

Website-Tutorial

(A Tiny Website Kit)

Hubert Winkel

Bearbeitungssand : 28.11.2025

Inhaltsverzeichnis

Vorbemerkung.....	3
Einleitung.....	4
Konzeptionelles.....	6
Layout.....	6
Styles.....	6
Content.....	7
Navigation.....	7
Modularisierung.....	7
Architektur.....	7
Voraussetzungen zum Tutorial.....	8
Tutorial: Layouts.....	9
Strukturierung mit HTML5.....	9
Gestaltung mit CSS3.....	11
Die Grenzen von HTML und CSS.....	16
Tutorial: Musikkatalog.....	17
Projektplan.....	17
Schrittweise Implementierung.....	18
Schritt 0.....	18
Schritt 1.....	24
Schritt 2.....	28
Schritt 3.....	33
Schritt 4.....	35
Schritt 5.....	39
Schritt 6.....	43
Schritt 7.....	48
Schritt 8.....	50
Fertig! Fertig?.....	53
Noch ein paar Übungsaufgaben.....	53
Apropos „Musik“.....	54
Bonus: Schritt 9.....	54
Anhang: CMS.....	59
„Ihre Website in 10 Minuten!“.....	59
Was genau ist ein CMS.....	59
Brauche ich ein CMS für meine Website?.....	59

Vorbemerkung

zur Entstehung dieses Website-Tutorials und seinem technologischen Stand.

Dieser Baukasten sowie das Konzept dazu sind etwa vor 10 Jahren im Rahmen der Entwicklung einer inzwischen sehr umfangreichen Website entstanden. Dieses Dokument und das Tutorial sind ein Ergebnis meines Bestrebens Ordnung in meinen gesammelten Wust aus Notizen und Code-Schnipseln aus dem Projekt zu bringen und in einer für mich selbst brauchbaren Form abzulegen.

Am meisten lernt man selbst, wenn man sich in die Situation versetzt, etwas jemand anderem beibringen zu wollen. Schön also, wenn auch jemand anderes mit diesem Baukasten etwas lernt. In erster Linie ist er aber aus Eigennutz entstanden. Und das erweist sich gerade jetzt als nützlich, wo ich nach langer Webentwicklungsabstinenz mich nun doch wieder damit befassen muss (und auch will) und ich feststelle, wie schnell man vergisst.

Der technologische Stand dieses Tutorials ist inzwischen nicht mehr *leading-edge*, denn die Technologie, Methoden und Tools haben sich rasant weiterentwickelt. Man würde heute vielleicht einiges anders konzipieren und umsetzen. Das hier zugrunde liegende Konzept dürfte aber immer noch hilfreich sein, um Webentwicklung grundsätzlich zu verstehen. Und der Code funktioniert auch heute noch – jedenfalls in meiner Umgebung. ;-)

Noch ein paar Hinweise bevor es losgeht:

1. *Code Reading im mitgelieferten Tutorial Code ist generell zu empfehlen. In dieses Dokument habe ich zwar viele Code Snippets eingefügt, um beim Lesen nicht ständig in die Code-Dateien schauen zu müssen, aber eben nicht den gesamten Code und letztlich erschließt sich das tiefere Verständnis für ein Programmsystems doch nur aus dem vollständigen Code.*
2. *Ich empfehle, die beschriebenen Schritte zunächst zu versuchen selbst zu erarbeiten, bevor man sich meinen Code anschaut. Sicher kommt Ihr dabei zu anderen oder gar besseren Lösungen. Im mitgelieferten Code ist jeder Schritt immer der Ausgangspunkt für den nächsten Schritt.*
3. *Ich stelle dieses Tutorial frei, aber ohne jegliche Gewähr und Service, als **feedbackware** zur Verfügung, d.h. ich bitte, mir per E-Mail an **tutorial@tinygiant.de** eine Rückmeldung zu geben, dass das Tutorial von meiner Website **tinygiant.de** runtergeladen wurde und ob es nützlich war.*

Einleitung

Hier gibt es ein kleines Tutorial zur Erstellung von Websites mit dem Schwerpunkt **Architektur**. Dabei wird ein Grundwissen über die **Basistechnologien** der Website-Entwicklung (*HTML*) und -Gestaltung (*CSS*) vorausgesetzt, sowie über die **fortgeschritteneren Technologien** PHP und MySQL. Auf die Installation dieser Komponenten wird hier nicht eingegangen, dazu gibt es reichlich Anleitungen im Internet (s.u.z.B.: *Peter Kropff*).

Zum Einstieg und zur Vertiefung von HTML kann das online verfügbare [HTML5-Handbuch](#) von *Stefan Münz* dienen, oder das [SelfHTML-Wiki](#), dessen Wurzeln ebenfalls bei *Stefan Münz* liegen.

Zu jedem Teilbereich der Webentwicklung findet man reichlich Tutorials im Internet. Sowohl die Grundlagen als auch fortgeschrittenes Wissen zu allen wichtigen Webtechnologien gibt es z.B. auf der sehr umfangreichen [Website von Peter Kropff](#). Da findet man alles Wesentliche zu HTML, CSS, JavaScript, PHP und MySQL für Anfänger und Fortgeschrittene unter einem Dach.

Recherchen mit einer Suchmaschine führen zu unzähligen weiteren Quellen.

Bei meinen Recherchen hatte ich damals aber keine befriedigende Quelle gefunden, die mir einen systematischen Einstieg zur Software-Architektur einer Website gegeben hätte. Die meisten Veröffentlichungen legen ihren Schwerpunkt auf Design- und Implementierungstechniken, wobei die Gesamtarchitektur meiner Meinung nach meist zu kurz kommt. Selbst wenn man sich mit den notwendigen Technologien hinreichend gut auskennt, kann man konzeptionell und strukturell beim Auf- und Ausbau seiner Website scheitern und nach einiger Zeit in seinem Code-Sumpf versinken. Man lernt es dann schließlich auf die harte Tour.

Um das zu vermeiden, soll dieses Tutorial einige grundsätzliche Hilfestellungen geben.

- **HTML, CSS**

Auf HTML als Auszeichnungssprache und CSS als Gestaltungssprache und wie beide funktionieren und zusammenspielen *wird hier nicht eingegangen*. Hier geht es um die architektonischen Aspekte, also um die Vorgehensweise, eine ansehnliche Website zu bauen, die strukturiert und modular weiterentwickelt werden kann. HTML und CSS reichen dazu aus, solange eine rein statische Website genügt, also z.B. zur Präsentation eines einfachen Personen- oder Firmenprofils oder zur Veröffentlichung eines kleinen Reiseberichts. Die Inhalte, einmal erstellt, ändern sich dann nur wenig und relativ selten.

- **PHP (OOP)**

Auch PHP, die serverseitige Programmiersprache, *soll hier nicht gelehrt werden*. Hier soll gezeigt werden, wie man mit PHP Programmbausteine erstellt und wie man sie dann zur Dynamisierung von Webseiten nutzt. An diesem Punkt ist man i.d.R. angekommen, sobald man Eingaben erfassen und bearbeiten will. Inzwischen unterstützt PHP die objekt-orientierte Programmierung (OOP) ausreichend gut und das sollte daher die bevorzugte Technik sein.

- **MySQL**

Sobald man seine Website mit Dynamik und Funktionalität anreichert, kommt man auch an den Punkt, wo man eine geeignete persistente Datenhaltung benötigt. Für Eingaben in einen Fragebogen und Anzeigen einfacher Auswertungen kommt man vielleicht noch mit Datendateien aus. Besser geht es aber auf jeden Fall mit einer Datenbank wie MySQL und sobald man es mit Daten zu tun hat, die miteinander in Beziehung stehen (z.B. in einem Shop), kommt man daran sowieso nicht mehr vorbei. Die tiefgehende Funktionsweise von MySQL *wird hier nicht behandelt*. Es soll aber gezeigt werden, wie Datenbankzugriffe in PHP gekapselt werden können.

- **Javascript**

Die clientseitige Programmiersprache Javascript dient der Dynamisierung der Website im Browser. Damit lassen sich viele Effekte und Funktionen realisieren, ohne dass der Webserver beteiligt ist. Das können einfache visuelle Effekte wie dynamische Buttons und Bildanzeigen sein, das können aber auch aufwendigere Prüfungen von Formulareingaben oder wesentlich komplexere Funktionen sein. *Javascript wird hier aber (fast) kein Thema sein.*

Im praktischen Teil des Tutorials werden im ersten Teil die Grundlagen zur Layout-Entwicklung behandelt. Im zweiten Teil folgt dann die schrittweise Entwicklung einer Beispielanwendung, wobei es sich um eine typische Listenverwaltung handelt. Diese soll als roter Faden für die eigene Vertiefung der Web-Technologien dienen. Auf die Beschreibung zu vieler Details wird dabei verzichtet. „*Der Weg ist das Ziel!*“

Den Code für alle Implementierungsschritte gibt es als Download. Er ist die eigentliche Dokumentation des hier Beschriebenen, sollte also gründlich studiert werden, wenn man alle Details verstehen möchte. Es sei auch noch angemerkt, dass unsere Entwicklungsmethode und der entstandene Code nur *eine* Möglichkeit ist. Es gibt in der Software-Entwicklung immer andere und bessere Wege, die jeder mit wachsendem Kenntnisstand selbst findet.

Konzeptionelles

Wir beginnen mit ein paar konzeptionellen Anmerkungen zu den wichtigsten Grundbegriffen für die Entwicklung einer Website:

- Layout und Styles – Aufteilung und Gestaltung der Website
- Content und Navigation – Inhalt und Wegführung durch eine Website
- Modularisierung und Architektur – Bausteine und Bauprinzip einer Website

Ganz egal, was das Thema der Website sein wird, sie wird Inhalt haben, der Inhalt muss strukturiert werden, er wird in einem Layout präsentiert werden, er wird im Laufe der Zeit mehr werden. Zu den Inhalten muss navigiert werden können und es wird sich dabei nicht nur um Text handeln, es werden Bilder, evtl. Videos dazu kommen und es wird der Wunsch nach Funktionalität dazu kommen. Meist beginnt das mit einem Kontaktformular und entwickelt sich dann zu komplexeren Dingen.

Layout

Was auch immer die Website inhaltlich bietet, sie benötigt eine Grundstruktur. Da ist zunächst einmal ein ordnendes Layout: Kopf, Rumpf, Fuß. Den Hauptteil stellt der Rumpf dar, der klassischerweise in einen, zwei oder drei Bereiche unterteilt wird und der meist zwischen einem Kopf- und Fußbereich eingefasst ist. Die Grafik skizziert die klassischen Layouts.

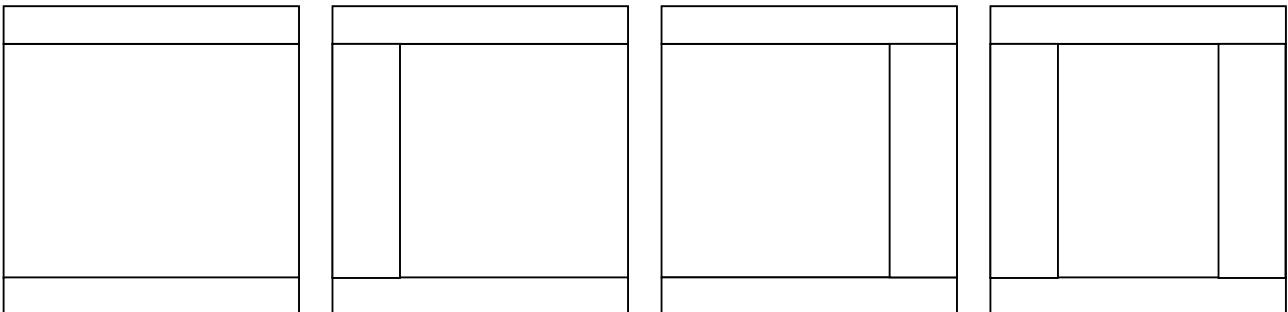


Abb. 0: Klassische Layouts

Styles

Die Blöcke in Abb. 1 lassen sich zwar in HTML definieren, das ist aber eine rein abstrakte Beschreibung, die dem Browser keine Information zur Anzeige liefert. Positionieren kann er die Blöcke allein mit HTML nicht. Dazu benötigt er Gestaltungsanweisungen, sogenannte *Styles*, die mit CSS formuliert werden.

Content

Sehen wir Layout und Styles als das *Skelett* einer Website, so ist der Content das *Fleisch*. Trotz allen Beiwerks ist der Inhalt die eigentliche Information, woran der Besucher interessiert ist. Diese Information wird ihm seitenweise, in geeigneten Portionen aufbereitet, dargeboten. Die Anzahl der Seiten einer Website richtet sich nach dem Umfang des Informationsangebot.

Navigation

Sofern die Website aus mehr als einer Seite besteht, benötigt der Besucher eine *Navigation*, um sich von einer Seite zu einer anderen bewegen zu können. Je nach gewähltem Basis-Layout kann die Navigation in den Kopf, in die linke oder in die rechte Spalte gesetzt werden. Das waren zumindest bisher die üblichen Varianten. Inzwischen ist das Thema Navigation komplexer geworden und es werden Untergliederungen in Haupt- und Nebennavigationen üblich, wobei die Hauptnavigation in den Kopfbereich oder in die linke oder rechte Spalte gesetzt wird, und z.B. eine Nebennavigation zu Standardseiten wie Impressum, Kontaktformular oder Sitemap in den Fußbereich. Darauf wollen wir uns hier beschränken, wohl wissend, dass durch den Vormarsch der mobilen Geräte viele neue Spielarten hinzugekommen sind. Überhaupt haben die mobilen Geräte jede Menge neuer Facetten für die Website-Entwicklung mit sich gebracht, die wir hier aber außer Acht lassen.

Modularisierung

Wie die vorigen Abschnitte zeigen, spielen mehrere verschiedene Konzepte zusammen, damit eine Website so funktioniert, wie es die Besucher gewohnt sind. Daher ist es sehr wichtig, dass man geeignete, handhabbarer Bauteile hat, um eine Website systematisch entwickeln zu können und um nicht bei wachsender Größe irgendwann im Chaos zu versinken. Dieses Tutorial soll eine Vorgehensweise aufzeigen, wie man zu geeigneten Bauteilen kommt und wie man sie im Laufe fortschreitender Entwicklung anpasst oder ausbaut.

Der Vorgang heißt *Modularisierung*, die Bauteile heißen *Module*.

Architektur

Was nun noch fehlt ist ein Bauprinzip, nach dem die Bauteile zusammengesetzt werden. Auch das ist Thema dieses Tutorials. Alles in allem spricht man dann bei der Software-Entwicklung von *Architektur*.

