1 Einleitung

Im Fach *Web-Engineering* werden vor allem Webapplikationen behandelt, die entweder im (mehr oder minder) gesicherten Intranet einer Firma / Organisation oder im Internet eingesetzt und angewendet werden können. Gegenüber der bloßen Präsentation von Informationen ermöglichen Webanwendungen umfangreiche Interaktionen der Benutzer.

Die Aufgaben im Praktikum *Web-Engineering* sollen Ihnen einen ersten Eindruck verschaffen, wie Webanwendungen grundsätzlich aufgebaut sein können, welche softwaretechnischen Aspekte wichtig und welche Hilfsmittel erforderlich sind.

Nachfolgend wird ein kurzer Überblick präsentiert. Im Rahmen der weiteren Veranstaltungen zum Fach werden diese wesentlich vertieft und weitere Aspekte, Konzepte und Implementierungsvarianten behandelt.

2 Webapplikationen

2.1 Konstruktionsprinzip

Wie alle Applikationen werden auch Webapplikationen nach Prinzipien und Methoden des *Software Engineering* entworfen und implementiert. Ein wichtiges Prinzip dabei ist die **Trennung von Struktur, Verhalten und Präsentation**:

- **Struktur** umfasst alle Aspekte, die den statischen Aufbau von Bestandteilen einer Applikation beeinflussen, z.B.
  - Datenstrukturen allgemein
  - Struktur der Datenbasis
  - Aufteilung in Komponenten (als Bestandteil der Architektur)
  - Struktur der Benutzungsschnittstelle
- zum **Verhalten** zählen alle dynamische Aspekte, also Änderungen (und die daraus resultierenden Aktionen) z.B.
  - durch Handlungen des Benutzers
  - durch die Auswertung von Daten
  - durch die Kommunikation mit anderen Anwendungen
- **Präsentation** meint nicht nur eine visuelle Darstellung, sondern auch unterschiedliche Erscheinungsformen von Daten, z.B.:
  - als relationale Datenbasis
  - als XML-Datei
  - als JSON-Datei.

2.2 Architektur

Webapplikationen sind i.d.R. sog. *Client-Server*-Anwendungen. Es handelt sich somit um verteilte Anwendungen, die aus 1 bis N Clients und einem Server bestehen. Diese tauschen nach einem festgelegtem Verfahren (Protokoll) Leistungen aus:

- Client:
  - fordert Leistungen an
  - erhält als Ergebnis dieser Leistungsanforderung Daten
  - wertet die gelieferten Daten aus.
- Server:
  - bietet Leistungen an
  - stellt Leistungen nur auf Anforderung zur Verfügung
  - liefert als Ergebnis der Leistungen Daten.

In einer Client-Server-Architektur ist immer der Client der aktive Teil! D.h., alle Aktivitäten werden durch den Client veranlasst, der Server wartet auf Leistungsanforderungen durch den Client, um diese dann auszuführen. Der Server ist inaktiv, wenn es keine Leistungsanforderungen gibt. Der Server kann auch nicht ohne Leistungsanforderung Daten an den Client ausliefern.

(Hinweis: es gibt verschiedene Verfahren, um diese Einschränkung aufzuheben. Diese werden in einem späteren Teil der Vorlesung behandelt.)

2.3 Webclient

Bei Webanwendungen dient der Client normalerweise zur Visualisierung der gelieferten Daten. Technische Basis dieser Clients sind Webbrowser wie FireFox oder Chrome. Im weiteren Verlauf werden diese Clients als *Webclients* bezeichnet.

Im Rahmen des Praktikums *Web-Engineering* werden bei der Implementierung der Webclients genutzt:

- HTML5
  - Auszeichnungssprache vor allem zur Beschreibung der Struktur von Inhalten (siehe weiter unten)
  - intern wird damit eine baumartige Datenstruktur aufgebaut, die als DOM (Document Object Model) bezeichnet wird
- CSS (1,2,3) (siehe weiter unten)
  - Sprache zur Beschreibung der Präsentation von Inhalten
- BOM (Browser Object Model)
- JavaScript (siehe weiter unten).

