Datenbanken in Web-Applikation einbinden

M151

Autor:
Peter Wlodarczak

Datum:
März, 2023

Inhaltsverzeichnis

1 Einführung in die Web Programmierung

1.1 Das World Wide Web (WWW)

Das World Wide Web (WWW), allgemein als Internet oder Web bekannt, ist ein Informationssystem, welches Dokumente und andere Web Ressourcen wie Bilder oder physische Geräte über sogenannte URLs (Uniform Resource Locator) identifiziert. Ein URL hat z. B. folgendes Format:

```
https://www.example.com
```

Die Ressourcen werden über das Hypertext Transfer Protocol (HTTP) übertragen, und z. B. in einem Browser als Web Seite angezeigt. Die Ressourcen werden über einen Web Server zur Verfügung gestellt wie in Abbildung 1 dargestellt.

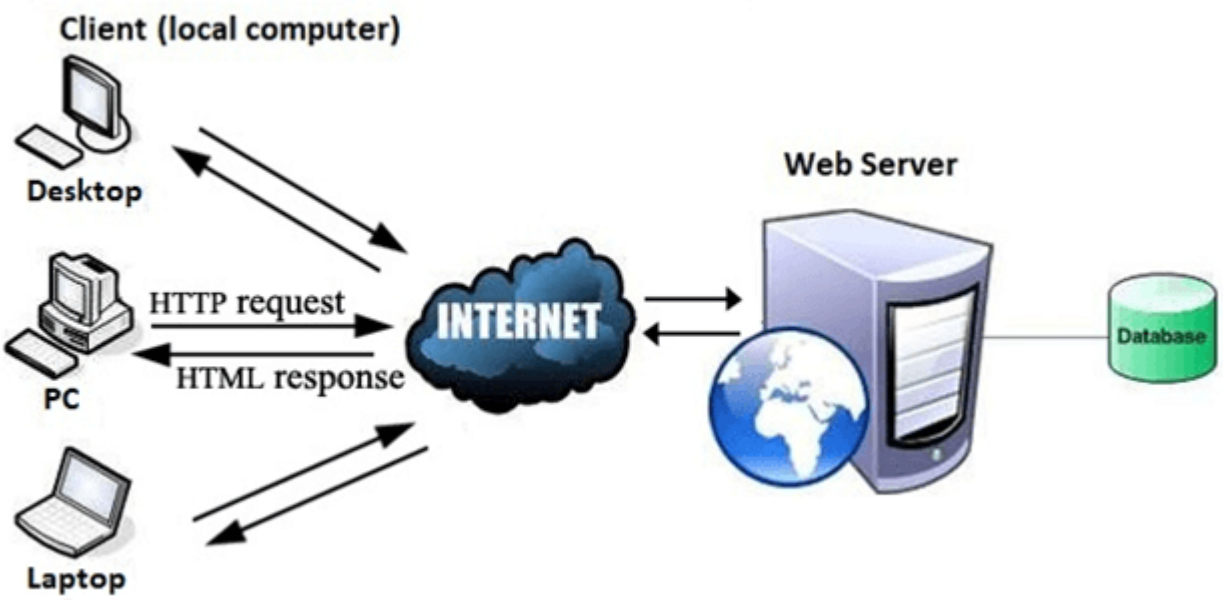

Abbildung 1: Browser - Server Kommunikation

Das Internet und WWW sind nicht Synonyme. Das WWW benutzt das Internet für die Kommunikation. Z. B. Voice over IP (VoIP) benutzt auch das Internet für die Kommunikation, hat aber mit dem WWW nichts zu tun.

1.1.1 Web 2.0

Web 2.0 ist ein Begriff, der die Entwicklung des Internets von einem statischen, inhaltsorientierten Medium zu einem interaktiven, User-zentrierten Medium beschreibt. Es

bezieht sich auf eine Reihe von Technologien und Services, die es den Usern ermöglichen, online zu interagieren, zu teilen und zu erstellen.
Im Gegensatz zum Web 1.0, das hauptsächlich aus statischen Webseiten bestand, die von wenigen ausgewählten Personen erstellt und aktualisiert wurden, ist das Web 2.0 von Benutzern erstellt und gepflegt. Es ermöglicht den Benutzern, Inhalte in Form von Text, Bildern und Videos zu teilen, Blogs und Foren zu erstellen, soziale Netzwerke aufzubauen und sich an Online-Kollaborationen zu beteiligen.
Einige der bekanntesten Beispiele für Web-2.0-Technologien und Dienste sind Wikipedia, Facebook, Twitter, YouTube und Instagram. Diese Plattformen sind alle darauf ausgerichtet, es Benutzern zu ermöglichen, Inhalte zu erstellen, zu teilen und zu interagieren.
Zusammenfassend lässt sich sagen, dass das Web 2.0 eine Änderung der Art und Weise darstellt, wie Menschen das Internet nutzen und darauf reagieren, und dass es den Schwerpunkt auf die Zusammenarbeit, Interaktivität und den Austausch von Informationen und Ideen legt.