Voraussetzungen zum Tutorial

Um den praktischen Teil des folgende Tutorials durchführen zu können, benötigt man eine lokale Webentwicklungsumgebung. Das ist zunächst ein Softwarepaket bestehend aus Webserver, PHP und MySQL. Für Windows empfehle ich dafür das kostenlose Paket UwAMP, da es ohne aufwendige Installation funktioniert: runterladen, auspacken und starten:

UwAMP: <http://www.uwamp.com/en/>

Für alle Umgebungen (Windows, Linux, Mac) kann man XAMPP nehmen:

XAMPP: <https://www.apachefriends.org/de/download.html>

Zum Editieren kann man im Prinzip jeden beliebigen ASCII-Editor nehmen, ein wenig Komfort bei der Arbeit macht's allerdings deutlich angenehmer. Es gibt viele Alternativen, IDE grafisch oder textbasiert, mit unterschiedlich hohen Einstiegshürden, z.B.: Visual Studio Code (VS Code), NetBeans, Eclipse PDT oder CodeLite. Es sollte für jeden Geschmack etwas zu finden sein.

Tutorial: Layouts

Strukturierung mit HTML5

Beginnen wir mit der praktischen Umsetzung unserer konzeptionellen Überlegungen. Obwohl wir uns hier nicht tiefergehend mit HTML und CSS beschäftigen wollen, mag es gut sein, ein paar ihrer Grundeigenschaften zu demonstrieren und dadurch besser zu verstehen. Ganz wichtig dabei ist der wesentliche Unterschied, dass HTML für Struktur und CSS für das Layout und Styling zuständig ist. Vereinfacht kann man sagen: HTML legt fest, *was* die einzelnen Bereiche und Elemente sind, während CSS festlegt, *wie* sie angezeigt werden sollen.

Wie erstellt man eine Seite mit einer Kopf- und Fußzeile, einer Navigationspalte und einem Hauptbereich für den Inhalt. Wir beschreiben diese Struktur mit HTML5, das passende Tags (Bezeichner) dafür bereit stellt: `<header>`, `<footer>`, `<nav>` und `<main>`.

Frühere HTML-Versionen hatten keine speziellen Tags für diese Bereiche, sondern man behalf sich dort mit dem allgemeinen Tag `<div>` und gab ihm über das `id`-Attribut einen entsprechenden Namen, also `<div id="header">`, `<div id="footer">`, usw. Wir werden in unseren Beispielen aber durchgehend **HTML5** nutzen.

HTML liefert eine reine Strukturbeschreibung für den Browser, die für den menschlichen Benutzer der Seite nicht direkt sichtbar ist. Dem Betrachter wird diese Struktur erst sichtbar gemacht, indem man den Strukturblöcken Inhalte und sichtbare Eigenschaften zuordnet. Die vier genannten Strukturelemente als HTML-Datei zeigt das Listing 1. Öffnet man sie im Browser, so sieht man zunächst ... *nichts*. Erst wenn man sich den Quelltext im Browser anzeigen lässt, sieht man den Code der Seite, d.h. der Browser hat ihn korrekt geladen und interpretiert. Anzuzeigen hat er durch diesen Code alleine noch nichts, denn weder Anzeige-Attribute noch Inhalte sind in diesem Code enthalten.

```
1  <!DOCTYPE html>
2  <html lang="de">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="description" content="This is the LoremIpsum Tutorial 0.0">
6      <title>LoremIpsum Tutorial 0.0</title>
7  </head>
8  <body>
9      <header></header>
10     <main></main>
11     <nav></nav>
12     <footer></footer>
13 </body>
14 </html>
```

Listing 1: index-00.html

Das ändert sich mit Listing 2, in dem wir ein paar Inhalte in die einzelnen Bereiche einfügen. Nun werden zumindest alle Inhalte im Browserfenster angezeigt, allerdings ohne dass das Layout besonders aufregend aussieht. Die einzelnen Bereiche werden einfach sequentiell nacheinander angezeigt. Von Spalten ist noch nichts zu erkennen. Der Browser zeigt aber standardmäßig schon mal ein gewisses Styling für Überschriften, Aufzählungen und den Adresseintrag in der Fußzeile.

```

1  <!DOCTYPE html>
2  <html lang="de">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="description" content="This is the LoremIpsum Tutorial 0.1">
6    <title>LoremIpsum Tutorial 0.1</title>
7  </head>
8  <body>
9    <header><h1>In Voluptate Velit</h1></header>
10   <main>
11     <h1>Lorem Ipsum</h1>
12     <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
13     sed eiusmod tempor incididunt ut labore et dolore magna
14     aliqua. Ut enim ad minim veniam, quis nostrud exercitation
15     ullamco laboris nisi ut aliquid ex ea commodi consequat.
16     Quis aute iure reprehenderit in voluptate velit esse
17     cillum dolore eu fugiat nulla pariatur. Excepteur sint
18     obcaecate cupidatat non proident, sunt in culpa qui officia
19     deserunt mollit anim id est laborum.</p>
20   </main>
21   <nav>
22     <h1>Officia</h1>
23     <ul>
24       <li><a href="#">navigare necesse est</a></li>
25       <li><a href="#">vivere non est necesse</a></li>
26       <li><a href="#">sed sine vita non navigamus</a></li>
27     </ul>
28   </nav>
29   <footer><address>&copy;2016 &middot; Julius Caesar &middot; Roma &middot;
30     Bella Italia</address>
31   </footer>
32 </body>
33 </html>

```

Listing 2: index-01.html



Abb. 1: Browser-Anzeige ohne Styles

Mit Listing 3 gehen wir noch ein Schrittchen weiter, indem wir die Bereiche `<main>` und `<nav>` in einen gemeinsamen Bereich `<section>` stecken und den Inhalt von `<main>` in einen Bereich `<article>`. Der Grund wird gleich ersichtlich, wenn wir die Seite mit CSS gestalten, denn dann können wir bedarfsgerecht größeren oder kleineren Strukturelemente Anzeige-Attribute (Styles) zuordnen, was eine flexiblere Gestaltung ermöglicht.

Gestaltung mit CSS3

Um nun unsere Strukturblöcke endlich sichtbar zu machen, weisen wir den Browser außerdem an, die Layout-Anweisungen aus der Datei `styles-02.css` zu verwenden.

```
1 <!DOCTYPE html>
2 <html lang="de">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="description" content="This is the LoremIpsum Tutorial 0.2">
6   <title>LoremIpsum Tutorial 0.2</title>
7   <link rel="stylesheet" href="css/styles-02.css"/>
8 </head>
9 <body>
10  <header><h1>In Voluptate Velit</h1></header>
11  <section>
12    <main>
13      <article>
14        <h1>Lorem Ipsum</h1>
15        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
16        sed eiusmod tempor incididunt ut labore et dolore magna
17        aliqua. Ut enim ad minim veniam, quis nostrud exercitation
18        ullamco laboris nisi ut aliquid ex ea commodi consequat.
19        Quis aute iure reprehenderit in voluptate velit esse
20        cillum dolore eu fugiat nulla pariatur. Excepteur sint
21        obcaecat cupiditat non proident, sunt in culpa qui officia
22        deserunt mollit anim id est laborum.</p>
23      </article>
24    </main>
25    <nav>
26      <h1>Officia</h1>
27      <ul>
28        <li><a href="#">Lorem</a></li>
29        <li><a href="#">Ipsum</a></li>
30        <li><a href="#">Dolor</a></li>
31      </ul>
32    </nav>
33  </section>
34  <footer><address>&copy;2016 &middot; Julius Caesar &middot; Roma &middot;
35    Bella Italia</address>
36  </footer>
37 </body>
38 </html>
```

Listing 3: index-02.html

Um nun unsere Strukturblöcke endlich sichtbar zu machen, weisen wir den Browser außerdem an, die Layout-Anweisungen aus der Datei `styles-02.css` zu verwenden.

```
1  /* Definition des Seitenlayouts */
2  body {
3      border: 1px solid lightgray;
4      margin: 0 auto;
5      width: 960px;
6  }
7
8  header {
9      border: 1px solid lightgray;
10     min-height: 80px;
11     text-align: center;
12     vertical-align: middle;
13 }
14
15 header h1 {
16     padding-top: 20px;
17 }
18
19 section {
20     padding-left: 180px;
21 }
22
23 main {
24     background-color: white;
25     width: 100%;
26     min-height: 400px;
27 }
28
29 main h1 {
30     font-size: 18pt;
31 }
32
33 nav {
34     border: 1px solid lightgray;
35     margin-left: -100%;
36     min-height: 400px;
37     right: 180px;
38     width: 180px;
39 }
40
41 nav h1 {
42     font-size: 12pt;
43 }
44
45 main, nav, aside {
46     float: left;
47     position: relative;
48 }
49
50 footer {
51     border: 1px solid lightgray;
52     clear: both;
53     font-size: 8pt;
54     padding: 10px;
55     text-align: right;
56 }
```

Listing 4: styles-02.html

Und siehe da, im Browser erscheint ein zweispaltiges Layout mit Kopf- und Fußbereich und der Navigation links neben dem Hauptbereich für den eigentlichen Inhalt. Diese Gestaltung erhält die Seite durch die Anweisungen von *styles-02.css*, die im wesentlichen die Anzeigepositionen und -größen von `<header>`, `<footer>`, `<section>`, `<main>` und `<nav>` definieren sowie Rahmen um die Bereiche zeichnen. Ein Blick auf die Anweisungen sollte eine Vorstellung geben, was sie bewirken. Detailliertere Erläuterungen soll es hier nicht geben, dazu sei auf die früher genannten Quellen zu HTML und CSS verwiesen.

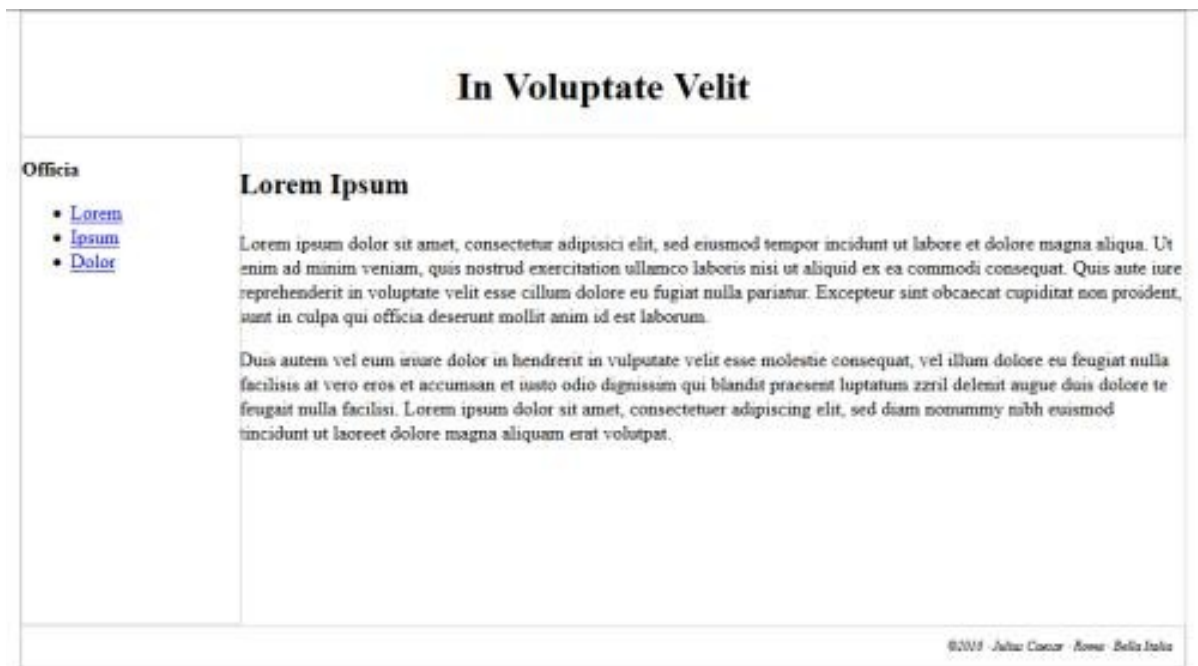


Abb. 2: zweispaltiges Layout, schmucklos

Für ein dreispaltiges Layout fügen wir ein weiteres Strukturelement `<aside>` in den HTML-Code ein und erweitern unsere Styles um Layout-Anweisungen für den neuen Bereich in *styles-03.css*.

```
1  <!DOCTYPE html>
2  <html lang="de">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="description" content="This is the LoremIpsum Tutorial 0.3">
6      <title>LoremIpsum Tutorial 0.3</title>
7      <link rel="stylesheet" href="css/styles-03.css"/>
8  </head>
9
10 <body>
11     <header><h1>In Voluptate Velit</h1></header>
12     <section>
13         <main>
14             <article>
15                 <h1>Lorem Ipsum</h1>
16                 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
17                 sed eiusmod tempor incididunt ut labore et dolore magna
18                 aliqua. Ut enim ad minim veniam, quis nostrud exercitation
19                 ullamco laboris nisi ut aliquid ex ea commodi consequat.
20                 Quis aute iure reprehenderit in voluptate velit esse
21                 cillum dolore eu fugiat nulla pariatur. Excepteur sint
22                 obcaecat cupiditat non proident, sunt in culpa qui officia
23                 deserunt mollit anim id est laborum.</p>
24             </article>
25         </main>
26         <nav>
27             <h1>Officia</h1>
28             <ul>
29                 <li><a href="#">Lorem</a></li>
30                 <li><a href="#">Ipsum</a></li>
31                 <li><a href="#">Dolor</a></li>
32             </ul>
33         </nav>
34         <aside>
35             <h1>What is Lorem Ipsum?</h1>
36             <p>Lorem Ipsum is simply dummy text of the printing and
37             typesetting industry. Lorem Ipsum has been the industry's
38             standard dummy text ever since the 1500s, when an unknown
39             printer took a galley of type and scrambled it to make
40             a type specimen book. It has survived not only five
41             centuries, but also the leap into electronic typesetting,
42             remaining essentially unchanged.</p>
43         </aside>
44     </section>
45     <footer><address>&copy;2016 &middot; Julius Caesar &middot; Roma &middot;
46         Bella Italia</address>
47     </footer>
48 </body>
49 </html>
```

Listing 5: index-03.html

Die Ansicht im Browser zeigt nun ein dreispaltiges Layout mit Kopf- und Fußbereich, die Navigation links, eine zusätzliche Spalte rechts und den Hauptbereich mittig dazwischen. Das soll an Beispielen zur Umsetzung der früher erwähnten klassischen Website-Strukturen reichen. Die Beispiele lassen sich relativ schnell ändern, um ein zweispaltiges Layout mit Navigation rechts zu bekommen. Das ist sicher eine gute kleine Übungsaufgabe.