Es gibt auch Clients, die bei Webanwendungen genutzt werden, die über keine interaktive Benutzungsschnittstelle verfügen. Darauf wird in weiteren Vorlesungen noch eingegangen.

2.4 Webserver / Applikationsserver

Die Server in Webanwendungen nehmen die Leistungsanforderungen der Clients entgegen (*Requests*), verarbeiten diese und übermitteln die Ergebnisse zurück an die anfragenden Clients (*Responses*).

Webserver sind i.d.R. “Standard”-Server im Webumfeld, die eine Reihe von Aufgaben erledigen:

- Auslieferung statischer Ressourcen, z.B.
  - Textdateien, die in HTML5 notierte Inhalte aufweisen (“Webseiten”)
  - Textdateien, die in CSS notierte Regeln enthalten
  - Textdateien, die JavaScript-Code enthalten
  - Bilder
- Ausführen externer Programme (d.h. Programme, die nicht direkt im Webserver selber implementiert sind):
  - über die sog. CGI-Schnittstelle
  - über Erweiterungen / zuladbare Module wie z.B. Interpreter für Progammiersprachen.

Eine weitverbreitete Variante dieser Art von Webservern ist der Apache-Server.

Bei einem (Web-)Applikationsserver liegt der Focus auf den fachlichen Aspekten der Applikation und der Implementierung der daraus folgenden Verfahren und Abläufe. Die speziell programmierten Teile des (Web-)Applikationsserver werden direkt mit den Komponenten verbunden, die den Kern des Webservers im Applikationsserver bilden.

Im Rahmen des Praktikums *Web-Engineering* werden bei der Implementierung der Applikationsserver genutzt:

- Python als Programmiersprache (siehe weiter unten)
- das Python-Framework cherrypy
  - implementiert einen Webserver
  - bietet Schnittstellen und Methoden zur Implementierung der Anwendungslogik
- JSON (JavaScript Object Notation) zur einfachen Implementierung der Speicherung von Daten.

3 HTTP

HTTP (*Hypertext Transfer Protocol*) ist das Verfahren, nach dem bei einer Webapplikation Webclient und Webserver Leistungen austauschen. Basiseigenschaften der derzeit noch hautpsächlich eingesetzten Version 1 (1.1, 1.0) sind z.B.:

- die Inhalte werden als Text ausgetauscht, ggf. werden Ersatzdarstellungen (wie z.B. Base64-Codierungen) verwendet
- Request wie Response sind in einen Header-Bereich und einen Body-Bereich unterteilt
  - im Header-Bereich werden steuernde und beschreibende Daten eingetragen
  - im Body-Bereich werden die eigentlichen Inhalte eingetragen
- die Kommunikation erfolgt zustandlos, d.h. Request - Response bilden einen eigenständigen Ablauf, der unabhängig von vorangangenen oder nachfolgende Requests - Responses ist.

Sie können den Datenverkehr mit den Entwicklungswerkzeugen der Webbrowser beobachten.

4 HTML

HTML ist eine sog. Auszeichnungssprache: mit speziellen Hinweisen werden in einem Text die Bedeutungen der einzelnen Textbestandteile gekennzeichnet.

Im Fach *Web-Engineering* wird die Variante 5 (HTML5) verwendet. Es wird hier **immer** die strengere syntaktische Form benutzt, bei der die einzelnen Auszeichnungen (*Tags*) immer geschlossen werden:

- entweder durch ein korrespondierendes End-Tag, falls es zum Tag untergeordnete Elemente geben darf, z.B.

```
<p>
   ein Text
</p>
```

- oder im Element selber, z.B.

```
<input type="text" />
```

Die Verwendung einer anderen Syntax ist im Praktikum nicht zulässig!

