|  |  |
| --- | --- |
| Gymnázium Dr. J. Pekaře Mladá Boleslav | |
| **Maturitní práce** | |
|  | |
| **DIGITALIZACE FORMULÁŘE** | |
| **pomocí programovacího jazyka PHP** | |
| **Předmět:** Informatika a výpočetní technika | |
| Datum a rok odevzdání: | Jméno a příjmení: |
| **23.3.2014** | **Pavel DVOŘÁK** |
| Ročník: **8.O** | |
| Počet stran: **20** | |
| Počet příloh: **2** | |

*Prohlášení*

*Prohlašuji, že jsem tuto maturitní práci vypracoval samostatně a že jsem uvedl všechny literární i elektronické prameny, z nichž jsem čerpal. Dále prohlašuji, že jsem při práci na projektu, jež je součástí této maturitní práce, nijak nezneužil autorská práva a že jsem všechny potřebné licence k použitému softwaru nebo kódu řádně uvedl.*

*………………………………*

Obsah

[Seznam příloh 4](#_Toc383349258)

[1 Úvod 5](#_Toc383349259)

[2 Představení použitých technologií 8](#_Toc383349260)

[2.1 HTML 8](#_Toc383349261)

[2.2 CSS 8](#_Toc383349262)

[2.3 JavaScript 9](#_Toc383349263)

[2.4 SQL a MySQL 10](#_Toc383349264)

[2.5 PHP 10](#_Toc383349265)

[2.6 Server 10](#_Toc383349266)

[2.7 Editor 11](#_Toc383349267)

[2.8 Písmo 11](#_Toc383349268)

[2.9 Ikonové písmo 11](#_Toc383349269)

[3 Uživatelská část 12](#_Toc383349270)

[3.1 Základní struktura formuláře 12](#_Toc383349271)

[3.2 Hlavička 13](#_Toc383349272)

[3.3 Tělo 14](#_Toc383349273)

[3.3.1 Struktura stránky 14](#_Toc383349274)

[3.3.2 Formulářová pole 14](#_Toc383349275)

[3.3.3 Navigace 15](#_Toc383349276)

[3.4 Vzhled 16](#_Toc383349277)

[3.4.1 Stránka 16](#_Toc383349278)

[3.4.2 Formulářové prvky 18](#_Toc383349279)

[3.4.3 Navigace 19](#_Toc383349280)

[3.4.4 Písmo 20](#_Toc383349281)

[3.4.5 Prefixy 21](#_Toc383349282)

[4 Strojová část 22](#_Toc383349283)

[4.1 PHP v HTML kódu 22](#_Toc383349284)

[4.2 Zpracování odeslaných dat do sessions 22](#_Toc383349285)

[4.3 Zápis dat do cookies 24](#_Toc383349286)

[4.4 Zápis dat do databáze 24](#_Toc383349287)

[4.5 Zápis dat do souboru 26](#_Toc383349288)

[4.6 Odeslání potvrzujícího emailu 28](#_Toc383349289)

[4.7 Získání zapsaných dat 29](#_Toc383349290)

[4.8 Vícenásobné vkládání kódu 31](#_Toc383349291)

[4.9 Detekce chyb a chybová stránka 32](#_Toc383349292)

[5 Závěr 33](#_Toc383349293)

[Seznam zdrojů 34](#_Toc383349294)

[Seznam obrázků 34](#_Toc383349295)

[Seznam ukázek kódu 34](#_Toc383349296)

[Přílohy 36](#_Toc383349297)

# Seznam příloh

[Příloha 1](#_Toc382991364)

[Příloha 2](#_Toc382991365)

# Úvod

V dobách, kdy technologie propojení počítačů pomocí sítí za účelem sdílení souborů, dat a webových prezentací pozvolna zaplavuje lidstvo, většina lidí používajících počítač a internet hledá způsob, jak by si díky novým technologiím zjednodušila práci.

Právě v těchto dobách také vychází z módy klasické papírové formuláře a pozvolna je nahrazují ty internetové. Jsou přehledné a jednodušeji uspořádané, uživatel je okamžitě upozorněn, jestliže udělá chybu a může ji snadno ihned opravit, tyto formuláře jsou také lépe strojově i klasicky čitelné, díky čemuž jsou moderní a široce oblíbené.

Jenže každý, kdo takový formulář nějakým způsobem potřebuje vytvořit, řeší vzniklý problém velmi těžce. Nabízí se samozřejmě více možností řešení takového problému. Pokud opravdu chceme jít s dobou a využít výhod webového formuláře, máme na výběr z několika variant. První možností je vytvořit daný formulář pomocí kancelářského balíku. Je to možnost sice nejjednodušší, ale postrádá mnoho výhod těch klasických webových formulářů, jaké známe. Druhou možností pak je najmout si někoho, kdo takový formulář na webové stránky vytvoří. Ale podmínkou je, mít dostatek financí, což v dnešní „šetřivé“ době není zcela optimální.

A tak některým nezbývá, než nějaký jednoduchý internetový formulář vypracovat vlastními silami. V tuto chvíli se obvykle naprostý nováček musí „prokousat“ skrz základy programování vůbec, musí se poprat hned s minimálně čtyřmi programovacími jazyky najednou a často musí také dešifrovat kód po svém předchůdci. Zde už závisí právě na člověku samotném, jakým způsobem chce nebo spíše může daný problém řešit.

První možností tvorby vlastního formuláře je tzv. redakční systém (například Drupal nebo Wordpress). V tomto případě je proniknutí do takového systému často i bez jakýchkoliv znalostí programovacích jazyků jednoduché a intuitivní. Tento člověk má štěstí a může se směle pustit do tvorby, přičemž má k dispozici nespočet knih, manuálů, dokumentace i návodů od fanoušků a to nejen na internetu, ale i v knihkupectví nebo na přednáškách. Jenže tahle skvělá možnost má svůj háček. Často by to totiž znamenalo přetvořit celý původní web, do čehož se pouštět je nejen časově, ale i fyzicky a psychicky náročné.