In Voluptate Velit		
Officia <ul style="list-style-type: none"> • Lorem • Ipsum • Dolor 	Lorem Ipsum <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquid ex ea commodo consequat. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p> <p>Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>	What is Lorem Ipsum? <p>Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged.</p>
©2016 · Julius Caesar · Roma · Bella Italia		

Abb. 3: dreispaltiges Layout, schmucklos

Unsere Beispiele zeigen nun zwar die jeweils gewünschte Aufteilung und Anordnung der Seite, hübsch ist das allerdings so nicht. Eine optische Verschönerung der Seite erreicht man durch erweiterte bzw. zusätzliche CSS-Anweisungen. So verwandeln wir die Ansicht aus Abb. 2 mit erweitertem CSS in die Ansicht aus Abb. 4 ohne die HTML-Datei zu verändern. Das Aussehen einer HTML-Datei kann also komplett verändert werden, allein durch Austausch der benutzten CSS-Datei.

Wenn man mal staunen und sich inspirieren lassen will, was alles mit CSS möglich ist, so besuche man die Website <https://csszengarden.com/tr/de/>

In der Datei *styles-10.css* können die zugehörigen Gestaltungsanweisungen studiert und mit der vorherigen Datei *styles-02.css* verglichen werden.



Abb. 4: *index-10.html* - zweispaltiges Layout, aufgehübscht

Die Grenzen von HTML und CSS

Allein mit *HTML* und *CSS* ist sehr viel möglich, sowohl strukturell und inhaltlich als auch gestalterisch. *HTML* ist von Natur aus statisch, wohingegen sich mit *CSS* bereits manch erstaunliche dynamische Effekte erzielen lassen, wie Ein- und Ausblenden von Elementen oder ganzen Bereichen, Ändern von Positionen oder Größen, Farb- oder Formänderungen von Seitenbereichen oder sogar ein Klappmenü. Die dynamischen Möglichkeiten nur mit *HTML* und *CSS* bleiben aber beschränkt, da das keine Programmiersprachen sind. Konstrukte für Berechnungen, Abfragen oder Schleifen sind nicht vorhanden.

Durch Hinzunahme von *Javascript* öffnet sich ein enormes zusätzliches Potential, gerade was die Dynamisierung von Websites betrifft. Im Unterschied zu *HTML* und *CSS* ist *Javascript* eine vollwertige Programmiersprache, die man allerdings erst mal zu beherrschen lernen muss. Aber dann sind der Phantasie kaum noch Grenzen gesetzt. *Javascript* ist eine client-seitige Programmiersprache, d.h. die Programme – Skripte genannt – werden auf dem PC im Browser ausgeführt. *Javascript* ermöglicht aber auch die Kopplung mit einem Server (Stichwort: *AJAX*), so dass komplexe Client-Server-Anwendungen erstellt werden können.

Eine serverseitige Programmiersprache ist *PHP*, die meistens im Duo mit der Datenbank *MySQL* zum Einsatz kommt. Beides soll auch im folgenden Tutorial genutzt werden, um die Basis für eine modulare Architektur zu implementieren und das Grundwissen für deren weiteren Ausbau zu vermitteln.

Tutorial: Musikkatalog

Nach dem Einstieg über Layouts sollen nun anhand eines kleinen Beispielprojekts die Themen Modularisierung und Architektur behandelt werden. In dem Mini-Projekt erstellen wir einen einfachen Katalog zur Verwaltung von Musikalben. Das Beispiel ist typisch für jeglicher Art von Listenverwaltung, statt Musikalben kann man so natürlich auch Bücher oder Adressen oder sonst was verwalten.

Nach der Einführung einer Rahmenstruktur (*Framework*) werden sukzessiv Module eingefügt, die Inhalte anzeigen. Zunächst kommen die Daten aus Dateien, später aus einer Datenbank. Bevor wir uns dem praktischen Teil zuwenden, skizzieren wir aber noch kurz den Plan, den wir mit unserem Tutorial verfolgen. Danach folgt Schritt für Schritt die Entwicklung der Anwendung.

Projektplan

Grundgerüst einer modularen Website

- Statische und dynamische Teile einer Website
- Bausteinentwicklung mit PHP
- Rahmenseite mit Navigation (Pseudo-Modul „Dummy“)

Evolution des Website-Baukastens mit PHP und Dateien

- Text Inline oder aus Datei (Modul „Start“)
- Trennung von Datenverarbeitung und Inhaltsanzeige (MVC, Templates)
- Listenanzeige aus CSV-Datei (Modul „Catalog“)
- Tabellenanzeige aus CSV-Datei (Modul „Catalog“)
- Bildergalerie aus Bilderverzeichnis (Modul „Gallery“)

Evolution des Website-Baukastens mit PHP und MySQL

- Schnittstelle zur DB
- Datenzugriffsklassen (DAO)
- Umbau Albenliste aus DB (Modul „Catalog“)
- Umbau Bildergalerie aus DB (Modul „Gallery“)
- Formulareingabe für neue Alben (Modul „Insert“)
- Ausbau Albenliste für Insert, Update und Delete (Module „Catalog“, „InsUpd“, „Delete“)
- Umbau zur CRUD-Komponente (Zusammenfassung der Module „Catalog“, „InsUpd“, „Delete“)

Schrittweise Implementierung

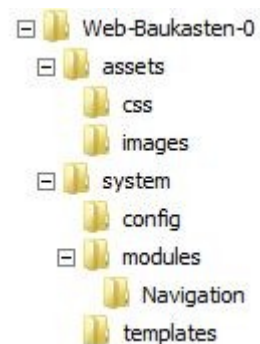
Bereits in der Vorbemerkung dieses Dokuments habe ich auf die Bedeutung des Code Readings hingewiesen. Bei der hier praktizierten schrittweisen Implementierung ist das besonders wichtig, da einige Code-Dateien bei jedem Schritt angepasst werden müssen, die hier im Text nicht jedes Mal in aller Ausführlichkeit erwähnt werden sollen. Das betrifft insbesondere den Orner `assets` und die Datei `index.php` und die Klasse `Navigation.php`.

Schritt 0

Das ist der Einstieg in das Tutorial zur bausteinartigen Entwicklung einer dynamischen Website. Hier wird das Fundament für unseren kleinen Baukasten („A Tiny Website Kit“) gelegt, auf dem alles Weitere aufbaut:

- die grundsätzliche Verzeichnisstruktur
- ein Template für zweispaltiges Layout mit Kopf- und Fußbereich
- ein Stylesheet zur Gestaltung der Website

Wir legen uns einen Projektordner in der Document Root unseres lokalen Webservers an und nennen ihn z.B. „Web-Baukasten-0“. Im Unterordner „assets“ liegen die Dateien, die eher layout-orientiert sind wie Bilder, CSS-Definitionen und Javascripts (Ressourcen), im Unterordner „system“ die eher anwendungsorientierten, also solche, die PHP-Code enthalten.



Auf oberster Ebene darin befindet sich die Datei `index.php`, die der Startpunkt für unsere Website ist. Diese `index.php` sieht deutlich anders aus als die Startdatei `index.html` aus dem Layout-Tutorial.

```

1  /* import and instantiate page and module classes */
2  require_once('system/Page.php');
3  $page = new Page();
4
5  /* import and instantiate the Navigation class */
6  require_once('system/modules/Navigation/Navigation.php');
7  $nav = new Navigation();
8
9  /* switch to selected page module */
10 parse_str($_SERVER['QUERY_STRING'], $param);
11 $module = (empty($param['module'])) ? 'Start' : $param['module'];
12
13 switch ($module){
14     default:
15         $mod = $page;
16 }
17
18 /* generate page */
19 require_once('system/templates/page_default.php');
```

Listing 0-1: `index.php`

Hier wird kein Layout definiert, sondern es handelt sich um PHP-Code zur Initialisierung der Website und zur Ansteuerung aufgerufener Module. Hier wird zunächst festgelegt, welche anderen PHP-Code-Dateien benötigt werden (`require_once`) und es werden einigen Variablen Anfangswerte zugewiesen (`$nav` Navigation, `$module` Name des aufgerufenen Moduls, `$mod` Klasse des aufgerufenen Moduls). Der genaue Zweck dieses Codes wird mit der fortschreitenden Entwicklung des Beispielprojektes klarer werden.

Das Layout wird in der Datei `page_default.php` definiert, welches die Grundstruktur unserer Website wie im **Tutorial zu Layouts** beschreibt. Allerdings ist der Code nun auch kein reiner HTML-Code mehr, sondern es finden sich nun PHP-Anweisungen an den Stellen zwischen den HTML-Tags, wo Inhalte dynamisch per Programm eingefügt werden sollen.

```
1  <!DOCTYPE html>
2  <html lang="de">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="description" content="<?php echo $mod->getDescription(); ?
>">
6      <meta name="author" content="Hubert Winkel">
7      <meta name="generator" content="www.tinygiant.de">
8      <title><?php echo $mod->getTitle(); ?></title>
9      <?php echo $mod->getStyles(); ?>
10     <?php echo $mod->getScripts(); ?>
11 </head>
12 <body>
13     <header><h1><?php echo $mod->getHeader(); ?></h1></header>
14     <section>
15         <main>
16             <div class="inside">
17                 <h1><?php echo $mod->getHeadline(); ?></h1>
18                 <?php echo $mod->getContent(); ?>
19             </div>
20         </main>
21         <nav>
22             <div class="inside">
23                 <?php echo $nav->getMenu(); ?>
24             </div>
25         </nav>
26     </section>
27     <footer><address><?php echo $mod->getFooter(); ?></address></footer>
28 </body>
29 </html>
```

Listing 0-2: page_default.php

Hier erkennen wir nun die Layout-Struktur wieder, wie wir sie im Layout-Tutorial erarbeitet haben. Mit der Anweisung `require_once('system/templates/page_default.php')` wird diese Struktur am Ende von `index.php` geladen. Ein Vergleich mit `index.html` aus dem Layout-Tutorial macht die analogen Konstruktionen deutlich.

PHP-Code wird in HTML-Code eingebettet, indem er in das Tag-Konstrukt `<?php ... ?>` gesetzt wird, wodurch der Webserver solche Einschübe vom PHP-Interpreter in HTML-Code umwandeln

lässt, bevor er sie dem Browser übergibt. In diesem Fall sind das alles Ausgabemethoden von Klasseninstanzen, die den Variablen `$mod` und `$nav` zugewiesen wurden (z.B. bei der Initialisierung in `index.php`). Diese Methoden liefern bei Aufruf der Seite dynamisch verschiedene Inhaltsblöcke.

Die Ausgabemethoden für die verschiedenen Inhaltsanteile sind alle in der wichtigen Basisklasse `Page.php` gebündelt. Sie ist sozusagen die Mutter aller Modulklassen von der die zu entwickelnden Module die wesentlichen Eigenschaften und Methoden dann erben werden.

In `index.php` wird sie mit der Anweisung `require_once('system/Page.php')` inkludiert.

```
1  class Page {
2      /* properties */
3      private $description = 'This is my Music Directory App';
4      private $title = 'Website-Baukasten 0';
5      private $header = 'Tutorial: Musikkatalog';
6      private $headline = '???';
7      private $footer = '&copy;2016 &middot; Julius Caesar &middot; Roma &middot; Bella
Italia';
8
9      private $styles = array(
10         '<!-- page styles -->',
11         '<link rel="stylesheet" href="assets/css/styles.css"/>'
12     );
13
14     private $scripts = array(
15         '<!-- page scripts -->'
16     );
17
18     /* methods */
19
20     /* public setter */
21     public function setDescription($description){$this->description = $description;}
22     public function setTitle($title){$this->title = $title;}
23     public function setHeader($header){$this->header = $header;}
24     public function setHeadline($headline){$this->headline = $headline;}
25     public function setFooter($footer){$this->footer = $footer;}
26
27     /* public getter */
28     public function getDescription(){return $this->description;}
29     public function getTitle(){return $this->title;}
30     public function getHeader(){return $this->header;}
31     public function getHeadline(){return $this->headline;}
32     public function getFooter(){return $this->footer;}
33
34     /* public methods */
35     public function getStyles() {
36         $links = '';
37         foreach ($this->styles as $value) {
38             $links .= $value . "\n";
39         }
40         return $links;
41     }
42
43     public function getScripts() {
44         $scripts = '';
45         foreach ($this->scripts as $value) {
46             $scripts .= $value . "\n";
47         }
48         return $scripts;
49     }
```

```
50
51     public function getContent(){
52         parse_str($_SERVER['QUERY_STRING'], $param);
53         $module = (empty($param['module'])) ? '' : $param['module'];
54         $output = '<p>Modul <b>'.$module.'</b> gibt es (noch) nicht :-(</p>';
55         return $output;
56     }
57
58 } // end class Page
```

Listing 0-3: Page.php

Die Basisklasse stellt quasi die Standardklasse für beliebige Module dar. Sie definiert sozusagen die grundlegende Infrastruktur, indem sie Minimalversionen der benötigten Ausgaben für Inhaltsblöcke der Website bereitstellt, sofern ein Modul keine individuellen Inhalte liefert. Da wir in diesem Schritt 0 noch kein inhaltserzeugendes Modul haben, werden hier noch alle Ausgaben von `page.php` geliefert.

Während die `public`-getter-Methoden Standardelemente einer Website wie *Description*, *Title*, *Header* und *Footer* liefern (momentan nur Textkonstanten), erzeugen die Methoden `getStyles()` und `getScripts()` die benötigten Referenzen zu CSS- und Javascript-Dateien. So liefert `getStyles()` hier die Referenz zur CSS-Datei `styles.css`, wodurch unsere Seite schon mal ein ansehnliches Layout mit ein paar grafischen Elementen bekommt. Die Methoden der Klasse `Page` werden wir später in abgeleiteten Modulen individuell erweitern.

Ein Sondermodul, dass wir auch in diesem Schritt 0 schon als Minimalversion einführen wollen, ist ein Navigationsmodul zur Erzeugung eines Navigationsmenüs, auch wenn es vorerst außer der Menü-Überschrift noch nichts liefert.

```
1 class Navigation {
2
3     private $navigation = array();
4
5     public function getMenu(){
6         $menu = '<h1>Module</h1>';
7         $menu .= '<ul>';
8         foreach ($this->navigation as $key => $value) {
9             $menu .= '<li><a href="';
10            $menu .= $_SERVER['PHP_SELF'] . '?module=' . $value . '">';
11            $menu .= $key . '</a></li>';
12        }
13        $menu .= "</ul>\n";
14
15        return $menu;
16    }
17
18 } // end class Navigation
```

Listing 0-4: Navigation.php

Das noch leere Array `$navigation` in Zeile 3 wird später die Menüpunkte enthalten, aus denen dann die Methode `getMenu()` ein klickbares Menü erzeugt.

