel Detail-Arbeit
    - Sie sorgen später für ein reibungsloses Funktionieren im Zusammenspiel mit Controllern und Models

HTML-Formulare

- Rails-Helper, Beispiel:

```
= form_for 'weather_data' do |f|
  = f.text_field :sky    # Beschreibung der Bewölkung
  = f.number_field :temp # Temperatur in °C
  = f.submit "Abschicken"
```

ergibt folgenden HTML-Code:

```
<form action="/weather_data" method="post">
  <!-- einige Extras hier nicht gezeigt… -->
<input type="text" id="weather_data_sky" name="weather_data[sky]"/>
<input type="number" id="weather_data_temp"
  name="weather_data[temp]" />
<input type="submit" id="weather_data_submit" name="commit"
  value="Abschicken"/>
</form>
```

HTML-Formulare

- Weiterer Stoff zum Nachlesen:
  - Unterelement „fieldset“
  - Unterelement „button“
  - CSS für Formulare
- Demos (sofern Zeit)
  - Bei wiki.selfhtml.org
  - Bei w3schools.com