1.1.2 Web 3.0

Das Web 3.0, auch als "Semantic Web" oder "Web of Data" bezeichnet, bezieht sich auf eine hypothetische Weiterentwicklung des Internets, die auf der Idee basiert, dass Maschinen die Bedeutung von Informationen auf der Web-Ebene verstehen und miteinander interagieren können.
Im Web 3.0 werden Daten nicht nur als Text auf einer Website oder als Datei betrachtet, sondern als "semantische" Informationen, die von Maschinen gelesen und verarbeitet werden können. Es würde eine Umgebung schaffen, in der Maschinen und Systeme besser und intelligenter miteinander kommunizieren und zusammenarbeiten können.
Einige der Technologien, die für die Realisierung des Web 3.0 erforderlich sind, umfassen künstliche Intelligenz (KI), maschinelles Lernen, Natural Language Processing (NLP), Blockchain-Technologie und das Internet der Dinge (IoT). Ein Beispiel für eine Web 3.0-Anwendung könnte eine intelligente virtuelle Assistentin sein, die in der Lage ist, natürliche Sprache zu verstehen und menschliche Anforderungen zu erfüllen, indem sie auf Informationen zugreift und mit verschiedenen Systemen interagiert.
Obwohl das Konzept des Web 3.0 noch in der Entwicklung ist und es noch keine einheitliche Definition gibt, sind viele Experten der Meinung, dass es das Potenzial hat, das Internet zu einem noch leistungsfähigeren und nützlicheren Werkzeug für die Menschheit zu machen.

1.2 HTTP

HTTP (Hypertext Transfer Protocol) bildet die Basis der Kommunikation des WWW. HTTP ist ein Client-Server Kommunikationsprotokoll. HTTP wurde von der Internet Engineering Task Force (IETF) und dem World Wide Web Consortium (W3C) standardisiert. Es funktioniert nach dem Request-Response Prinzip. Ein Client, z. B. ein Browser, schickt einen HTTP Request in Form einer Message an den Server. Der Web Server antwortet mit einer Response Message, z. B. eine HTML Seite mit zusätzlichen Inhalten (Content)

wie Bilder oder Videos im Message Body. Heute wird meistens die verschlüsselte Variante, HTTPS (Hypertext Transfer Protocol Secure), verwendet.
Neben Browsern können auch andere Applikationen wie mobile Apps HTTP Requests senden und Web Content anzeigen.
Es gibt folgende HTTP Methoden:

- GET
- POST
- PUT
- HEAD
- DELETE
- CONNECT
- OPTIONS
- TRACE
- PATCH

Die am häufigsten benutzten Methoden sind GET und POST.
Die **HTTP GET** Methode wird verwendet, um Daten von einer Ressource anzufordern. Bei einem HTTP Get Request wird ein Query String im URL als key/value Paar mitgeliefert. Z. B. ein Request zu einer Confluence Seite kann so aussehen:

```
https://confluence.domain.com/pages/viewpage.action?pageId=132521448
```

Ein paar Anmerkungen zu HTTP Get Requests:

- GET Requests können gecached werden
- GET Requests werden in der Browser History gelogged
- GET Requests können als Browser Bookmark verwendet werden
- GET Requests sollten nie verwendet werden mit sensiblen Daten
- GET Requests können nur verwendet werden, um Daten anzufordern, nicht zum verändern

HTTP Post Request werden verwendet um Daten zum Server zu schicken um eine Resource zu kreieren/verändern.
Die Daten, welche zum Server geschickt werden, werden im Body des Post Requests gespeichert. Ein Beispiel für einen Post Request kann wie folgt aussehen:

```
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

licenseID=string&content=string&/paramsXML=string
```

Post ist eine der am häufigsten benutzten HTTP Methoden.
Ein paar Anmerkungen zu HTTP Post Requests:

- Post Requests werden nie gecached
- Post Requests werden nicht in der Browser History gelogged
- Post Requests können nicht als Browser Bookmark verwendet werden
- Post Requests haben keine Restriktion bezüglich der Datenlänge

Wird nach einem Post Request der Back Button im Browser geklickt, werden die Daten nochmals zum Server geschickt.
Eile Liste von allen HTTP Methoden ist in Tabelle 1: HTTP MethodenTabelle 1 gegeben.

S.N.	Methode	Beschreibung
1	**GET**	GET Requests fragen Daten von einem Server über einen URL an. GET Requests sollten nur Daten anfragen, nicht verändern.
2	**HEAD**	Gleich wie GET, aber es wird nur die Statuszeile und der header übertragen.
3	**POST**	POST Request senden Daten im Body des Posts zum Server.
4	**PUT**	Ersetzt den gesamten Content in der Target Resource mit dem Content, der zum Server gesendet wird.
5	**DELETE**	Löscht die Ziel Resource über einem URL.
6	**CONNECT**	Stellt einen Tunnel zum Server her über einen URL.
7	**OPTIONS**	Beschreibt die Optionen für die Kommunikation mit der Ziel Resource.
8	**TRACE**	Führt einen Message Loop Back Test aus mit dem Pfad zu der Target Resource
9	**PATCH**	Ersetzt einen Teil einer Resource

Tabelle 1: HTTP Methoden

Nachdem der Server einen HTTP Request erhalten und interpretiert hat, antwortet er mit einer HTTP Response. Eine HTTP Response kann so aussehen:

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC 11 - //W3C // DTD XHTML 1.0 S t r i c t / /
E N"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de"
lang="de">
<head>
<title>Title</title>
</head>
<body>
<h1>Heading 1</h1>
</body>
</html>
```

Hier konnte der Request erfolgreich bearbeitet werden, was mit dem Status Code `HTTP/1.1 200 OK` in der ersten Zeile bestätigt wird. Kann eine Seite nicht gefunden werden oder wurde der Zugriff verweigert, antwortet der Server mit einer Fehlermeldung. Ein Status Code besteht aus drei Stellen, wobei die erste Stelle den Typ des Codes definiert. Tabelle 2 listet die 5 Typen von Codes auf.

S.N.	Code	Beschreibung
1	**1xx: Informational**	Der Request wurde erhalten und wird bearbeitet
2	**2xx: Success**	Der Request wurde erhalten und konnte erfolgreich bearbeitet werden
3	**3xx: Redirection**	Der Request wird weitergeleitet
4	**4xx: Client Error**	Der Request enthält ungültigen Syntax oder kann nicht bearbeitet werden
5	**5xx: Server Error**	Der Server konnte den Request nicht bearbeiten

Tabelle 2: HTTP Status Codes

Die HTTP Status Codes können erweitert werden und eine HTTP Applikation muss nicht alle Status Codes verstehen.

1.2.1.1 Session Management

HTTP ist **stateless**, d. h. jeder Request/Response Aufruf wird separat und ohne Kontext behandelt. Die Idee von Session Management ist, Aufrufe vom selben Client in den selben Kontext zu stellen. Dazu stellt der Server einen Identifier aus, das **Session Token**, und schickt es dem Client. Der Client speichert den Identifier und schickt ihn bei allen Folge Requests mit. Eine Web Session ist somit eine Sequenz von HTTP Requests und Responses, welche einem bestimmten User zugeordnet sind. Mit **Session** wird die Zeit bezeichnet, in welcher ein Benutzer eine Website benutzt.

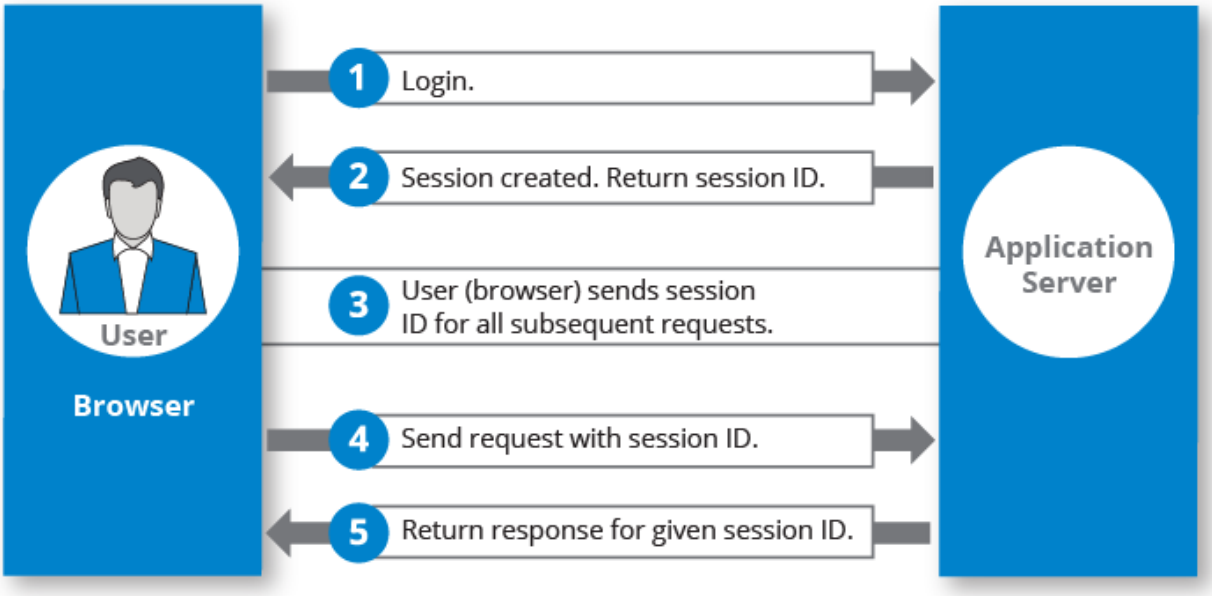

Abbildung 2: Web Session

In einer typischen Browser-Server Session speichert der Browser Informationen als key/value Paar als Cookies pro Domäne. Ein Server mit einer bestimmten Domäne kann nur auf seine Cookies zugreifen, nicht auf Cookies von anderen Domänen. Ein Server kann den Client Identifier

- über den Set-Cookie Response Header, oder
- im Response Body (JSON/XML)

setzen.
Der Client kann den Client Identifier

- über den Cookie Request Header
- im URL z. B. als SID (Session Identifier)

zurückschicken.
Ein User kann auch anonym sein, ohne Login, um eine Session mit dem Server aufzubauen.

1.3 HTTP/2

HTTP/2 ist eine stark überarbeitete Version von HTTP/1.1. Die Spezifikation wurde 2015 im RFC 7540 veröffentlicht. Die Folgeversion, HTTP/3 ist zur Zeit immer noch im Draft Statium, wird aber bereits von vielen Browsern unterstützt.
Die Hauptziele von HTTP/2 sind die Verringerung der Latenzzeit durch die Ermöglichung eines vollständigen Request- und Response-Multiplexing, die Minimierung des Protokoll-Overheads durch eine effiziente Komprimierung der HTTP-Header-Felder und die Hinzufügung von Unterstützung für die Priorisierung von Anfragen und Server-Push. Zur Umsetzung dieser Anforderungen gibt es eine grosse Anzahl weiterer Protokollverbesserungen, wie z. B. neue Flusskontrolle, Fehlerbehandlung und Upgrade-Mechanismen. Dies sind die wichtigsten Funktionen, die jeder Webentwickler verstehen und in seinen Anwendungen nutzen sollte.
HTTP/2 verändert die Semantik von HTTP in keiner Weise. Alle Kernkonzepte, wie HTTP-Methoden, Statuscodes, URIs und Header-Felder, bleiben erhalten. Stattdessen ändert HTTP/2 die Art und Weise, wie die Daten formatiert (framed) und zwischen dem Client und dem Server transportiert werden, die beide den gesamten Prozess verwalten, und die Komplexität von den Applikationen verbirgt innerhalb des neuen Framing Layers. Infolgedessen können alle bestehenden Applikationen ohne Änderungen weiter genutzt werden.