Damit haben wir alle Bausteinchen besprochen, die für die Erzeugung unserer noch inhaltslosen Website zusammenwirken. Das folgende Bild zeigt deren Ansicht im Browser. Man schaue sich zum tieferen Verständnis auch mal den *Quelltext im Browser* an, in dem alle PHP-Anweisungen in HTML umgewandelt sind.

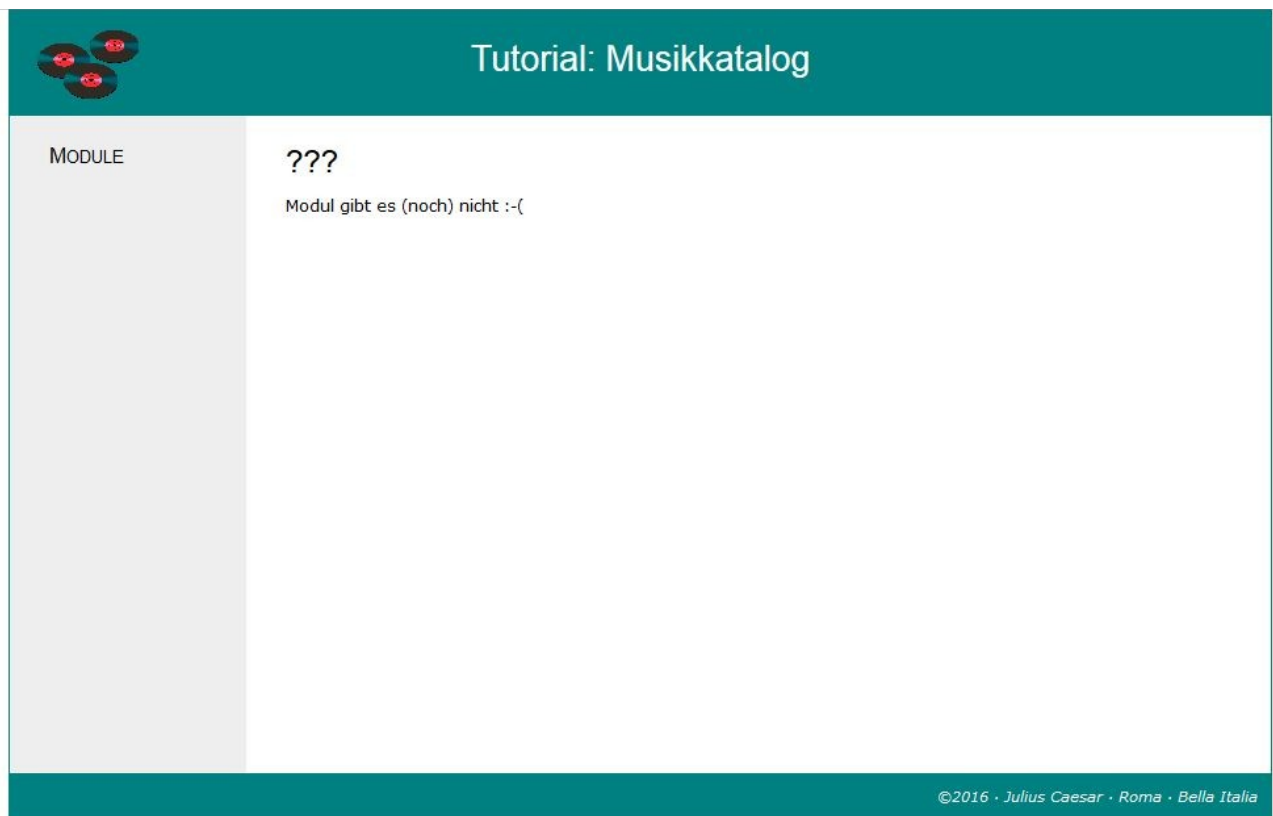


Abb. 0: Projekt Musikkatalog

Das Seitenlayout ist in der Datei `css/styles.css` beschrieben, die in `index.php` inkludiert wird und die die Styles für die Seitenbereiche, Schriften usw. definiert. Wie das im Detail funktioniert, lernt man im **Tutorial zu Layouts**.

Schritt 1

Schritt 1a

Die Website erhält ihr erstes Modul "Start" zur Ausgabe eines Begrüßungstextes.

Der Text steht vorerst noch inline im Modul Code.

Nachdem nun die Grundstruktur für unseren Musikkatalog geschaffen ist, können wir nun Module programmieren und in diesen Architekturrahmen einhängen und so die Website sukzessive ausbauen. Neue Module entstehen im Prinzip immer durch Ableitungen der Klasse **Page**. Wir nennen unser erstes einfaches Modul **Start** und legen es im Ordner *system/modules/Start* ab.

```
1  class Start extends Page {
2      /* properties */
3      private $headline = 'Hello World!';
4
5      /* getter */
6      public function getHeadline(){return $this->headline;}
7
8      /* public methods */
9      public function getContent(){ // Modulausgabe erzeugen
10         $content = '<div id="welcome"><p>';
11         $content .= 'Das ist die Begrüßungsseite zu unserem <i>modular</i>'
zu entwickelnden ';
12         $content .= 'Beispielprojekt. Wir werden sukzessive Module für
weitere Seiten hinzufügen ';
13         $content .= 'und so schrittweise eine kleine Anwendung zur
Verwaltung von Musikalben ';
14         $content .= 'aufbauen. Dabei kommen die Web-Technologien HTML, CSS,
PHP, MySQL ';
15         $content .= 'und schließlich auch noch Javascript zum Einsatz.</p>';
16         $content .= '<p>Unser Schwerpunkt liegt allerdings mehr auf einer
systematischen ';
17         $content .= '<i>modularen Entwicklungsmethodik</i> und darauf, wie
man diese ';
18         $content .= 'verschiedenen Technologien zusammenstößelt, ohne die
Übersicht ';
19         $content .= 'zu verlieren, als auf ihren vielen programmtechnischen
Details.';
20         $content .= '</p></div>';
21
22         $content .= '<p><b>Tutorial Web-Baukasten - Schritt 1a</b></p>';
23         $content .= '<p>Die Website erhält ihr erstes Modul "Start" zur
Ausgabe eines Begrüßungstextes.<br>';
24         $content .= '<i>Der Text steht inline im Modul Code.</i></p>';
25
26         return $content;
27     } // end getContent()
28
29 } // end class Start
```

Listing 1a-1: Start.php

Schritt 1b

Die Website erhält ihr erstes Modul "Start" zur Ausgabe eines Begrüßungstextes.

Der Text wird nun in eine externe Datei namens *welcome.html* ausgelagert, die im Modul Start in die Variable `$content` eingelesen wird. Das ist flexibler, da der Text geändert werden kann, ohne in den Modulcode eingreifen zu müssen.

```
1 class Start extends Page {
2     /* properties */
3     private $headline = 'Hello World!';
4
5     /* getter */
6     public function getHeadline(){return $this->headline;}
7
8     /* public methods */
9     public function getContent(){ // Modulausgabe erzeugen
10
11         $content = '<div id="welcome">';
12         // Welcome-Datei als String einlesen
13         $content .=
14 file_get_contents('system/modules/Start/assets/welcome.html');
15         $content .= '</div>';
16
17         return $content;
18     }
19 } // end class Start
```

Listing 1b-1: Start.php

```
1 <p>Das ist die Begrüßungsseite zu unserem <i>modular</i> zu entwickelnden
Beispielprojekt. Wir werden sukzessive Module für weitere Seiten hinzufügen und
so schrittweise eine kleine Anwendung zur Verwaltung von Musikalben aufbauen.
Dabei kommen die Web-Technologien HTML, CSS, PHP, MySQL und schließlich auch
noch Javascript zum Einsatz.</p>
2 <p>Unser Schwerpunkt liegt allerdings mehr auf einer systematischen
<i>modularen</i> Entwicklungsmethodik</i> und darauf, wie man diese verschiedenen
Technologien zusammenstößelt, ohne die Übersicht zu verlieren, als auf ihren
vielen programmtechnischen Details.</p>
3 <p><b>Tutorial Web-Baukasten - Schritt 1b</b></p>
4 <p>Die Website erhält ihr erstes Modul "Start" zur Ausgabe eines
Begrüßungstextes.<br>
5 <i>Der Text steht in einer externen Datei.</i></p>
```

Listing 1b-2: welcome.html

Um dieses neue Modul über das Navigationsmenü auswählen zu können, müssen wir das Array im Modul Navigation nun entsprechend befüllen. Der (Platzhalter) soll einfach erinnern, dass die Liste hier für weitere Module ergänzt werden muss. Das werden wir aber künftig nicht mehr bei jedem neuen Modul erwähnen.

```
1 class Navigation {
2
3     private $navigation = array(
4         'Startseite' => 'Start',
5         '(Platzhalter)' => 'Dummy'
6     );
7
8     public function getMenu(){
9
10
11
12
13
14
15
16
17
18
19     }
20
21 } // end class Navigation
```

Listing 1a/1b-3: Navigation.php

Damit neue Module im Navigationsmodul auch aufgerufen werden können, muss jeweils ein `require_once`, eine Objektinstanziierung und ein case im switch in *index.php* ergänzt werden.

```
1 /* error reporting settings */
2 \error_reporting(\E_ALL ^ \E_NOTICE); // suppress notices
3
4 /* import page and module classes */
5 require_once('system/Page.php');
6 $page = new Page();
7 require_once('system/modules/Navigation/Navigation.php');
8 $nav = new Navigation();
9 require_once('system/modules/Start/Start.php');
10 $start = new Start();
11
12 /* switch to selected page module */
13 parse_str($_SERVER['QUERY_STRING'], $param);
14 $module = (empty($param['module'])) ? 'Start' : $param['module'];
15
16 switch ($module){
17     case 'Start':
18         $mod = $start;
19         break;
20     default:
21         $mod = $page;
22 }
23
24 /* generate page */
25 require_once('system/templates/page_default.php');
```

Listing 1a/1b-4: index.php

Die wesentlichen Schritte zur Erweiterung der Website um ein neues Modul sind also:

- Erweiterung der Klasse Navigation.php
- Erweiterung von index.php
- Programmierung der Klasse des neuen Moduls sowie evtl. weiterer Code-Teile

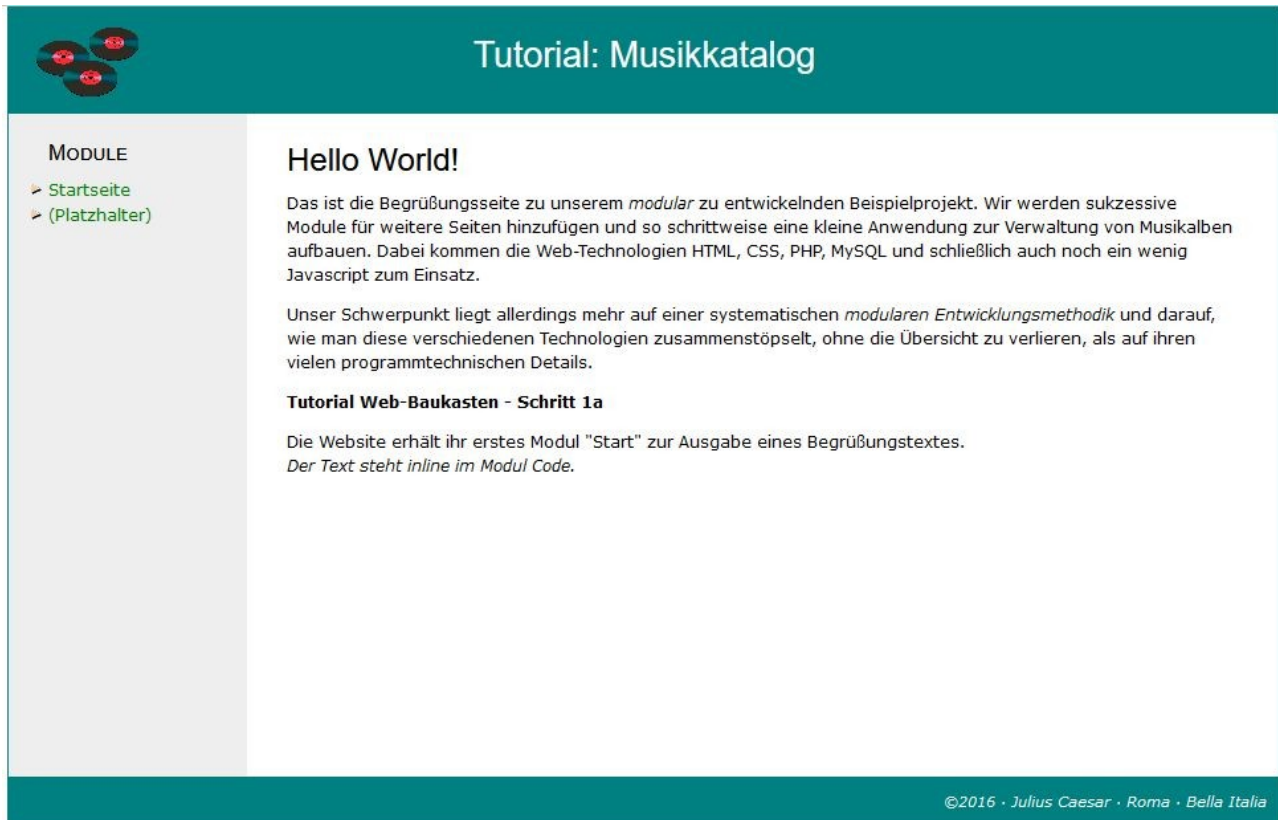


Abb. 1a/b: Projekt Musikkatalog – Schritt 1a bzw. 1b

Die Ausgabe im Browser für Schritt 1a bzw. 1b ist bis auf eine geringe Textanpassung gleich.

Schritt 2

Schritt 2a

Die Website erhält ihr zweites Modul "Catalog" zur Anzeige von Musikalben aus einer CSV-Datei.

Diese erste Version zeigt die Albenliste als einfache Textzeilen aus einer Datei *alben.csv* an. Die Klasse *Catalog.php* wird wieder von der Klasse *Page.php* abgeleitet und die Methode *getContent()* wird ausprogrammiert, d.h. die Albendatei wird eingelesen und direkt in der Methode für die Ausgabe aufbereitet.

```
1  class Catalog extends Page {
2      /* properties */
3      private $headline = 'Meine Alben';
4      private $listlength = '20'; // length of displayed album list
5
6      private $styles = array(
7          '<!-- module styles -->'
8      );
9
10     private $scripts = array(
11         '<!-- module scripts -->'
12     );
13
14     /* getter */
15     public function getHeadline(){return $this->headline;}
16
17     /* methods */
18     public function getStyles() {
19         $links = parent::getStyles();
20         foreach ($this->styles as $value) {
21             $links .= $value . "\n";
22         }
23         return $links;
24     }
25
26     public function getScripts() {
27         $scripts = parent::getScripts();
28         foreach ($this->scripts as $value) {
29             $scripts .= $value . "\n";
30         }
31         return $scripts;
32     }
33
34     public function getContent(){ // Modulausgabe erzeugen
35         $content = '<div id="catalog">';
36
37         // CSV-Datei einlesen und aufbereiten
38         $handle = fopen ('system/modules/Catalog/assets/alben.csv','r');
39         $header = fgetcsv ($handle, 1000, ";"); // 1. Zeile = Spaltennamen
40         while ( ($row = fgetcsv ($handle, 1000, ";")) ) { // Zeilen in Array
41             $content .= implode(" - ", $row) . '<br>'; // Minuszeichen als
42             // Trenner
43         }
44     }
45 }
```

```

43         fclose ($handle); // CSV-Datei schliessen
44
45         $content .= '</div>';
46
47         return $content;
48     }
49
50 } // end class Catalog

```

Listing 2a-1: Catalog.php

Die Albendatei befindet sich im Unterordner `assets` des Moduls `Catalog`.