Mit HTML5 definiert man eine Datenstruktur: eine Baumstruktur, die im Webbrowser in Form des DOM (Document Object Model) nach der Interpretation der HTML5-Quelltexte vorliegt. Eine vollständige Beschreibung, die hier *HTML-Dokument* genannt wird, enthält folgende Elemente:

- Kennzeichnung, dass es sich um HTML5 handelt (`DOCTYPE`-Angabe)
- das Wurzelelement `html`, das alle anderen Elemente umschließt
- das Element `head`, das beschreibende Daten enthält
- das Element `body`, das den eigentlichen Inhalt umschließt.

Es ist sinnvoll, stets einen Titel anzugeben (Element `title`) und die Zeichenkodierung (spezielle Form des Element `meta`) anzugeben.

```
<!DOCTYPE html>
<html>
   <head>
      <title>Hier steht der Titel</title>
      <meta charset="UTF-8" />
   </head>
   <body>
      <!-- so schreibt man einen
      Kommentar -->
   </body>
</html>
```

(Hinweis: es sind heutzutage verkürzte Schreibweisen zulässig. Die Verwendung wird kontrovers diskutiert. Ich empfehle sie nicht. Für den Anfänger ist schwerer nachvollziehbar, was eigentlich beschrieben wird.)

Keine Angst vor deutschen Umlauten!

Mitunter scheint die Darstellung deutscher Umlaute oder anderer Sonderzeichen nicht richtig zu erfolgen. Das passiert immer dann, wenn man zwei wichtige Regeln / Einstellungen nicht beachtet:

- geben Sie im `head` des HTML-Dokuments die Zeichenkodierung "UTF-8" an (siehe Beispiel oben)
- speichern Sie den Quelltext des HTML-Dokuments stets mit der Zeichenkodierung "UTF-8" ab
  - bei vielen Quelltexteditoren gibt es die Möglichkeit, ggf. die Zeichenkodierung des Quelltextes zu ändern, wenn man nicht mit der gewünschten Kodierung begonnen hat.

Verwenden Sie bei der Erstellung von Quelltexten für HTML-Dokumente stets Einrückung, um die Baumstruktur zu verdeutlichen. Die Einrückung ist syntaktisch nicht erforderlich, aber hilfreich beim Lesen!

Fehlertoleranz oder: wie merke ich, ob ich etwas falsch gemacht habe

Schlichte kurze Antwort: entweder gar nicht oder dadurch, dass die Elemente nicht so dargestellt werden, wie Sie es erwartet haben.

Lange Antwort: die Webbrowser ignorieren bei der Interpretation von HTML5 fehlerhafte Konstrukte und Schreibweisen und ersetzen sie teilweise sogar durch mehr oder weniger sinnvolle Annahmen. Fehlermeldungen erfolgen nicht. Dadurch sind Fehler oft schwer nachvollziehbar. Mit Hilfe der Entwicklerwerkzeuge können Sie feststellen, ob der notierte HTML5-Quelltext tatsächlich so interpretiert wurde, wie Sie sich das vorgestellt haben.

5 CSS

Mit CSS wird beschreiben, welche Elemente in der HTML5-Baumstruktur wie dargestellt werden sollen. Dazu werden in einer CSS-Datei Regeln notiert, die so aufgebaut sind:

- zunächst wird ein CSS-Selektor angegeben, der der Elementauswahl dient
- dann werden die CSS-Darstellungseigenschaften aufgeführt, die berücksichtigt werden sollen; die Angaben werden durch geschweifte Klammern umschlossen.