1.4 Web Applikationen

1.4.1 Model – View - Controller

Web Applikationen werden typischerweise nach dem MVC (Model – View - Controller) **Design Pattern** erstellt. Design Pattern sind Vorlagen, die bei immer wiederkehrenden Design Problemstellungen in der Softwareentwicklung, zur Lösung eingesetzt werden. Beim MVC wird eine Applikation mit GUI in drei Ebenen aufgeteilt, um damit ein flexibles und stabiles System zu erhalten.

- **M**odel: Daten und Business Logik
- **V**iew: Darstellung der Benutzerschnittstelle
- **C**ontroller: Steuerung der Applikation

Der Vorteil von MVC ist, dass der Code übersichtlicher, besser wartbar und wiederverwendbar wird. Da die Ebenen Model und View relativ unabhängig voneinander sind, können verschiedene Views mit demselben Model zusammenspielen (ohne, dass das Model das merkt), oder die verschiedenen Ebenen einfacher ausgetauscht werden.
Diese Architektur wird sowohl in Webapplikationen als auch in Client/Server-Applikationen eingesetzt.
Model: Das Model stellt sämtliche Zugriffe auf die Daten bereit. Es kommuniziert dazu mit dem Datei- und Datenbanksystem.
View: Die View steuert die Anzeige der Daten für den Benutzer und nimmt die Eingaben des Benutzers entgegen.

In einer Webapplikation enthält diese Schicht sowohl Teile die auf dem Server ausgeführt werden, als auch Teile die auf dem Client ausgeführt werden.
Controller: Der Controller steuert die ganze Verarbeitung. Er fordert beim Model die Daten an und steuert die benötigte View für die Interaktion mit dem Benutzer.
Je nach Umsetzung der MVC-Architektur wird die Business Logik im Model oder im Controller umgesetzt. Abbildung 3 zeigt eine generische MVC Architektur.

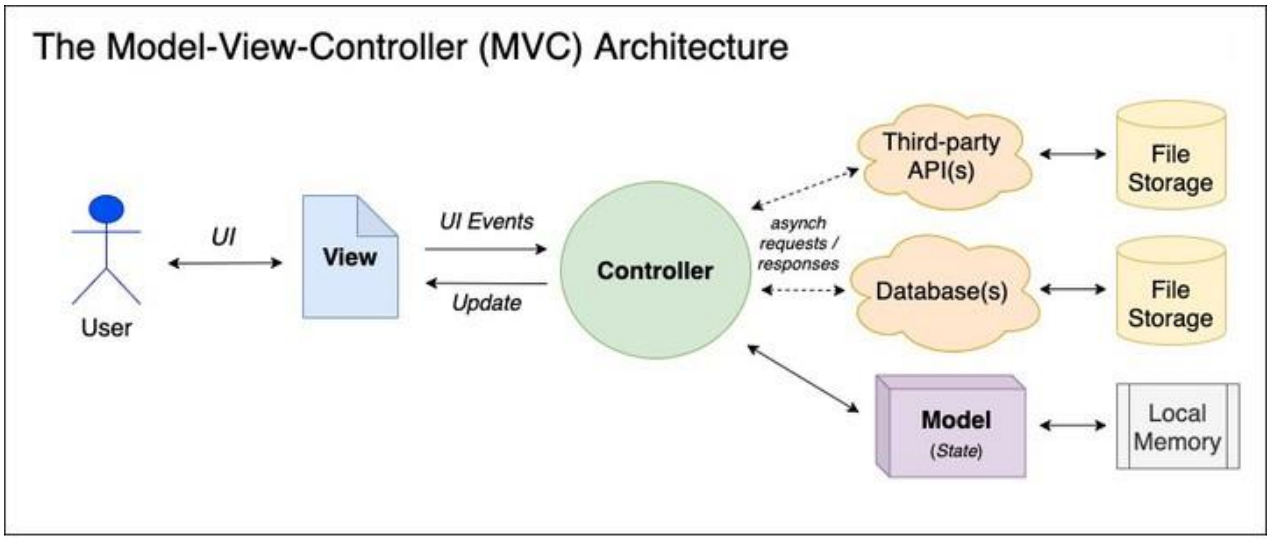

Abbildung 3: Model View Controller - Design Pattern

MVC ist ein generisches Design Pattern. Je nach eingesetzter Technologie wird es unterschiedlich implementiert.

1.4.2 HTML / CSS

HTML (Hypertext Markup Language) ist für die Struktur einer Webseite zuständig. In HTML können zum Beispiel Überschriften, Absätze, Texte und Bilder definiert werden. Hypertext bezeichnet die Links zwischen den Web Seiten. Markup wird verwendet um die Struktur der Web Seite mit Tags zu definieren.
Die aktuelle Version von HTML, HTML5, enthält zahlreiche Erweiterungen gegenüber HTML4, unter anderem für Multimedia Inhalte, Scalable Vector Graphics (SVG) und mathematische Formeln.
CSS (Cascading Style Sheets) ist für den Stil und das Layout einer Webseite zuständig. Mit CSS können Stile wie Farben, Schriftarten und Ränder definieren und sogar einfache Animationen erstellt werden.
Beide Sprachen, HTML und CSS, sind unabhängig und sollten in separaten Dateien gespeichert werden.
Mit HTML wird der Inhalt einer Web Seite definiert, während CSS das Styling des Inhalts definiert.
Mit HTML und CSS können sehr komplexe Websites erstellt werden. Diese Websites sind jedoch statisch, d. h. die Besucher können die Seiten zwar ansehen, haben aber keine Möglichkeit, mit ihnen zu interagieren (ausser durch Anklicken von Links).

Um dynamische Websites zu programmieren, die interaktiv sind, brauchen wir eine zusätzliche Sprache wie **JavaScript** oder **Java**. Mit diesen Sprachen können ganze Webanwendungen entwickelt werden, in denen die User zum Beispiel Charts zeichnen, ein Spiel spielen oder einen Chat nutzen können.