Tutorial: Musikkatalog

MODULE

- Startseite
- Musikkatalog**

Meine Alben

Louis Jordan - Rock'n'Roll - Mercury - 8382192 - 1989 - Rock'n'Roll der 40er/50er Jahre
 Cat Stevens - Wild World - Pulsar - Puls 028 - 1992 - Querschnitt aus verschiedenen Alben
 Steve Winwood - Chronicles - Island - 258595 - 1987 - Solo-Album mit neueren Songs
 Them - Them Featuring Van Morrison - DERAM - 8209352 - 1990 - 45 Songs auf 2 CDs
 Bob Dylan - Greatest Hits - CBS - 4609079 - 1967 - Das Wichtigste von Dylan
 Beatles - The Beatles / 1962-1966 - Apple - CDS 7970362 - 1993 - The Red Album (Doppel-CD)
 Beatles - The Beatles / 1967-1970 - Apple - CDS 7970392 - 1993 - The Blue Album (Doppel-CD)
 Rolling Stones - Singles Collection - ABKCO - 8444812 - 1986 - The London Years auf 3 CDs
 Harry Belafonte - Greatest Hits - Worlds Star Collection - WSC 99011 - 1987 - Sammlung bekannter Hits
 Peter Tosh - Mama Africa - EMI - CDP 7916712 - 1983 - Reggae vom Feinsten
 Stevie Wonder - Song Review - MOTOWN - 5307572 - 1996 - A Greatest Hits Collection
 Simon & Garfunkel - Silent Voices - Back Biter - BB 61047 - 1985 - Die bekanntesten Hits
 Benny Goodman - Swing, Swing, Swing - ORBIS London - JAZ CD 004 - 1994 - Classic Jazz Collection
 Spider Murphy Gang - Single Hit Collection 1980-1993 - EMI - 7813542 - 1993 - Bayrischer Rock'n'Roll, zeitlos gut
 Trio Bavario - Bavario - Trikont - EFA 00159 - 1989 - Bayrisch-Lateinamerikanische Rhythmen
 Raffi - Bananaphone - Shoreline - CD 8062 - 1995 - Kinderlieder aus Kanada

©2016 · Julius Caesar · Roma · Bella Italia

Abb. 2a: Modul `Catalog` mit Ausgabe `list.php`

Schritt 2b

Die Website erhält ihr zweites Modul "Catalog" zur Anzeige von Musikalben aus einer CSV-Datei.

- Diese zweite Version zeigt die Albenliste über Templates an.
- Es gibt zwei Templates zur wahlweisen Anzeige als Textzeilen oder als Tabelle.
- Für die Tabellenanzeige wird das Stylesheet erweitert.

Wir orientieren uns mit unserem Projekt am Entwurfsmuster MVC (Model-View-Controller), d.h. wir möchten die drei Aspekte *Datenmodell* (model), *Ansicht* (view) und *Programmsteuerung* (controller) möglichst getrennt halten. Bisher war alles so einfach, dass diese Trennung nicht zum Tragen kam. Nun wird sie aber folgendermaßen erkennbar:

M(odel): die Datei `alben.csv`

V(iew): zwei Templates `list.php` oder `table.php` zur wahlweisen Ausgabe als Liste oder Tabelle

C(ontroller): die Klasse `Catalog.php`

```
1  class Catalog extends Page {
2      /* properties */
3      private $headline = 'Meine Alben';
4      private $listlength = '20'; // length of displayed album list
5
6      private $styles = array(
7          '<!-- module styles -->'
8      );
9
10     private $scripts = array(
11         '<!-- module scripts -->'
12     );
13
14     /* getter */
15     public function getHeadline(){return $this->headline;}
16
17     /* methods */
18     public function getStyles() {
19         $links = parent::getStyles();
20         foreach ($this->styles as $value) {
21             $links .= $value . "\n";
22         }
23         return $links;
24     }
25
26     public function getScripts() {
27         $scripts = parent::getScripts();
28         foreach ($this->scripts as $value) {
29             $scripts .= $value . "\n";
30         }
31         return $scripts;
32     }
33 }
```

```

34     public function getContent(){ // Modulausgabe erzeugen
35         $content = '<div id="catalog">';
36         $rows = array();
37
38         // CSV-Datei einlesen und aufbereiten
39         $handle = fopen ('system/modules/Catalog/assets/alben.csv', "r");
40         $header = fgetcsv ($handle, 1000, ";"); // 1. Zeile = Spaltennamen
41         while ( ($data = fgetcsv ($handle, 1000, ";")) ) { // Zeilen in Array
42             $i = 0;
43             foreach ($data as $value) { // key-value Array pro Zeile
44                 $row[$header[$i++]] = utf8_encode($value); // UTF8 wg.
45             } // foreach $data
46             $rows[] = $row; // array of arrays
47         }
48         fclose ($handle); // CSV-Datei schliessen
49
50         // use a template to format the view
51         include_once('system/modules/Catalog/templates/table.php');
52
53         $content .= '</div>';
54
55         return $content;
56     }
57
58 } // end class Catalog

```

Listing 2b-1: Catalog.php

Die Albendaten werden dieses Mal nicht direkt ausgegeben, sondern in ein Array geladen und dann an ein Ausgabe-Template (View) übergeben. Im Code-Beispiel ist das `table.php` für eine Tabellenausgabe, übergibt man an `list.php`, so wird es eine Listenausgabe (wie in 2a).

```

1  $content .=
2  '<div id="albenliste">
3      <table>
4      <tr>
5          <th>Interpret</th>
6          <th>Titel</th>
7          <th>Label</th>
8          <th>Master#</th>
9          <th>Jahr</th>
10         <th>Bemerkung</th>
11     </tr>';
12
13     $nr = 0;
14     foreach ($rows as $row) {
15         $class = ($nr%2 == 0)?'':' class="even"'; // style für gerade Zeilen
16         $content .= '<tr'. $class . '>';
17         foreach ($row as $key => $value) {
18             $content .= '<td>' . $value . '</td>';
19         }
20         $nr++;


```

```

21     $content .= '</tr>';
22 }
23 $content .= '</table></div>';

```

Listing 2b-2: table.php



Tutorial: Musikkatalog

MODULE

- Startseite
- Musikkatalog

Meine Alben

Interpret	Titel	Label	Master#	Jahr	Bemerkung
Louis Jordan	Rock'n'Roll	Mercury	8382192	1989	Rock'n'Roll der 40er/50er Jahre
Cat Stevens	Wild World	Pulsar	Puls 028	1992	Querschnitt aus verschiedenen Alben
Steve Winwood	Chronicles	Island	258595	1987	Solo-Album mit neueren Songs
Them	Them Featuring Van Morrison	DERAM	8209352	1990	45 Songs auf 2 CDs
Bob Dylan	Greatest Hits	CBS	4609079	1967	Das Wichtigste von Dylan
Beatles	The Beatles / 1962-1966	Apple	CDS 7970362	1993	The Red Album (Doppel-CD)
Beatles	The Beatles / 1967-1970	Apple	CDS 7970392	1993	The Blue Album (Doppel-CD)
Rolling Stones	Singles Collection	ABKCO	8444812	1986	The London Years auf 3 CDs
Harry Belafonte	Greatest Hits	World Star Collection	WSC 99011	1987	Sammlung bekannter Hits
Peter Tosh	Mama Africa	EMI	CDP 7916712	1983	Reggae vom Feinsten
Stevie Wonder	Song Review	MOTOWN	5307572	1996	A Greatest Hits Collection
Simon & Garfunkel	Silent Voices	Back Biter	BB 61047	1985	Die bekanntesten Hits
Benny Goodman	Swing, Swing, Swing	ORBIS London	JAZ CD 004	1994	Classic Jazz Collection
Spider Murphy Gang	Single Hit Collection 1980-1993	EMI	7813542	1993	Bayrischer Rock'n'Roll, zeitlos gut
Trio Bavario	Bavario	Trikont	EFA 00159	1989	Bayrisch-Lateinamerikanische Rhythmen
Raffi	Bananaphone	Shoreline	CD 8062	1995	Kinderlieder aus Kanada

©2016 • Julius Caesar • Roma • Bella Italia

Abb. 2b: Modul Catalog mit Ausgabe table.php

Schritt 3

Die Website erhält ihr drittes Modul "Gallery" zur Anzeige von Cover Images aus einem Bilderverzeichnis. Die Sortierungen von Albenliste und Cover-Galerie sind *nicht* synchronisiert.

Die Cover Images sind im Unterordner assets/covers abgelegt und werden zur Anzeige von dort geladen. Zur Anzeige wird das Template fromdir.php als View verwendet. Wie im vorigen Schritt erläutert, kann die Anzeige leicht durch andere Views verändert werden.

```
1 class Gallery extends Page {
2     /* properties */
3     private $headline = 'Cover Galerie';
4     private $listlength = '12'; // length of displayed cover list
5
6     /* getter */
7     public function getHeadline(){return $this->headline;}
8
9     /* public methods */
10    public function getContent(){
11        $content = '<div id="gallery">';
12
13        // use a template to format the view
14        include_once('system/modules/Gallery/templates/fromdir.php');
15        $content .= '</div>';
16
17        return $content;
18    }
19
20 } // end class Gallery
```

Listing 3-1: Gallery.php

```
1 $path = 'system/modules/Gallery/assets/covers/';
2 $files = scandir($path);
3 foreach ($files as $file) {
4     if (substr($file,0,1) != '.') { // filename must not start with dot
5         $name = explode('.', $file);
6         $content .= '';
7     }
8 }
```

Listing 3-2: fromdir.php

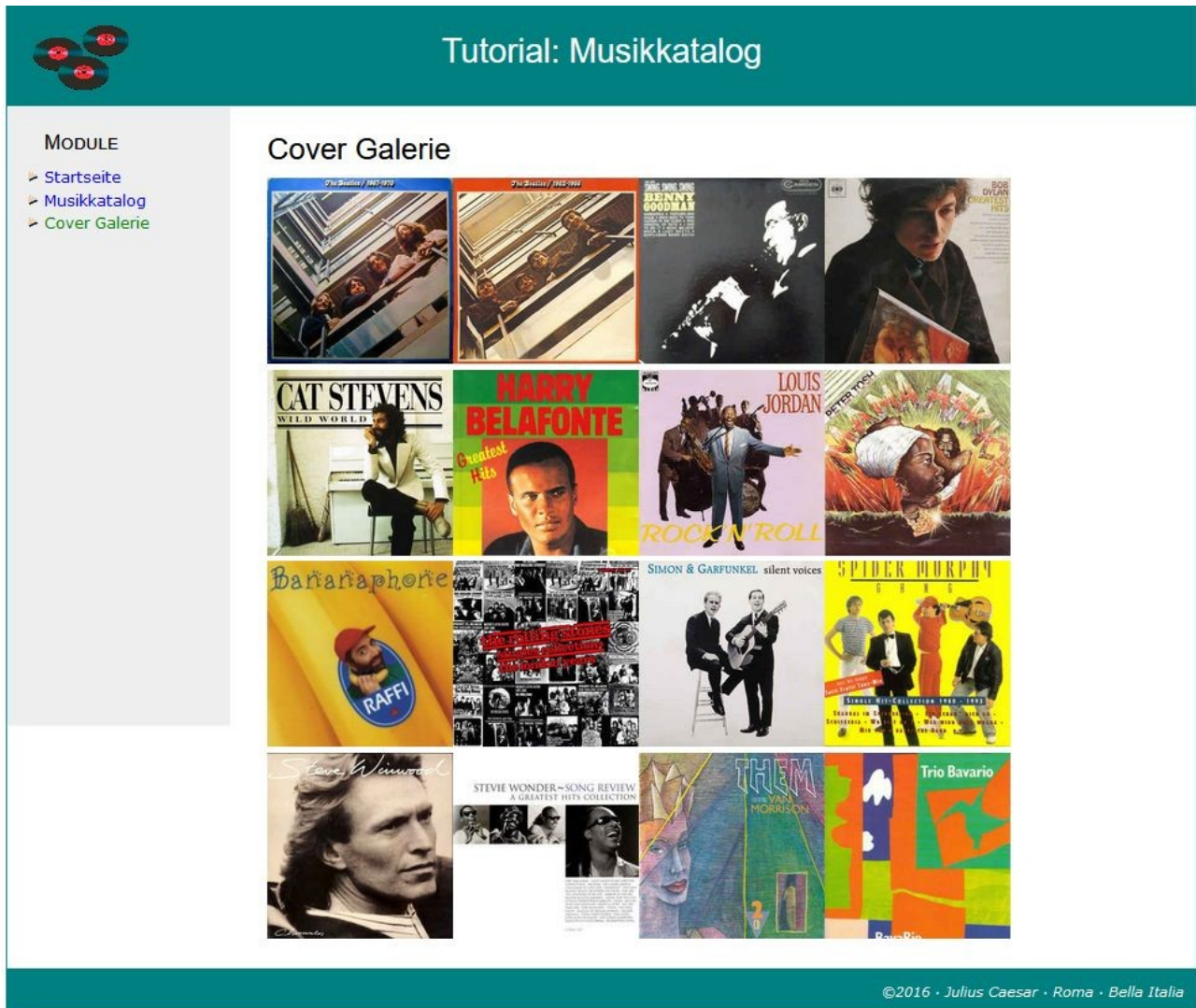


Abb. 3: Modul Gallery mit Ausgabe fromdir.php

Schritt 4

Die Datenhaltung und -beschaffung der Module der Website wird vom Dateisystem auf eine MySQL-Datenbank umgestellt. Die Änderungen bleiben nach außen hin unsichtbar, für den Besucher der Website gibt es keine sichtbaren Veränderungen. Jedenfalls fast: Die Sortierung in Albenliste und Cover-Galerie ist nun synchron, da die Cover-Referenzen nun mit den Albendaten aus der Datenbank gelesen werden.