Um CSS-Regeln in einem HTML-Dokument zu verwenden, notiert man sie entweder direkt in einem `style`-Abschnitt oder gibt dort eine `@import`-Anweisung an. Dann wird die angegebene CSS-Datei verwendet.

```
<style>
   @import "/webteams.css";
</style>
```

Einfache Beispiele für CSS-Regel sind:

```
p {            /* alle p-Elemente, d.h. alle Absätze */
   color: red; /* Textfarbe */
}

div {          /* alle div-Elemente, d.h. alle Abschnitte */
   margin-left: 200px; /* linker Rand */
}

table {        /* alle table-Elemente, d.h. alle Tabellen */
   border: 1px solid; /* Rahmen um die Tabelle, nicht die Zellen! */
   border-collapse: collapse; /* kein Leerraum zwischen einzelnen Zellen */
}

th, td {       /* alle th- und td-Element, d.h. alle Header - und alle einfachen Zellen */
border: 1px solid; /* Rahmen um die Zellen! */

}
```

Fehlertoleranz oder: wie merke ich, ob ich etwas falsch gemacht habe

Probleme können entweder mit den CSS-Selektoren oder den CSS-Darstellungseigenschaften auftreten:

- zu einem CSS-Selektor werden keine passenden Elemente gefunden
  - das kann schlicht daran liegen, dass es keine entsprechende Struktur im HTML-Dokument gibt (kein Fehler!)
  - oder der CSS-Selektor ist falsch notiert (das wäre dann ein Fehler)
- es werden passende Elemente gefunden, aber die CSS-Darstellungseigenschaften zeigen keine Wirkung; dass kann daran liegen
  - dass die CSS-Darstellungseigenschaften nicht auf das Element angewendet werden können (sozusagen: sie passen nicht) oder
  - dass die CSS-Darstellungseigenschaften falsch notiert wurden (das wäre dann ein Fehler).

Wie bei HTML5 erfolgen keine direkten Fehlermeldungen. Nicht erkannte, falsch geschriebene CSS-Selektoren oder CSS-Darstellungeneigenschaften können jedoch mit den Hilfsmitteln der Entwicklungswerkzeuge der Webbrowser erkannt werden.

6 Python

Python ist eine allgemein einsetzbare Programmiersprache. Python-Programme werden i.d.R. interpretiert. Das bedeutet, dass eine Python-Quelltextdatei linear abgearbeitet wird, Anweisung für Anweisung. Einen Startpunkt wie die `main`-Routine bei C oder C++ gibt es nicht. Streng genommen gibt es auch keine Deklarationen, z.B. von Datenstrukturen / Klassen oder Prozeduren/Funktionen/Methoden. Es handelt sich bei Python in diesem Fällen um spezielle Anweisungen, die zur Speicherung / Bereitstellung der entsprechenden Elemente führen. Auf diese kann dann später zugegriffen werden:

```
def MeineFunktion():      # die def-Anweisung dient zur Speicherung der Definition einer Funktion
   print("MeineFunktion!")

MeineFunktion()           # damit wird die zuvor gespeicherte Funktion aufgerufen und ausgeführt
```

Im Fach *Web-Engineering* wird Python 3 verwendet. Alle Python-Quelltexte werden UTF-8-kodiert gespeichert.

Prozeduren/Funktionen und Klassen

Es können sowohl einfache Prozeduren/Funktionen als auch Klassen mit Methoden definiert und benutzt werden.

Modul-/Paket-Konzept

Jede Python-Quelltextdatei stellt ein Modul dar. Die Python-Quelltextdatei, die dem Python-Interpreter als Argument zur Ausführung angegeben wird, ist sozusagen das 'Main'-Modul (offiziell gibt es diese Bezeichnung nicht!).