Klassische Hypertext Navigation mit HTML ist statisch. Benutzer können die Informationen auf den Seiten lesen, aber nicht mit der Seite interagieren. Informationen wie Börsenkurse oder Sportresultate, welche sich schnell ändern, konnten nur über einen Refresh der Web Page aktualisiert werden. Um dynamische Webseiten zu generieren, wurden die Web Technologien erweitert. Dynamische Web Seiten können sowohl auf dem Server, server-side, wie auch im Browser, client-side, gesteuert werden.
Eine dynamische server-side Web Page wird typischerweise über die CGI (Common Gateway Interface) Schnittstelle erstellt. Über CGI lässt sich aus dem Web Server ein externes Programm aufrufen um z. B. Daten aus einer Datenbank in der Web Page anzuzeigen. Abbildung 4 zeigt den Aufruf über CGI aus einem Web Server auf ein Backend System.

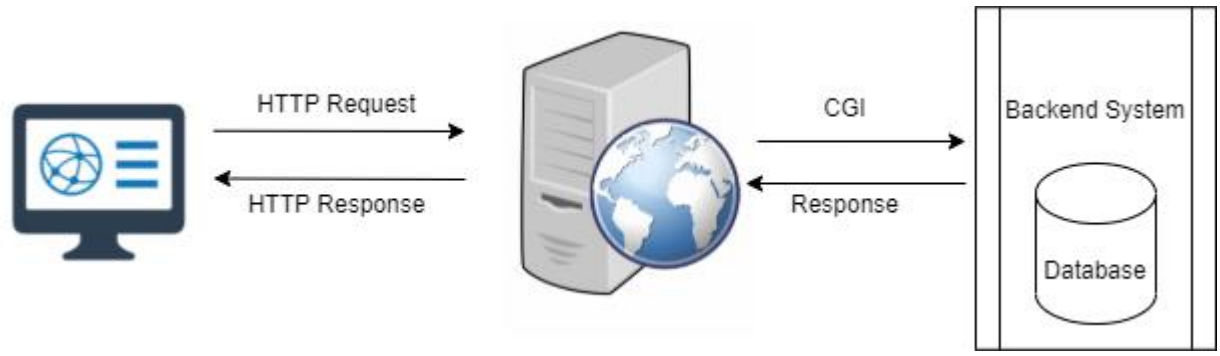

Abbildung 4: Server-side dynamische Web Seite

Typischerweise werden serverseitig Programmiersprachen wie JavaScript, Python, Rubi oder PHP verwendet, um dynamische Web Seiten zu erstellen. In der Praxis haben sich Software Bundles wie der LAMP Stack durchgesetzt, um dynamische Web Pages darzustellen. LAMP steht für **L**inux, **A**pache, **M**ySQL, **P**erl/**P**HP/**P**ython. Viele Linux Distributionen haben den Web Server Apache, die Datenbank MySQL und die Interpreter für Perl, PHP oder Python in ihrem Repository. Oft werden aber auch andere Web Server wie Nginx oder Datenbanken wie PostgreSQL eingesetzt. Der LAMP Stack bildet auch die Basis für viele Web Content Management Systeme (WCMS) wie Drupal, WordPress oder Joomla.

1.4.3 DOM

Dynamische client-side Web Pages benutzen JavaScript im Browser um Web Pages oder den Content dynamisch zu verändern. JavaScript kann mit dem **DOM** (Document Object Model) interagieren um Informationen in der Web Page abzufragen und die Page zu verändern. Ein DOM ist eine Repräsentation einer HTML Seite oder eines XML Dokuments als Baumstruktur (render tree). Abbildung 5 zeigt einen DOM einer Web Seite.

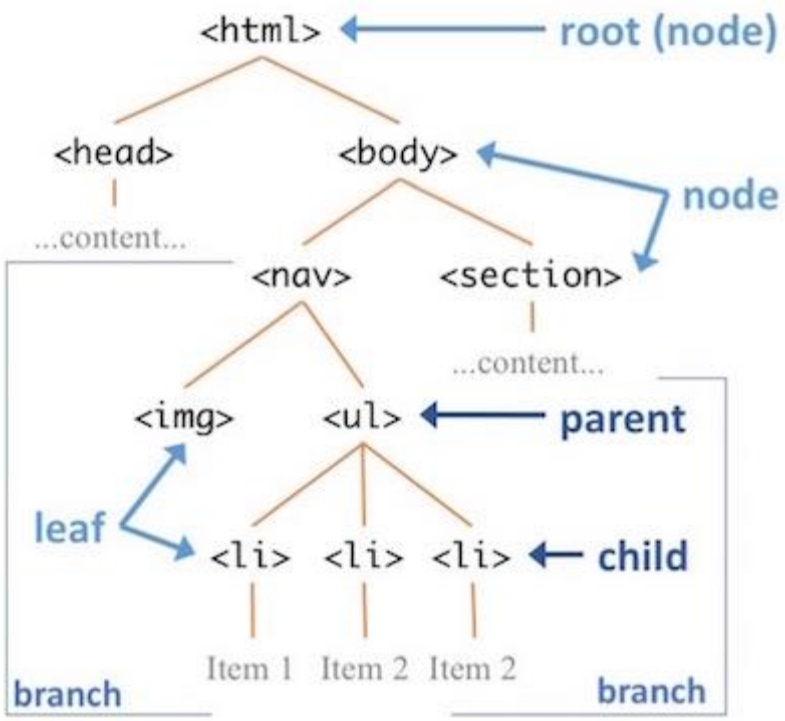

Abbildung 5: Document Object Model

Wird eine HTML Seite im Browser geladen, muss er zuerst die HTML Seite parsen, dann erstellt er den render tree.
Über den DOM lassen sich einzelne Elemente der Web Page, die Nodes, ansprechen und verändern.
Wie für HTML gibt es auch für CSS ein Objekt Model, das CSSOM (Cascading Style Sheets Object Model). Über CSSOM lässt sich ebenfalls mit JavaScript der CSS Style manipulieren.
Mit JavaScript kann auch über **AJAX** (Asynchronous JavaScript and XML) Content wie Wetterradar, welcher sich laufend ändert, nachgeladen werden ohne Refresh der ganzen Web Page. Mit AJAX lässt sich asynchron (im Hintergrund) Content nachladen, ohne dass die geladene Web Page neu geladen werden muss. So werden z. B. bei Eingabefeldern Vorschläge angezeigt, wenn man anfängt, einen Suchbegriff einzugeben (autocomplete).

1.5 Web Applikationen erstellen

Um eine Web Applikation zu entwickeln, benötigt man einen Texteditor um die Web Seiten zu erstellen und einen Browser, um die Website zu testen. Entwicklungsumgebungen wie Visual Studio Code oder IntelliJ IDEA vereinfachen aber die Entwicklung von Web Seiten erheblich mit Funktionen wie:

- Graphischem User Interface (GUI)
- Code completion
- Syntax Highlighting
- Refactoring
- Debugger
- Unit Test Framework

Um nur einige zu nennen. Hier verwenden wir Visual Studio Code. Es ist kostenlos und läuft unter Linux, Windows und macOS.

1.5.1 Visual Studio Code installieren

Visual Studio Code herunterladen:

```
https://code.visualstudio.com/#alt-downloads
```

Die Extension:

Live Server

Installieren. Die Live Server Extension ist ein lokaler Web Server, mit welchem statische und dynamische Seiten angezeigt werden können.

1.5.2 Aufbau einer HTML Seite

Eine HTML Seite besteht aus **Tags**. Ein HTML Tag besteht meistens, nicht immer, aus einem Start Tag und einem End Tag, z. B. `<html> … </html>`, wobei der End Tag einen Slash, / enthält. Zwischen den Tags befindet sich der Content.
Attribute deklarieren weitere Informationen für ein HTML Element. Sie bestehen aus einem Namen und einem Wert. Ein Beispiel für einen Tag mit einem Attribut ist in Abbildung 6 dargestellt.

Abbildung 6: HTML Element

Eine HTML Seite besteht im Allgemeinen im Minimum aus folgenden Elementen.

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Page Title in browser tab</title>
  </head>
  <body>
    <h1>Heading 1</h1>
    <p>A paragraph</p>
  </body>
</html>
```

Die einzelnen Elemente sind:

- In der ersten Zeile sollte immer `<!DOCTYPE html>` stehen. Damit wird dem Browser der Typ des Dokuments mitgeteilt.
- Der `<html>`-Tag kennzeichnet den Anfang und `</html>` das Ende des Dokuments.
- Der `<head>`-Tag enthält zusätzliche Informationen über die Seite. Im Gegensatz zum `<body>`-Tag erscheinen diese Informationen nicht im Hauptbereich des Browsers.
- Innerhalb des `<head>`-Tags sollte eine Angabe über den Zeichensatz, das Character Set, gemacht werden: `<meta charset="utf-8">`. Wenn das Character Set nicht angeben wird, werden einige Sonderzeichen möglicherweise nicht korrekt angezeigt.
- Der `<meta>`-Tag hat keinen schliessenden Tag. Es gibt einige Elemente ohne schliessenden Tag (`<br>, <img>` usw.), aber das ist die Ausnahme.
- Als nächstes steht das Element `<title>`. Der Titel wird in der Titelleiste am oberen Rand Ihres Browserfensters angezeigt.
- Alles innerhalb des `<body>`-Tags wird im Hauptbereich des Browsers angezeigt.
- Ein `<h1>` definiert die Hauptüberschrift. Unterüberschriften können mit `<h2>, <h3>, <h4>, <h5>` und `<h6>` erstellt werden.
- Text zwischen `<p>` und `</p>` ist ein Absatz, Paragraph.
- Nach jedem einleitenden Tag sollte das nächste Element zur besseren Übersichtlichkeit eingerückt werden (mit einem Tab oder zwei Leerzeichen).

Diese Basisstruktur kann für alle Web Pages verwendet werden. Um der Seite ein Bild hinzuzufügen, wird der `img` Tag benutzt.

```html
<img src="img/samurai.jpg" alt="Picture of Samurai">
```

1.5.2.1 Absoulte und relative URLs

URLs werden für das `src`-Attribut von Bildern und auch für das `href`-Attribut von Links verwendet. Die URL gibt die "Adresse" einer Datei an (zum Beispiel eine andere Webseite oder ein Bild). Je nachdem, wo sich die Datei befindet, muss entweder eine relative oder eine absolute URL verwendet werden.
Wenn eine Datei Teil derselben Website ist, kann eine relative URL verwendet werden. Wie im obigen Beispiel dargestellt, ist dies nur der Name und der relative Pfad der Datei. Folgendes Beispiel zeigt relative und absolute URLs:

```
    <!-- Relative URLs -->
     <a href="image-gallery.html">Image Gallery</a>
     <a href="blog/first-blog-entry.html">My First Blog
Entry</a>
     <a href="../image-gallery.html">Back to Image
Gallery</a>

     <!-- Absolute URLs -->
     <a href="http://www.my-colleague.com/blog.html">Blog of
a Colleague</a>
```