Für diesen Schritt sind einige Vorbereitungen notwendig:

- MySQL muss installiert und gestartet sein
- eine Datenbank mit einer Tabelle musik muss angelegt sein
- die Musiktabelle muss mit Daten gefüllt sein
- eine Klasse Album.php wird als Zugriffsschicht zur Datenbank musik benötigt
- die Klasse page.php wird um Methoden zur Datenbankverbindung ergänzt.

Im Unterordner db gibt es Skripte zum Anlegen und Befüllen der Datenbanktabelle musik.

```
1 DROP TABLE IF EXISTS album;
2 CREATE TABLE album (
3     idAlbum INTEGER(10) UNSIGNED NOT NULL auto_increment,
4     interpret VARCHAR(128) NOT NULL,
5     titel VARCHAR(128) NOT NULL,
6     label VARCHAR(30) NULL,
7     masternr VARCHAR(20) NULL,
8     jahr CHAR(4) NULL,
9     bemerkung TEXT NULL,
10    coverfile VARCHAR(255) NULL,
11    mp3file VARCHAR(255) NULL,
12    PRIMARY KEY(idAlbum)
13 ) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Listing 4-1: musik.sql

Das Feld mp3file in Zeile 11 werden wir vorerst nicht nutzen, es wird einfach schon mal für künftigen Ausbau mitangelegt.

In der Klasse `Page.php` im Unterordner `system` werden die Methoden zur Datenbankverbindung ergänzt.

```
1 class Page {
  .
  .
17
18     /* DB connection properties (adjust to your environment!) */
19     protected $dbhost = 'localhost:3306';
20     protected $dbuser = 'root';
21     protected $dbpass = '';
22     protected $dbname = 'musik';
23     protected $mysqli = null;
24
25     /* protected methods */
26     protected function connectDB() {
27
28         if ($this->mysqli != null) { return $this->mysqli; }
29
30         $this->mysqli = new mysqli($this->dbhost, $this->dbuser, $this->
>dbpass, $this->dbname);
31         if ($this->mysqli->connect_errno) {
32             die('<p class="error">MySQL Error ' . $this->mysqli->
>connect_errno . ': ' . $this->mysqli->connect_error . '</p>');
33         }
34
35         return $this->mysqli;
36     }
37
38     protected function disconnectDB() {
39         $this->mysqli->close();
40     }
  .
  .
78 } // end class Page
```

Listing 4-2: Page.php

Mit der Klasse `Album.php` im Unterordner `db/dao` wird ein Datenzugriffsobjekt (DAO, *Data Access Object*) realisiert. Die Klasse enthält alle Methoden zur Manipulation der Musiktable. Da sie etwas länglich ist, wird sie hier nicht extra aufgelistet, sondern zum *Code Reading* empfohlen.

In der Klasse `Gallery.php` ist die Methode `getContent()` nun für den Datenbankzugriff angepasst worden.

```
1  class Gallery extends Page {
2      /* properties */
3      private $headline = 'Cover Galerie';
4      private $listlength = '20'; // length of displayed cover list
5
6      /* getter */
7      public function getHeadline(){return $this->headline;}
8
9      /* public methods */
10     public function getContent(){
11         $db = $this->connectDB();
12         $album = new Album($db);
13         $fields = 'interpret, titel, coverfile';
14         $rows = $album->load_fields($fields, $this->listlength);
15         $content = '<div id="gallery">';
16         // use a template to format the view
17         include_once('system/modules/Gallery/templates/fromdb.php');
18         $content .= '</div>';
19         $this->disconnectDB();
20
21         return $content;
22     }
23
24 } // end class Gallery
```

Listing 4-3: Gallery.php

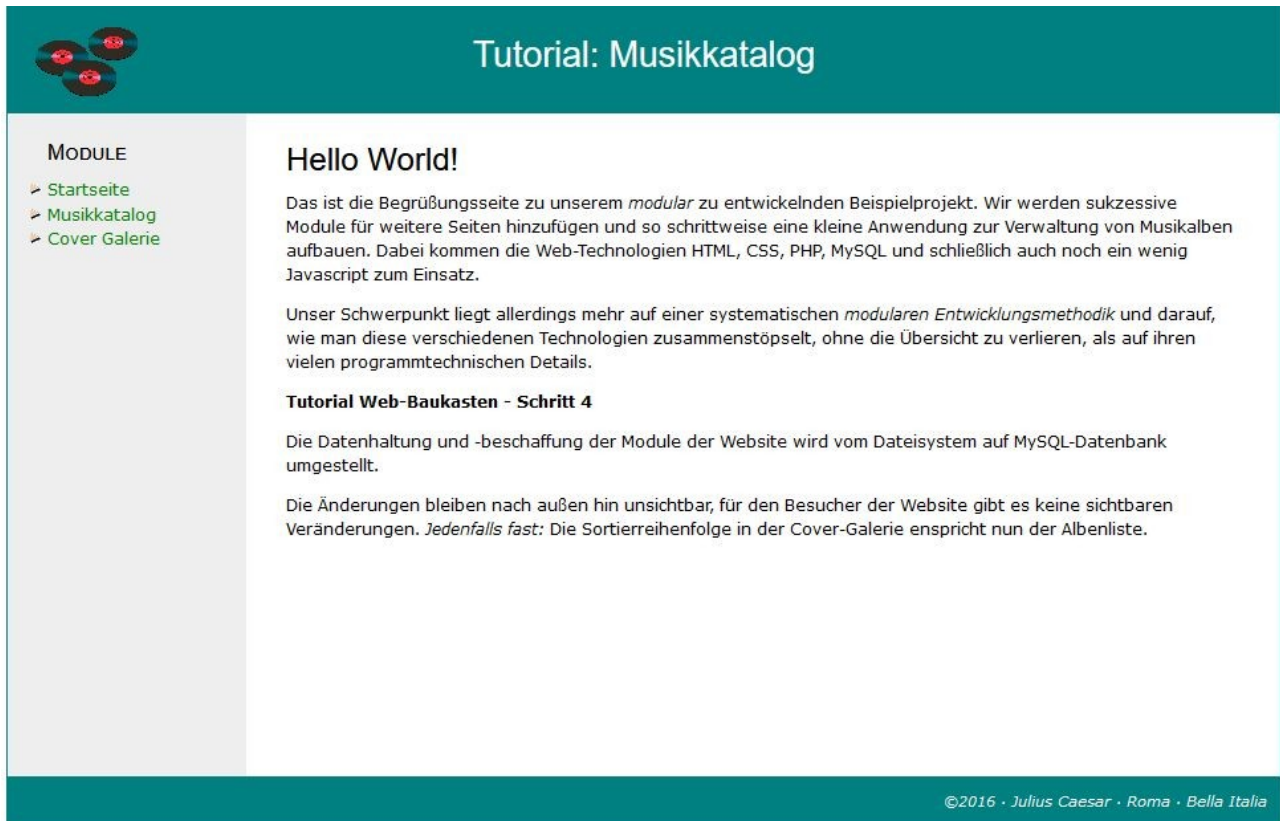


Abb. 4: Gallery.php

Schritt 5

Die Website erhält ihr viertes Modul "Insert" zur Erfassung neuer Musikalben und deren Eintragung in die Datenbank.

Wenn dieses Tutorial über den reinen Lerneffekt hinaus nützlich sein soll, müssen natürlich eigene Albendaten gepflegt werden können. Die mitgelieferten Daten dienen lediglich als Beispiele. In diesem Kapitel geht es daher um die Datenpflege des Musikkatalogs.

Als erstes soll ein neues Album eingefügt werden können und wir implementieren eine Funktion „Album einfügen“ als Modul „Insert“ mittels einer Klasse `Insert.php`.

```
1  class Insert extends Page {
2      /* properties */
3      private $headline = 'Neues Album erfassen';
4      private $frm; // form variables
5      private $errors = array(); // error messages
6
7      /* getter */
8      public function getHeadline(){return $this->headline;}
9
10     /* methods */
11     public function getContent(){
12         if (!$this->isValidFormInput()) {
13             $content = $this->getForm();
14         } else {
15             $this->addAlbumToDb();
16             $content = '<p>Die Daten wurden gespeichert.</p>';
17             // clear form to prevent multiple processing
18             unset($_POST);
19         } // end if isValidFormInput()
20
21         return $content;
22     }
23
24     /* private methods */
25     private function getForm(){
26         include_once('system/modules/Insert/templates/form_insert.php');
27         return;
28     }
29
30     private function isValidFormInput() {
31         /* validate the form, and return true, if no errors. */
32         $this->frm = $_POST;
33         if (empty($this->frm)) {return false;}
34         unset($this->errors);
35         if (empty($this->frm['interpret'])) {
```

```
36         $this->errors['interpret'] = 'Interpret fehlt';
37     }
38     if (empty($this->frm['titel'])) {
39         $this->errors['titel'] = 'Titel fehlt';
40     }
41     if (!empty($this->frm['jahr']) &&
42         ($this->frm['jahr'] < 1900 || $this->frm['jahr'] >
43         date('Y')))) {
44         $this->errors['jahr'] = 'Jahr ungültig';
45     }
46     if (!empty($this->frm['coverfile'])) {
47         $part = explode('.', $this->frm['coverfile']);
48         $pattern = "[a-z0-9_-]+/i";
49         // nur Buchstaben, Ziffern, _ oder -
50         preg_match($pattern, $part[0], $matches);
51         if (empty($part[0]) or $part[0] != $matches[0] or
52             ($part[1] != 'jpg' and $part[1] != 'png'
53              and $part[1] != 'gif'))
54         {
55             $this->errors['coverfile'] = 'Dateiname ungültig: '
56                                     . $matches[0];
57         }
58     }
59     return (count($this->errors) == 0) ? true : false;
60 } // end function validateForm
61
62 public function addAlbumToDb() {
63     // insert new album into DB
64     $db = $this->connectDB();
65     $album = new Album($db);
66     $album->setinterpret(addslashes(trim($this->frm['interpret'])));
67     $album->settitel(addslashes(trim($this->frm['titel'])));
68     $album->setlabel(addslashes(trim($this->frm['label'])));
69     $album->setmasternr(addslashes(trim($this->frm['masternr'])));
70     $album->setjahr(addslashes(trim($this->frm['jahr'])));
71     $album->setbemerkung(addslashes(trim($this->frm['bemerkung'])));
72     $album->setcoverfile(addslashes(trim($this->frm['coverfile'])));
73
74     $err = $album->insert();
75     $this->disconnectDB();
76
77     return;
78 } // end function addAlbumToDb
79
80 } // end class Insert
```


Listing 5-1: Insert.php

Das Modul prüft als erstes, ob es gültige Eingabedaten aus einem Formular `form_insert.php` bekommen hat. Falls nicht, lädt es das Eingabeformular und zeigt es an. Der Benutzer füllt das Formular aus und schickt es ab. Dadurch wird das Modul nochmals zur Prüfung der Eingaben aufgerufen. Sind alle Daten korrekt eingegeben, werden das neue Album in die Datenbank eingefügt und eine Erfolgsmeldung ausgegeben. Gab es Fehler bei der Prüfung, so werden diese markiert und das Formular mit den Eingabedaten und Fehlermeldungen zur Korrektur wieder ausgegeben. Das läuft solange in einer Schleife bis alle Eingaben als korrekt akzeptiert werden können.

```
1  <form name="entryform" method="post"
2      action="<?php echo $_SERVER['PHP_SELF'] . '?module=Insert'; ?>">
3  <fieldset><legend><b>Dateneingabe</b></legend>
4  <table>
5      <tr><td colspan="2"><p><i>mit * markierte Felder müssen ausgefüllt
werden!</i></p></td>
6  </tr>
7  <tr>
8      <td class=label>Interpret* </td>
9      <td><input type="text" name="interpret"
10          value="<?php echo $this->frm['interpret']; ?>">&nbsp;<?php echo '<span class="error">' . $this->errors['interpret'] .
11          '</span>'; ?></td>
12  </tr>
13  .
14  .
15  .
46  <tr>
47      <td><p>&nbsp;</p></td>
48      <td><input type="submit" value="speichern"></td>
49  </tr>
50  </table>
51  </fieldset>
52  </form>
```

Listing 5-2: Ausschnitt des Formulars `form_insert.php`

Nach Klicken des Submit-Buttons (Zeile 48) wird die Action des Formulars (Zeile 2) aufgerufen, die darin besteht, dass das Modul „Insert“ sich selbst wieder aufruft.



Tutorial: Musikkatalog

MODULE

- Startseite
- Musikkatalog
- Cover Galerie
- Album hinzufügen

Neues Album erfassen

Dateneingabe

*mit * markierte Felder müssen ausgefüllt werden!*

Interpret*

Titel*

Label

MasterNr.

Jahr

Bemerkung

Cover-Datei (.jpg, .png oder .gif)

©2016 · Julius Caesar · Roma · Bella Italia

Abb. 5-1: Formular des Moduls Insert

Schritt 6

Schritt 6a


Das Modul "Insert" wird in die Kataloganzeige integriert und die Funktionen "Update" und "Delete" werden vorbereitet.

Das Hinzufügen eines neuen Albums allein reicht nicht, um den Musikkatalog umfänglich zu pflegen, man muss auch Datensätze ändern und löschen können. Das ist auch nicht gut über das Navigationsmenü zu machen, da man ja angeben können muss, welche Alben denn geändert oder gelöscht werden sollen. Das geht am besten direkt aus der Kataloganzeige heraus, wo man die vorhandenen Alben für die Funktionen *Update* und *Delete* auswählen kann. Auch die Funktion *Insert* ist in der Kataloganzeige besser aufgehoben als im Navigationsmenü.

Diese Änderungen sind einfach über die View `table.php` der Kataloganzeige zu machen, indem in der Tabelle links vor den Albenzeilen Icons für „ändern“ und „löschen“ und unter der Tabelle ein Button „Album einfügen“ eingefügt werden. Die Icons bleiben vorerst noch funktionslos.