Weitere Module werden bei der Ausführung eines Programms durch das Modul-/Paket-Konzept von Python berücksichtigt:

- mit der Anweisung `import` wird auf anderen Module zugegriffen (*nicht* vergleichbar mit `include` in C oder C++!)
- damit werden auch Module der Standardbibliothek angesprochen
- die `import`-Anweisung wird oft an den Anfang eines Python-Quelltextes gesetzt; das ist aber nicht zwingend erforderlich, `import`-Anweisungen können auch an anderen Stellen auftreten; das kann aber auch unübersichtlich sein
- auf Bezeichner, die in einem anderen, importierten Modul definiert werden, wird mit Hilfe des `.`-Operators zugegriffen (Element-Selektion wie bei C-/C++-Datenstrukturen)
- ein Verzeichnis wird als ein Paket mit einzelnen Modulen interpretiert, wenn sich im Verzeichnis die Datei `__init__.py` befindet (man beachte: 2 Unterstriche vor und nach init !); diese Datei darf leer sein, sie muss nur existieren.

Mit dem Modul-/Paket-Konzept können sehr komplex strukturierte Python-Anwendungen erstellt werden, die trotzdem gut wartbar bleiben!

Kontrollfluss: Einrückung als syntaktisches Element

Python-Quelltexte sind zeilenorientiert aufgebaut. Je Zeile wird eine Anweisung notiert, ein Abschlusszeichen wird *nicht* verwendet (also: im Gegensatz zu C/C++/Java etc. schließen die Anweisungen/Zeilen nicht mit einem ; ab!).

Zur Steuerung des Kontrollflusses stehen u.a. zur Verfügung:

- Fallunterscheidungen (`if/elif/else`)
- Schleifen (`while`, `for`).

Blöcke werden in Python durch *einheitliche Einrückung* gekennzeichnet im Gegensatz zu C/C++/Java etc., wo Blöcke in geschweifte Klammern gesetzt werden. Die Einrückung ist ein syntaktisches Element. Verletzungen der Regeln zur Einrückung führen zu Fehlermeldungen des Python-Interpreters. Damit wird eine übersichtliche Schreibweise erzielt.

In der Praxis treten dabei gerade bei Anfängern **zwei Probleme** auf:

1. die Einrückung wird nicht einheitlich durchgeführt, d.h. es wird unterschiedlich tief eingerückt oder es werden sowohl Tabulatorzeichen als auch Leerzeichen zur Einrückung verwendet
   - **dringende Empfehlung:** verwenden Sie **nur Leerzeichen** zur Einrückung, rücken Sie einheitlich ein (z.B. 2, 3 oder 4 Leerzeichen)
2. es wird durch die Einrückung ein falscher Bezug hergestellt.

Das Problem 2 kann man bei folgendem Quellcode diskutieren:

Beispiel 1

```
1    # x werde irgendwie zur Verfügung gestellt
2    if x > 10:
3       x += 1
4    y = 20 * x
5    print(y)
```

Beispiel 2

```
1    # x werde irgendwie zur Verfügung gestellt
2    if x > 10:
3       x += 1
4       y = 20 * x
5    print(y)
```

Im 2. Beispiel befindet sich die Zuweisung an die Variable y im Block der Fallunterscheidung. Wenn die Bedingung nicht zutrifft, wird dieser Block nicht ausgeführt, an y erfolgt keine Zuweisung. Eventuell ist y sogar undefiniert. Vielleicht war Beispiel 1 gemeint, aber es wurde falsch editiert. Solche Fehler lassen sich nur schwer zur Laufzeit entdecken.

Dynamische Bindung

Die Bindung eines Bezeichners an ein Element (Daten, Datenstrukturen oder Programmeinheiten wie Funktionen) erfolgt stets erst zur Laufzeit. Einem Bezeichner kann man daher nicht von vorneherein ansehen, welcher (Daten-)Typ eingesetzt wird. Die Vor- und Nachteile dieses Konzepts werden in den Veranstaltungen weiter behandelt.

Datenstrukturen

Wichtige Datenstrukturen in Python sind z.B.:

- Zeichenketten (*Strings*), die mit vielen Methoden einfach bearbeitet werden können
- Listen, die dynamisch aufgebauten Arrays in C/C++ entsprechen; sie können beliebig viele Elemente unterschiedlichen Typs enthalten
- Dictionaries, Key-Value-Paare, die den Maps der C++-Standardbibliothek entsprechen.