1.5.3 CSS - Cascading Sytle Sheets

HTML beschreibt die Struktur einer HTML Seite, CSS beschreibt die visuelle Darstellung. CSS kann direkt in der HTML Seite definiert werden, es hat aber Vorteile, HTML und CSS in separaten Files zu definieren.
Um ein CSS in einer HTML Page zu referenzieren, wird das `Link` Element im `head` Block verwendet:

```
<head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="main.css">
   <title>Samurai page</title>
</head>
```

Ein CSS File, welches die Hintergrundfarbe eines HTML Elements ändert,z. B. den h2 Header, kann so ausssehen:

```
h2 {
    background-color: #607d8b;
    color: #ffffff;
}
```

Eine CSS Regel besteht aus drei Elementen: einem **Selektor**, einem **Property** und einem **Wert**.

Abbildung 7: CSS Syntax

Selektoren definieren, für welchen Bereich in der HTML Seite die CSS Rule angewendet werden soll. Ein Selektor, der auf alle `p` Elemente, Paragraph, angewendet werden soll, sieht so aus:

```
p {
  ...
  }
```

Wird auf alle HTML Elemente

```
<p>....</p>
<p>....</p>
```

angewendet.

2 JavaScript

JavaScript ist eine leichtgewichtige interpretierte Programmiersprache. JavaScript wurde entwickelt, um netzwerkorientierte Anwendungen zu programmieren. JavaScript ist offen und cross-platform (plattformunabhängig).
Moderne Browser wie Chrome oder Firefox enthalten einen JavaScript Interpreter. JavaScript kann in HTML Seiten verwendet werden, um die Web Seiten interaktiver zu machen oder Eingaben zu überprüfen, z. B. ob eine Email Adresse ein gültiges Format hat oder ob Pflichtfelder ausgefüllt sind. JavaScript kann aber auch serverseitig benutzt werden. JavaScript Frameworks wie node.js oder Express.js wurden für die serverseitige JavaScript Programmierung Entwickelt.
JavaScript wird am häufigsten Clientseitig eingesetzt. Das Script wird direkt in eine HTML Seite integriert oder Referenziert. Dadurch ist die HTML Seite nicht mehr statisch und kann mit dem User interagieren.

2.1 Vorteile von JavaScript

Vorteile von JavaScript sind:

- Weniger Interaktion mit dem Server, weniger Netzwerkverkehr, weniger Serverlast
- Sofortiger User Feedback, ohne auf eine Antwort vom Server warten zu müssen
- Interaktivere Web Seiten, z. B. wenn ein Benutzer mit der Maus über eine Web Seite fährt
- Mehr Web Seitenfunktionalität wie drag-and-drop oder sliders

2.2 Limiten von JavaScript

Limiten von JavaScript sind:

- Kein File Zugriff, aus Sicherheitsgründen kann nicht auf lokale Files zugegriffen werden
- Kein Multithreading oder Multiprocessing Support
- Kein Networking Support

2.3 JavaScript Basics

JavaScript Statements werden zwischen den `<script> ... </script>` HTML Tags eingefügt:

```
<script language = "javascript" type = "text/javascript">
         JavaScript Code
</script>
```

Der `<script>` Tag benachrichtigt den Browser, dass er den Interpreter startet. `language` spezifiziert die Script Sprache, wobei neuere Versionen von HTML dieses Attribut nicht mehr unterstützen. `type` ist das heute Empfohlenen Attribut und sollte auf `text/javascript` gesetzt werden.

JavaScript kann irgendwo in der Web Seite stehen. Es ist aber im Allgemeinen empfehlenswert, das Script im Header `<head>` zu programmieren.
Ein JavaScript in einer Website mit dem HTML Code kann so aussehen:

```
<html>
   <body>
      <script language = "javascript" type =
"text/javascript">
         //<!--
            document.write("Hello brave new World!")
         //-->
      </script>

      <noscript>
         Sorry...JavaScript is needed to go ahead.
      </noscript>
   </body>
</html>
```

Es schreibt `Hello brave new World!` auf die Web Seite. Der HTML Kommentar Tag `//<!- - ... //-->` verhindert Probleme mit Browsern, welche kein JavaScript unterstützen.
Am Ende eines Statements kann optional ein Semikolon stehen.
JavaScript ist case sensitive.
Kommentare können mit `//` gekennzeichnet werden, mehrzeilige Kommentare zwischen `/* ... */`.
Für Browser, welche kein JavaScript unterstützen, kann der `<noscript>` Tag verwendet werden. Dann wird die zwichen den `<noscript> ... </noscript>` Message angezeigt.

2.3.1 JavaScript Positionierung

JavaScript kann grundsätzlich irgendwo in der HTML Seite definiert werden, im Header, im Body oder in einem externen File. Wenn das JavaScript beim Laden der Web Seite ausgeführt werden soll, wird es in den Body positioniert, wie im obigen `Hello brave new World!` Beispiel.
Wenn ein JavaScript bei einem bestimmten Event ausgeführt werden soll, z. B. wenn der Benutzer auf einen Button klickt, kann im Header eine JavaScript Funktion wie folgt definiert werden, welche beim Mouse Click Event aufgerufen wird:

```
<html>
   <head>
      <script type = "text/javascript">
         //<!--
            function sayHello() {
               alert("Hello World")
            }
         //-->
      </script>
   </head>

   <body>
      <input type = "button" onclick = "sayHello()" value =
"Say Hello" />
   </body>
</html>
```