```
1 // create table
2 $content .=
3 '<div id="albenliste">
4     <table>
5         <tr>
6             <th>&nbsp;</th>
7             <th>&nbsp;</th>
8             <th>Interpret</th>
9             <th>Titel</th>
10            <th>Label</th>
11            <th>Master#</th>
12            <th>Jahr</th>
13            <th>Bemerkung</th>
14        </tr>';
15
16 $nr = 0;
17 foreach ($rows as $row) {
18     $class = ($nr%2 == 0)?'':' class="even"'; // style für gerade Zeilen
19     $content .= '<tr'. $class . '>';
20     $content .= '<td><a href="#"></a></td>';
21     $content .= '<td><a href="#"></a></td>';
22     foreach ($row as $key => $value) {
23         $content .= '<td>' . $value . '</td>';
24     }
25     $nr++;
26     $content .= '</tr>';
27 }
28 $content .= '</table></div>';
29 // add button
30 $content .= '<div class="button"><a href="' . $_SERVER['PHP_SELF'] . '?
module=Insert">Album hinzufügen</a></div>';
```

Listing 6a-1: `table.php`



































Tutorial: Musikkatalog

MODULE

- Startseite
- Musikkatalog
- Cover Galerie

Meine Alben

	Interpret	Titel	Label	Master#	Jahr	Bemerkung
 	Louis Jordan	Rock'n'Roll	Mercury	8382192	1989	Rock'n'Roll der 40er/50er Jahre
 	Cat Stevens	Wild World	Pulsar	Puls 028	1992	Querschnitt aus verschiedenen Alben
 	Steve Winwood	Chronicles	Island	258595	1987	Solo-Album mit neueren Songs
 	Them	Them Featuring Van Morrison	DERAM	8209352	1990	45 Songs auf 2 CDs
 	Bob Dylan	Greatest Hits	CBS	4609079	1967	Das Wichtigste von Dylan
 	Beatles	The Beatles / 1962-1966	Apple	CDS 7970362	1993	The Red Album (Doppel-CD)
 	Beatles	The Beatles / 1967-1970	Apple	CDS 7970392	1993	The Blue Album (Doppel-CD)
 	Rolling Stones	Singles Collection	ABKCO	8444812	1986	The London Years auf 3 CDs
 	Harry Belafonte	Greatest Hits	Worlds Star Collection	WSC 99011	1987	Sammlung bekannter Hits
 	Peter Tosh	Mama Africa	EMI	CDP 7916712	1983	Reggae vom Feinsten
 	Stevie Wonder	Song Review	MOTOWN	5307572	1996	A Greatest Hits Collection
 	Simon & Garfunkel	Silent Voices	Back Biter	BB 61047	1985	Die bekanntesten Hits
 	Benny Goodman	Swing, Swing, Swing	ORBIS London	JAZ CD 004	1994	Classic Jazz Collection
 	Spider Murphy Gang	Single Hit Collection 1980-1993	EMI	7813542	1993	Bayrischer Rock'n'Roll, zeitlos gut
 	Trio Bavario	Bavario	Trikont	EFA 00159	1989	Bayrisch-Lateinamerikanische Rhythmen
 	Raffi	Bananaphone	Shoreline	CD 8062	1995	Kinderlieder aus Kanada

Album hinzufügen

©2016 • Julius Caesar • Roma • Bella Italia

Abb. 6a-1: Action Icons in der Kataloganzeige

Schritt 6b

Das Modul "Insert" wird zu "InsUpd" umgebaut, um die Funktionen "Insert" und "Update" zusammenzufassen.

Da das Formular für Insert oder Update identisch ist, außer dass im ersten Fall initial ein leeres Formular, im zweiten aber ein vorausgefülltes angezeigt werden muss und die Logik der Verarbeitung sehr ähnlich ist, wird das Modul „InsUpd“ als Fallunterscheidung realisiert. Um das zu verstehen, studiert man am besten den Beispielcode. Hier nur grob das Konzept.

```
1  <?php define("HEADINSERT", "Neues Album erfassen");
2  define("HEADUPDATE", "Albumdaten bearbeiten");
3
4  class InsUpd extends Page {
5      /* properties */
6      private $headline;
7      private $initUpdate = false;
8      private $frm = array(); // form variables
9      private $errors = array(); // error messages
10
11     public function __construct(){
12         $this->frm = $_POST;
13         parse_str($_SERVER['QUERY_STRING'], $param);
14         if ($param['act'] == 'upd'){ // update oder insert?
15             $this->frm['idAlbum'] = $param['id'];
16             $this->initUpdate = true;
17             $this->loadForm();
18         }
19         if(is_numeric($this->frm['idAlbum'])){
20             $this->headline = HEADUPDATE;
21         } else {
22             $this->headline = HEADINSERT;
23         }
24     }
25
26     .
27     .
28     .
29
30     public function writeAlbumToDb() {
31         // insert new album into DB
32         $db = $this->connectDB();
33         if (!$this->frm['idAlbum']) {$this->frm['idAlbum'] = 0;}
34         $album = new Album($db, $this->frm['idAlbum']);
35         $album->setinterpret(addslashes(trim($this->frm['interpret'])));
36         $album->settitel(addslashes(trim($this->frm['titel'])));
37         $album->setlabel(addslashes(trim($this->frm['label'])));
38         $album->setmasternr(addslashes(trim($this->frm['masternr'])));
39         $album->setjahr(addslashes(trim($this->frm['jahr'])));
40         $album->setbemerkung(addslashes(trim($this->frm['bemerkung'])));
41         $album->setcoverfile(addslashes(trim($this->frm['coverfile'])));
42     }
43 }
```

```

99         if($this->frm['idAlbum']){
100             $err = $album->update();
101         } else {
102             $err = $album->insert();
103         }
104         $this->disconnectDB();
105
106         return;
107     } // end function addAlbumToDb
108
109 } // end class InsUpd

```

Listing 6b-1: InsUpd.php

In den ersten beiden Zeilen wird eine Textkonstante definiert, um die Titelzeile passend für Update oder Insert auszugeben. Das passiert im Konstruktor der Klasse und dort wird auch vorbereitet, dass bei einem Update auch die Daten für die Formularanzeige aus der Datenbank gelesen werden.

Die Validierungsschleife der Eingabedaten bleibt wie im Modul „Insert“.

Die Icons für *ändern* und *löschen* im Musikkatalog müssen nun mit einem Link generiert werden, der die Klasse InsUpd.php mit dem Key für das gewählte Album aufruft.

```

.
.
17 $nr = 0;
18 foreach ($rows as $row) {
19     $class = ($nr%2 == 0)?'': ' class="even"'; // style für gerade Zeilen
20     $content .= '<tr'. $class . '>';
21
22     // Alternative ohne Javascript
23     $content .= '<td><a href="index.php?module=InsUpd&act=upd&id=' .
$row['idAlbum'] . '></a></td>';
24     $content .= '<td><a href="index.php?module=Delete&id=' .
$row['idAlbum'] . '></a></td>';
25
26     $i=0;
27     foreach ($row as $key => $value) {
28         $class = ($i == 0)?' class="hide"':''; // style für Album ID (no
display)
29         $content .= '<td'. $class . '>' . $value . '</td>';
30         $i++;
31     }
32     $nr++;
33     $content .= '</tr>';
34 }
.
.

```

Listing 6b-2: Ausschnitt von table.php

Schritt 6c

Das Modul "Delete" wird eingeführt und komplettiert die Katalogverwaltung.

Es fehlt noch die Löschfunktion, die wir jetzt als Modul „Delete“ implementieren wollen. Bei Klick auf das Lösch-Icon eines Albums im Musikkatalog wird diese in der Datenbank gelöscht und die Erfolgsmeldung gibt die Albendaten nochmal aus.

Achtung: Diese Implementierung löscht das ausgewählte Album umgehend ohne Nachfrage in der Datenbank! (Wir ändern das im nächsten Schritt)

```
1  class Delete extends Page {
2      /* properties */
3      private $headline = 'Album wurde gelöscht!';
4      private $idAlbum;
5
6      public function __construct(){
7          parse_str($_SERVER['QUERY_STRING'], $param);
8          $this->idAlbum = $param['id'];
9      }
10     /* getter */
11     public function getHeadline(){return $this->headline;}
12
13     /* public methods */
14     public function getContent(){
15         $db = $this->connectDB();
16         $album = new Album($db, $this->idAlbum);
17         $row = $album->load_id();
18         $album->delete();
19         $this->disconnectDB();
20
21         $content = '<div><p>';
22         foreach ($row as $key => $value) {
23             $content .= $key . ': ' . $value . '<br>';
24         }
25         $content .= '</p></div>';
26
27         return $content;
28     }
29 }
30 } // end class Delete
```

Listing 6c-1: InsUpd.php

Schritt 7

Für die Funktionsnavigation innerhalb des Moduls "Catalog" wird ein Javascript als Dispatcher eingeführt. Das ermöglicht eine Sicherheitsabfrage vor dem Löschen eines Albums.

Das Script befindet sich im Unterordner `Catalog/assets/js` und wird in `Catalog.php` inkludiert.


```
1  /*
2   * Javascript Funktionen zur Steuerung der Katalogbearbeitung
3   *
4   * Achtung: bei abgeschaltetem Javascript im Browser funktionieren
5   * die entsprechenden Steuerelemente auf der Website nicht mehr.
6   *
7   */
8  function listAlbum(){ // album list
9      window.location.assign("index.php?module=Catalog");
10 }
11
12 function insAlbum(){ // album insert
13     window.location.assign("index.php?module=InsUpd");
14 }
15
16 function updAlbum(id){ // album update
17     window.location.assign("index.php?module=InsUpd&act=upd&id=" + id);
18 }
19
20 function delAlbum(id){ // album delete
21     del = confirm("Soll das Album " + id + " gelöscht werden?");
22     if (del == false) {
23         window.location.assign("index.php?module=Catalog");
24     } else {
25         window.location.assign("index.php?module=Delete&id=" + id);
26     }
27 }
```

Listing 7-1: PageSwitcher.js

Die Aufrufe der Script-Funktionen werden in der View `table.php` des Musikkatalogs durch Klick auf die Icons ausgelöst.

```
// Alternative mit Javascript
$content .= '<td class="pointer"></a></td>';
$content .= '<td class="pointer"></a></td>';
```

Listing 7-2: PageSwitcher.js



































Tutorial: Musikkatalog

MODULE

- Startseite
- Musikkatalog
- Cover Galerie

Meine Alben

	Interpret	Titel	Label	Master#	Jahr	Bemerkung
 	Louis Jordan	Rock'n'Roll	Mercury	8382192	1989	Rock'n'Roll der 40er/50er Jahre
 	Cat Stevens	Wild World	Pulsar	Puls 028	1992	Querschnitt aus verschiedenen Alben
 	Steve Winwood	Chronicles	Island	258595	1987	Solo-Album mit neueren Songs
 	Them	Them Featuring Van Morrison	DERAM	8209352	1990	45 Songs auf 2 CDs
 	Bob Dylan	Greatest Hits	CBS	4609079	1967	Das Wichtigste von Dylan
 	Beatles	The Beatles /	Apple	CDS	1993	The Red Album (Doppel-CD)
 	Be				1993	The Blue Album (Doppel-CD)
 	Ro				1986	The London Years auf 3 CDs
 	Ha				1987	Sammlung bekannter Hits
 	Peter Tosh	Mama Africa	EMI	CDP 7916712	1983	Reggae vom Feinsten
 	Stevie Wonder	Song Review	MOTOWN	5307572	1996	A Greatest Hits Collection
 	Simon & Garfunkel	Silent Voices	Back Biter	BB 61047	1985	Die bekanntesten Hits
 	Benny Goodman	Swing, Swing, Swing	ORBIS London	JAZ CD 004	1994	Classic Jazz Collection
 	Spider Murphy Gang	Single Hit Collection 1980-1993	EMI	7813542	1993	Bayrischer Rock'n'Roll, zeitlos gut
 	Trio Bavario	Bavario	Trikont	EFA 00159	1989	Bayrisch-Lateinamerikanische Rhythmen
 	Raffi	Bananaphone	Shoreline	CD 8062	1995	Kinderlieder aus Kanada

Album hinzufügen

localhost

Soll das Album 5 gelöscht werden?

OK Abbrechen

©2016 · Julius Caesar · Roma · Bella Italia

Abb 7-1: Löschfunktion mit Sicherheitsabfrage

Schritt 8

Die Module "Catalog", "InsUpd" und "Delete" werden entsprechend ihrer logischen Zusammengehörigkeit zu einer größeren Einheit (*Komponente Catalog*), einem *CRUD-Modul* (Create, Read, Update, Delete), zusammengefasst.

Soweit ist das ist ein rein architektonischer Umbau, bei dem im wesentlichen die Klassen *Catalog*, *InsUp* und *Delete* im Modulordner *Catalog* zusammengeführt und die neuen Ablageorte in den Lade-Anweisungen `require_once` in `index.php` angepasst werden.

```
1      /* import db/dao classes */
2      require_once( 'db/dao/Album.php' );
3
4      /* import page and module classes */
5      require_once( 'system/Page.php' );
6      $page = new Page();
7      require_once( 'system/modules/Navigation/Navigation.php' );
8      $nav = new Navigation();
9      require_once( 'system/modules/Start/Start.php' );
10     $start = new Start();
11     | require_once( 'system/modules/Catalog/Catalog.php' );
12     | $cat = new Catalog();
13     | require_once( 'system/modules/Catalog/InsUpd.php' );
14     | $ins = new InsUpd();
15     | require_once( 'system/modules/Catalog/Delete.php' );
16     | $del = new Delete();
17     | require_once( 'system/modules/Catalog/ResetDB.php' );
18     | $rdb = new ResetDB();
19     | require_once( 'system/modules/Gallery/Gallery.php' );
20     $gal = new Gallery();
...

```

Listing 8-1: Ausschnitt index.php

Um die Datenbank wieder zu restaurieren, wenn man sie beim Testen verwuzzelt hat, wird im Musikkatalog noch ein Button „Datenbank zurücksetzen“ eingefügt.

Dazu gibt es einiges zu tun:

- Neue Klasse *ResetDB* implementieren
- Neue Methode `resetDB()` in Klasse `album.php` implementieren
- Neue Funktion `resetDB()` in `PageSwitcher.js` implementieren
- Modulaufruf in `index.php` einfügen
- Button in die View `table.php` einfügen

```
1 class ResetDB extends Page {
2     /* properties */
3     private $headline = 'DB reset';
4
5     /* getter */
6     public function getHeadline(){return $this->headline;}
7
8     /* public methods */
9     public function getContent(){
10         $db = $this->connectDB();
11         $album = new Album($db);
12         $album->resetDB();
13         $this->disconnectDB();
14
15         $content = '<div><p>Datenbank wurde zurückgesetzt!</p></div>';
16
17         return $content;
18     }
19
20 } // end class ResetDB
```

Listing 8-2: Neue Klasse ResetDB

```
1 public function resetDB(){
2     $this->mysqli->query("DELETE FROM album");
3     $this->mysqli->query("INSERT INTO `album` (`idAlbum`, `interpret`,
4     `titel`, `label`, `masternr`, `jahr`, `bemerkung`, `coverfile`, `mp3file`)
5     VALUES
6     (1, 'Louis Jordan', 'Rock´n´Roll', 'Mercury', '8382192', '1989',
7     'Rock\\'n\\'Roll der 40er/50er Jahre', 'LouisJordan-RocknRoll.jpg',
8     'LouisJordan.mp3'),
9     (2, 'Cat Stevens', 'Wild World', 'Pulsar', 'Puls 028', '1992',
10    'Querschnitt aus verschiedenen Alben', 'CatStevens-WildWorld.jpg',
11    'CatStevens.mp3'),
12    ...
13 }
```

Listing 8-3: Methode resetDB in Klasse Album