7 JavaScript

JavaScript hat nichts mit Java zu tun! Verwechseln Sie diese beiden grundverschiedenen Programmiersprachen nicht.

Lediglich ein Teil der Syntax, insbesondere der einfachen Kontrollflüsse, ist den Schreibweisen in C/C++/Java nachempfunden.

JavaScript-Progamme werden (wie Python-Programme) durch einen Interpreter schrittweise ausgeführt. Bei JavaScript ist es zusätzlich erforderlich, eine Ausführungsumgebung zur Verfügung zu stellen. Diese muss/sollte z.B. Ein-/Ausgabeoperationen zur Verfügung stellen. In JavaScript selber gibt es keine entsprechenden Konzepte.

JavaScript wird in erster Linie zur Programmierung der Webclients eingesetzt. Dort ist es defacto derzeit die einzige einsetzbare Programmiersprache (es gibt weitere clientseitige Programmiersprachen, die aber alle in JavaScript compiliert werden; früher verwendete Microsoft clientseitig VB-Script, das ist aber nur noch eine historische Randnotiz).

Die Ausführungsumgebung für JavaScript in Webbrowsern ist einerseits das DOM (Document Object Model) und andererseits das BOM (Browser Object Model), mit dem der Zugriff z.B. auf Bereiche oder ganze Webbrowser-Fenster möglich ist.

Bei den heute verfügbaren Webbrowsern können Sie bereits eine Vielzahl der Erweiterungen von JavaScript nutzen, die mit dem Sprachstandard “ES6” eingeführt wurden. Dazu zählt insbesondere eine an den Klassenkonzepten der Sprachen C++/Java orientierten Schreibweise für das Prototype-Konzept.

Prozeduren/Funktionen und Klassen

Es können sowohl einfache Prozeduren/Funktionen als auch Klassen mit Methoden (ES6) definiert und benutzt werden.

Prozeduren/Funktionen werden mit dem Schlüsselwort `function` eingeleitet. Sie sind selber ein Object und können daher in Variablen gespeichert werden.

```
window.onload = function () {
   // hier steht weiterer Quelltext
}
```

Im Beispiel wird dem Datenelement `onload`, das Bestandteil der globalen Variable `window` ist, der Wert des Funktionsausdrucks zugewiesen. Der Funktionsausdruck hat hier selber keinen Namen (anonymer Funktionsausdruck). Angesprochen wird die Funktion über den Bezeichner `window.onload`.

Kontrollfluss

Der Kontrollfluss wird ähnlich C/C++/Java notiert. Es stehen Fallunterscheidungen (`if/else` und `switch`) zur Verfügung sowie verschiedene Schleifenformen.

Dynamische Bindung

Wie bei Python erfolgt die Bindung eines Bezeichners an ein Element (Daten, Datenstrukturen oder Programmeinheiten wie Funktionen) erst zur Laufzeit. Einem Bezeichner kann man daher nicht von vorneherein ansehen, welcher (Daten-)Typ eingesetzt wird. Die Vor- und Nachteile dieses Konzepts werden in den Veranstaltungen weiter behandelt.

Datenstrukturen

Wichtige Datenstrukturen in JavaScript sind:

- Zeichenketten (*Strings*), die mit vielen Methoden einfach bearbeitet werden können
- Arrays, die wie die Listen in Python dynamisch aufgebaut sind; sie können beliebig viele Elemente unterschiedlichen Typs enthalten
- Dictionaries, Key-Value-Paare, die den Maps der C++-Standardbibliothek entsprechen; sie werden in JavaScript aber als ‘Objekte’ bezeichnet, da als Values auch Funktionen auftreten können (siehe oben); Einzelheiten werden im weiteren Verlauf der Vorlesung behandelt.