Wenn viel JavaScript Code verwendet wird, kann es schnell vorkommen, dass Code mit identischer Funktionalität mehrmals geschrieben wird. Um Redundanz zu vermeiden, kann der Code in ein externes File ausgelagert werden. Es empfiehlt sich, Scripts in einem separaten Verzeichnis abzulegen, z. B. `scripts/`.
Der HTML Code sieht dann so aus:

```
<html>
   <head>
      <script type = "text/javascript"
src="scripts/helloexternal.js"></script>
   </head>

   <body>
      <input type = "button" onclick = "sayHello()" value =
"Say Hello" />
   </body>
</html>
```

Das `helloexternal.js` enthält die JavaScript Funktion:

```
function sayHello() {
    alert("Hello World")
 }
```

2.3.2 Data Types

JavaScript ist eine dynamic typed Sprache. Im Gegensatz zu strongly typed Sprachen wie Java oder C/C++, müssen Variabeln nicht deklariert werden. JavaScript unterstützt folgende primitive Data Types:

- **Numbers**, z. B. 17, 10.5 etc.
- **Strings**, z. B. «This is a text string»
- **Booleans**, true oder false

Zusätzlich werden die Datentypen **`null`** und **`undefined.`** JavaScript unterscheidet nicht zwischen ganzen Zahlen, integer, und Gleitkommazahlen, float.
Um eine Variable zu definieren, wird das key word **`var`** verwendet:

```
var a = 42; //holding number
var b = "Darth Vader"; //holding string
```

2.3.3 Nicht primitive Data Types

JavaScript unterstützt folgende nicht primitive Datentypen **`Array, Object`** und **`RegEx.`**
In einem Array können mehrere Variablen des gleichen Typs gespeichert werden.
Der Syntax um ein Array zu erstellen ist wie folgt:

```
const cars = ["Tesla", "Ferrari", "Maserati"];
```

Die Elemente des Arrays können über den Index abgerufen werden:

```
const cars = ["Tesla", "Ferrari", "Maserati"];
let x = cars[0]; //x = “Tesla”
```

Die Array Elemente können auch über den Index geändert werden:

```
cars[0] = "Mustang"; //x = “Tesla”
```

Um durch ein Array zu iterieren, kann ein `for` Loop verwendet werden:

```
const stars = ["Beta Centauri", "Sirius", "Antares",
"Betelgeuse"];
let sLen = stars.length;
for (let i = 0; i < sLen; i++) {
    document.write(stars[i] += "<br>")
}
```

Arrays in JavaScript sind Objekte, welche Methoden enthalten, unter Anderem eine `toString()` und die `forEach()` Methode, um durch ein Array zu iterieren:

```
const greekgoods = ["Castor", "Pollux", "Arethusa",
"Dryade"];
greekgoods.forEach(myFunction);
function myFunction(value, index, array) {
    document.write(value += "<br>")
}
```

JavaScript Objekte bestehen aus Object Properties und Object Methods. Ein JavaScript Objekt wird erstellt mit:

```
const person = {
    firstName: "John",
    lastName: "Doe",
    age: 50,
    eyeColor: "blue"
};
```

Es enthält 4 Properties, `firstName`, `lastName`, `age`, und `eyeColor`. Properties können über den Namen angesprochen werden:

```
document.write(person.firstName + " " + person.lastName)
//or
document.write(person["firstName"] + " " +
person["lastName"])  //equivalent to first line
person["firstName"] = "Jane" //firstName = “Jane”
```

Objekte können auch Methoden enthalten. Methoden oder Funktionen sind Aktionen, welche mit Objekten ausgeführt werden können. Methoden werden in Properties als Funktionsdefinitionen gespeichert. Eine Methode in einem Objekt kann so aussehen:

```
const person = {
     firstName: "John", // Property and value
     lastName: "Doe",
     age: 50,
     eyeColor: "blue",
     fullName: function () { // Property and function
definition
                 return this.firstName + " " +
this.lastName;
     }
};
```

Das `this` Keyword bezeichnet den Owner der Funktion. In diesem Beispiel bezeichnet `this` das person Objekt, welches die Properties `firstName` und `lastName` besitzt.
Eine JavaScript Methode wird über den Objektnamen aufgerufen:

```
name = person.fullName()
```

Objekte können auch über das `new` Keyword erstellt werden:

```
x = new String(); // Declares x as a String object
z = new Boolean(); // Declares z as a Boolean object
```

Dies ist aber weniger effizient und sollte vermieden werden.

3 Anhang

3.1 Abkürzungen

Hier sind übersichtshalber die wichtigsten Abkürzungen aufgelistet:

Abkürzung	Ausgeschrieben
OOP	Object Oriented Programming Language
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
WWW	World Wide Web
War	Web Archive
Jar	Java Archive
NLP	Natural Language Processing

4 Referenzen

4.1 Referenzen

Beck, K., 2000. *Extreme programming explained: embrace change*. addison-wesley professional.

Evans, B. and Flanagan, D. 2018. Java in a Nutshell: A Desktop Quick Reference. O'Reilly Media.

Kernighan, B. W. and Ritchie, D. 1988, C Programming Language, Pearson Education.

Sedgewick, R. 2002, Algorithms in Java, Parts 1-4. Addison-Wesley Professional.

4.2 Web Referenzen

Diagrams.net: Security-first diagramming for teams, 2021
< https://www.diagrams.net/>

Extreme Programming: A gentle introduction, 2013
<http://www.extremeprogramming.org/>.

IETF - Internet Engineering Task Force, 2021
< https://www.ietf.org/>.

Java Class Library.
<https://docs.oracle.com/javase/8/docs/api/>.

Java Language Specification
<https://docs.oracle.com/javase/specs/index.html>.

W3C - World Wide Web Consortium, 2021
< https://www.w3.org/>.