```
1 <?php function resetDB(){ // reset Database
2     reset = confirm("Soll die Datenbank zurückgesetzt werden?\n\n");
3     if (reset == true) {
4         window.location.assign("index.php?module=ResetDB");
5     }
6 }
```


Listing 8-4: Funktion resetDB in PageSwitcher Script

```

1 <?php // mit Javascript
2 $content .= '<div class="button" onclick="insAlbum()">Album
  hinzufügen</div>';
3 $content .= '<div class="button" onclick="resetDB()">Datenbank
  zurücksetzen</div>';

```

Listing 8-4: Button „Datenbank zurücksetzen“ in View table.php



Tutorial: Musikkatalog (8)

MODULE

- Startseite
- Musikkatalog
- Cover Galerie

Meine Alben

	Interpret	Titel	Label	Master#	Jahr	Bemerkung
.../✗	Cat Stevens	Wild World	Pulsar	Puls 028	1992	Querschnitt aus verschiedenen Alben
.../✗	Steve Winwood	Chronicles	Island	258595	1987	Solo-Album mit neueren Songs
.../✗	Bob Dylan	Greatest Hits	CBS	4609079	1967	Das Wichtigste von Dylan
.../✗	Beatles	The Beatles / 1962-1966	Apple	CDS 7970362	1993	The Red Album (Doppel-CD)
.../✗	Rolling Stones	Singles Collection	ABKCO	8444812	1986	The London Years auf 3 CDs
.../✗	Peter Tosh	Mama Africa	EMI	CDP 7916712	1983	Reggae vom Feinsten
.../✗	Stevie Wonder	Song Review	MOTOWN	5307572	1996	A Greatest Hits Collection
.../✗	Simon & Garfunkel	Silent Voices	Back Biter	BB 61047	1985	Die bekanntesten Hits
.../✗	Benny Goodman	Swing, Swing, Swing	ORBIS London	JAZ CD 004	1994	Classic Jazz Collection
.../✗	Spider Murphy Gang	Single Hit Collection 1980-1993	EMI	7813542	1993	Bayrischer Rock'n'Roll, zeitlos gut
.../✗	Torffrock	Die Bagaluten-Fete	BMG	754321	1998	Geschichten aus Tofmoorholm

Album hinzufügen
Datenbank zurücksetzen

©2016 · Julius Caesar · Roma · Bella Italia

Abb 8-1: Projekt Musikkatalog – Schritt 8 (Endausbau)

Fertig! Fertig?

Software-Entwickler kennen den Spruch: „Software, die fertig ist, wird nicht mehr gebraucht“. Da ist was Wahres dran! So lange Software benutzt wird, findet sich auch immer etwas, das angepasst, verbessert, erweitert oder repariert werden muss. In dem Sinne gibt es kein „fertig“.

Das ist auch bei unserem Beispiel nicht anders. So wie es jetzt ist, kann man noch wenig damit anfangen. Es kann aber der Anfang für eine nützliche Software sein, ein Prototyp zum Ausbauen. Wohin der Ausbau oder Umbau gehen soll, kann jeder, der bis hier gefolgt ist, selbst bestimmen. Die geschaffene Basis trägt viele Ideen.

Wer bis hierher das Tutorial durchgestanden und verstanden hat, hat vielleicht Lust zur Vertiefung des Gelernten selbst noch etwas weiterzubasteln. Dazu seien nachfolgend noch ein paar Ideen für Übungsaufgaben mitgegeben. Sicherlich gibt es aber auch eigene Ideen zum weiteren Ausbau.

Noch ein paar Übungsaufgaben

- Fehlerbehandlung (!)
- Insert Loop (Button „weitere Alben einfügen“ o.ä. unter der Speichermeldung)
- List Button (Button „Albenliste anzeigen“ unter der Speichermeldung)
- Button „abbrechen“ (neben „speichern“) in Formular „Albumdaten bearbeiten“
- Modul in anderem Bereich anzeigen
(z.B. Zufallsanzeige eines Covers in linker Spalte unter dem Menü)
- Seitenweises Blättern in Katalog und Galerie
(die Litenanzeige ist derzeit auf 20 begrenzt)
- Navigationsmenü aus Datenbanktabelle erzeugen
- ... (was hättest *Du* gerne noch?)

Apropos „Musik“

Und schon kommt mir beim Schreiben dieses Dokumentes die nächste Idee: Wäre es nicht ein nettes Feature, auch etwas Musik mit dem Musikkatalog abzuspielen?

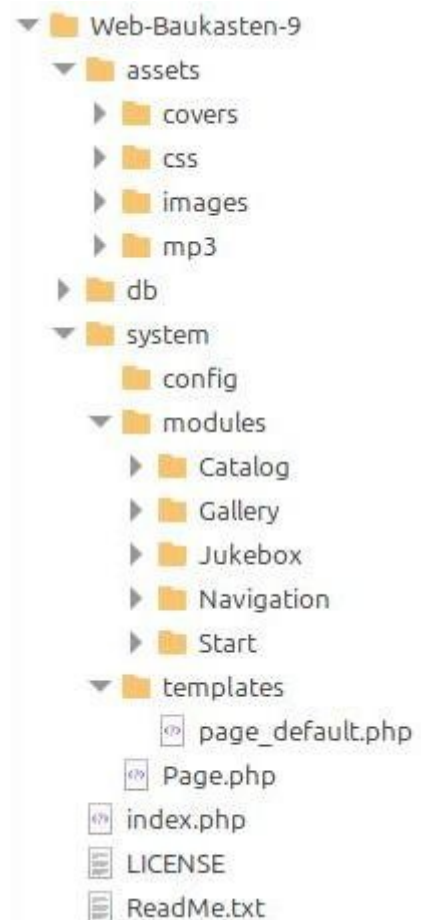
Mit dem *HTML5-Audio-Objekt* und etwas Javascript ist das gar nicht so schwer zu implementieren. Eine neue Anforderung könnte demnach z.B. sein, bei Klick auf ein Cover einen Song von dem Album als Hörprobe anzuspielden.

Bonus: Schritt 9

Ein neues Modul „Jukebox“ nutzt das HTML5-Audio-Objekt um Musik in den Musikkatalog zu bringen. Dazu muss der DB ein neues Feld für MP3-Dateien hinzugefügt werden. Das PHP-Modul liest die Daten aus der DB und übergibt sie an ein Javascript, welches für die Steuerung und Anzeige des Players sowie der Playlist sorgt.

Dazu gibt es wieder einiges zu tun:

- Asset-Ordner covers verschieben und neuen Ordner mp3 für Musikdateien anlegen
- Referenz in Gallery View `fromdb.php` anpassen
- Klasse `Jukebox.php` implementieren
- Javascript `jukebox.js` implementieren
- View (template) `viewPlayer.php` zur Anzeige von Player und Playlist implementieren
- Styles `jukebox.css` zur Gestaltung der Jukebox-Seite implementieren
- Array in Klasse `Navigation.php` erweitern
- `index.php` anpassen



Okay, let's do it!

```

1  class Jukebox extends Page {
2      /* properties */
3      private $headline = 'Sample Jukebox';
4      private $listlength = '20'; // length of displayed album list
5
6      private $styles = array(
7          '<!-- module styles -->',
8          '<link rel="stylesheet"
href="system/modules/Jukebox/assets/css/jukebox.css"/>'
9      );
10     private $scripts = array(
11         '<!-- module scripts -->',
12         '<script
src="system/modules/Jukebox/assets/js/jukebox.js"></script>'
13     );
14
15     /* getter methods */
16     public function getHeadline(){return $this->headline;}
17
18     public function getStyles() {
19         $links = parent::getStyles();
20         foreach ($this->styles as $value) {
21             $links .= $value . "\n";
22         }
23         return $links;
24     }
25     public function getScripts() {
26         $scripts = parent::getScripts();
27         foreach ($this->scripts as $value) {
28             $scripts .= $value . "\n";
29         }
30         return $scripts;
31     }
32     public function getContent(){
33         $db = $this->connectDB();
34         $album = new Album($db);
35         $content = '<div id="songlist">';
36         $fields = 'idAlbum, interpret, titel, coverfile, mp3file';
37         $rows = $album->load_fields($fields, $this->listlength);
38         // use a template to format the view
39         include_once('system/modules/Jukebox/templates/viewPlayer.php');
40         $content .= '</div>';
41
42         foreach ($rows as $row) {
43             $alben[] = $row;
44         }
45         $albenliste = json_encode($alben);
46         // Albendaten werden im JSON codiert für Javascript jukebox.js
47         // und unsichtbar in HTML-Seite bereitgestellt
48         echo '<div id="Alben" style="display: none;" >' . $albenliste .
'</div>';
49         $this->disconnectDB();
50         return $content;
51     }
52 } // end class Jukebox

```

Listing 9-1: Klasse Jukebox.php

Die „Jukebox“ soll links den Audioplayer anzeigen und rechts davon die Cover der Alben aus unserem Katalog als klickbare Playlist. Dazu implementieren wir folgende View:

```

1  $content .= '
2      <div id="playerArea" class="playerArea">
3          <div class="jukebox" align="center">
4              
5              <p id="titel" class="titel"></p>
6              <p id="artist" class="artist">click any album cover or</p>
7              <button onclick="playRandom()">play samples randomly</button>
8          </div>
9          <audio id="player" controls></audio>
10     </div>
11
12     <div id="coverlist" class="coverlist">';
13     $i = 0;
14     foreach ($rows as $row) {
15         implode(" - ", $row);
16         if (empty($row['coverfile'])){
17             $row['coverfile'] = "_NoCover.jpg";
18         }
19         $content .= "<img src='assets/covers/" . $row['coverfile'] . "'
height='80' width='80' class='song' onclick='playSong(" . $i++ . ")'>";
20     }
21
22     $content .= '</div>';

```

Listing 9-2: Template viewPlayer.php

In der Zeile 19 wird mit `onclick='playSong'` der Auslöser zum Abspielen des zum Cover gehörigen Songs gesetzt.

```

1  // Alben Daten werden von Jukebox.php aus DB gelesen
2  const alben = document.getElementById("Alben").innerHTML;
3  const albenliste = JSON.parse(alben);
4  // console.log(document.getElementById("Alben"));
5
6  const sysPath = "assets/covers/";
7  const modPath = "assets/mp3/";
8
9  // play selected song
10 function playSong(index) {
11     const player = document.getElementById("player");
12     const cover = document.getElementById("cover");
13     const titel = document.getElementById("titel");
14     const artist = document.getElementById("artist");
15
16     // ALLE 'song'-Elemente erst entselektieren:
17     // document.querySelectorAll('.song').forEach(el =>
18         // el.classList.remove('active'));
19     // JETZT dieses Element markieren: funktioniert so leider nicht :-

```



```
20     cover.src = isEmptyString(albenliste[index].coverfile) ? sysPath +
    "_NoCover.jpg" : sysPath + albenliste[index].coverfile;
21     // console.log(cover.src);
22     titel.innerText = albenliste[index].titel;
23     artist.innerText = albenliste[index].interpret;
24
25     player.src = isEmptyString(albenliste[index].mp3file) ? modPath +
    "_NoSong.mp3" : modPath + albenliste[index].mp3file;
26     // console.log(player.src);
27     player.play();
28 }
29
30 // play song radomly
31 function playRandom() {
32     const randomIndex = Math.floor(Math.random() * albenliste.length);
33     playSong(randomIndex);
34     // console.log(document.getElementById("Alben").innerHTML);
35 }
36
37 function setCover(coverfile) {
38     document.getElementById("cover").src = coverfile;;
39 }
40
41 function setVolume() {
42     const randomIndex = document.getElementById("jukebox");
43 }
44
45 // helper
46 function isEmptyString(value) {
47     value.trim();
48     return value == null || value === "";
49 }
```

Listing 9-3: Script jukebox.js

Das Script enthält die Funktionen zum Abspielen eines Songs bei Klick auf ein Cover oder auf den Button ‚randomly‘. Gleichzeitig wird das zugehörige Cover im Player angezeigt.

Das Styling der Anzeige von Player und Cover nebeneinander ist im Style Sheet jukebox.css definiert.



Abb 9-1: Die fertige „Jukebox“

Vielen Dank und viel Spaß denjenigen, die sich bis hierhin durchgebissen haben.

Anhang: CMS

Fast jeder hat heutzutage eine Website. Wie komme ich zu meiner?

„Ihre Website in 10 Minuten!“

Fürs Marketing ist, wie immer, alles ganz einfach: man installiert ein *Content Management System* (CMS) in ca. 10 Minuten und schon hat man (s)eine Website:

<https://www.youtube.com/watch?v=kpMGE8dHL4o> (dieses Werbevideo dauert 10 Minuten!)

Mag sein, wenn man die Installation schon ein paar Mal gemacht hat und dabei keinerlei Probleme auftreten, dass man die *Installation eines CMS* in 10 Minuten hinkriegt.

Allerdings: ein installiertes CMS ist noch lange nicht „Ihre Website“ und wie dieses Tutorial gezeigt haben mag, kann man für das ein oder andere Detail schon mal ein paar Stunden hinarbeiten bis es den Vorstellungen entspricht. Will man z.B. nur „unsere Website“ (also dieses *kleine* Tutorialbeispiel) *genau* so in einem CMS nachbauen, so plane man - je nach Kenntnisstand - reichlich Zeit dafür ein.

In 10 Minuten hat jedenfalls kein Mensch eine Website, die so ist, wie er sie haben möchte!

Was genau ist ein CMS

... und was kann man „Seriöses“ darüber sagen?

s. Wikipedia oder z.B.: <https://www.textbroker.de/content-management-system>

Brauche ich ein CMS für meine Website?

Das kommt ganz darauf an, ...

Eine „kleine“ Website, insbesondere, wenn nach der Ersterstellung nicht viele Änderungen und Pflege zu erwarten sind, kann man sich durchaus mit unserem Website-Baukasten bauen. Das grundsätzliche technische Verständnis sollte dieses Tutorial vermittelt haben.

Eine produktive Website mit einer gewissen Komplexität lebt allerdings meist stark und bedarf daher einiges an regelmäßiger Pflege der Funktionalitäten und Inhalte. Die Funktionalität überlässt man besser dem Entwicklerteam eines *Content Managment Systems* (CMS). Den verbleibenden Aufwand für Inhaltspflege und Gestaltung sollte man nicht unterschätzen.

Die Auswahl eines geeigneten CMS ist auch keine einfache Aufgabe, gibt es doch Hunderte davon für die verschiedensten Ansprüche. Ein Blick auf Wikipedia vermittelt einen Eindruck, was einen dabei erwartet.