Proto se člověk často rozhodne naspat si takový formulář sám, přičemž na internetu příliš mnoho „kuchařek“ k tomuto tématu není, u knih takový člověk netuší, jakou příručku má zakoupit a na přednáškách se už bohužel řeší pouze poněkud pokročilé funkce, kterým z pozice začátečníka rozumět nelze. Na internetových fórech se tedy často setkáme s otázkami typu, co je tenhle (často velice triviální) úsek kódu zač a co dělá?

Rozhodl jsem se tedy takovouto kuchařku vytvořit. Cílem mojí práce je naučit kohokoliv s mírnými základy HTML a programování webových prezentací, jakým způsobem vytvořit jednoduchý, přehledný, ale hlavně funkční webový formulář a jakým způsobem jej umístit na web. Pro názornost jsem si vybral jeden konkrétní, který mi v závěrečném ročníku střední školy schází v digitální podobě asi nejvíce. Jedná se o Přihlášku ke studiu na vysoké škole.

Pro nastínění řešení toho problému jsem si vybral programovací jazyk PHP. Jedná se o „skriptovací programovací jazyk, který je určený především pro programování dynamických internetových stránek a webových aplikací například ve formátu HTML“ (1). Jeho nejčastější využití je právě zajištění komunikace mezi klientem a serverem, což je pro tvorbu formulářů velice důležitá vlastnost. Mezi jeho největší přednosti patří přehledný manuál, a dokumentace k dispozici online, jeho dynamický vývoj, který zachovává ve většině případů zpětnou kompatibilitu[[1]](#footnote-1), a naprostá jednoduchost i pro úplné začátečníky. Díky tomu a nejen díky tomu je velmi oblíbený a rozšířený napříč webhostingy.

V práci tedy nejprve čtenáře seznámím s jednotlivými technologiemi a dalšími programovacími jazyky, které pro vytvoření formuláře budou potřeba. Následně se zaměřím na uživatelskou část. Rozvrhnu základní strukturu pomocí programovacího jazyka HTML a představím základní principy pro vzhled a uživatelskou přívětivost formuláře. Poté přejdu k té složitější, avšak neméně důležité strojové části. Zaměřím se na vygenerování formuláře počítačem, zpracování údajů do něj vložených a jejich zápis do šablony, odeslání na server nebo emailem, či uložení v klientské části. A na závěr představím další techniky uložení výstupu nebo jeho zpracování, popřípadě vyvolání výstupu kvůli jeho dalšímu použití, zreviduji a otestuji vytvořený formulář a případně dovysvětlím některé pojmy nebo drobné problémy, které by mohly při návrhu nastat.

# Představení použitých technologií

Jak jsem již naznačil v úvodní kapitole, webové prezentace nesestávají z jednoho programovacího jazyka, který by zajišťoval veškeré interakce mezi uživatelem, klientem a serverem. K různým záměrům používáme různé programovací jazyky a v našem případě tomu není jinak.

## HTML

První a nejzákladnější jazyk, který budeme potřebovat, je jazyk HTML (zkr. HyperText Markup Language, česky Hypertextový značkovací jazyk). Ten se stará o základní rozvržení webové stránky a označení jednotlivých prvků v ní. Tímto jazykem vytvoříme strukturu všech stránek a částí formuláře, vložíme do ní jednotlivé formulářové prvky a také vytvoříme šablonu samotné přihlášky, do které se budou zadaná data doplňovat. Budeme také potřebovat použít některé moderní technologie. Ty však nejsou úplně zpětně kompatibilní, proto si vypomůžeme tzv. polyfilly[[2]](#footnote-2), které nám v tomto směru pomohou.

## CSS

Druhou základní složkou, doplňující HTML je využití programovacího jazyka CSS (zkr. Cascading Style Sheet, česky Kaskádové styly). Tento jazyk určuje vzhled samotných prvků. Většina klientů má již pro samotné HTML prvky přednastavený styl vzhledu, některé styly jsou dokonce zakomponovány do samotné specifikace HTML. Každopádně takový základní vzhled rozhodně nestačí k vytvoření uživatelsky přehledného a příjemného formuláře, pro naše účely se tedy budou některé úpravy hodit. Jeho výhodou je možnost ke každému prvku HTML struktury přiřadit libovolný počet libovolných možností nastavení vzhledu jednoduchou formou. V poslední době se snaží klasické CSS nahradit tzv. CSS preprocesory, které rozšiřují samotné CSS o ještě rozsáhlejší kaskádovost nebo o funkce skriptovacích jazyků (například o proměnné). Jsou však o mnoho složitější a jejich pochopení vyžaduje pokročilé znalosti programování, proto pro naše potřeby bude bohatě stačit základní CSS.

## JavaScript

Jedním z dalších základních programovacích jazyků pro návrh webu je široce rozšířený JavaScript (čti „džavaskript“). Jedná se o skriptovací jazyk, jehož základní předností ve tvorbě webových stránek je jeho výrazná podpora práce s DOM[[3]](#footnote-3) stránky, díky čemuž je nejvíce rozšířen mezi webovými prohlížeči. Jeho nevýhody však velmi markantně ovlivňují tvorbu webových aplikací, proto není příliš vhodné jím navrhovat obsah stránky a proto jej také používat pro tento formulář až na výjimečné případy nebudeme. Pro názorný příklad jsem vybral ty nejmarkantnější nevýhody, jakými JavaScript trpí:

1. Uživatel si jej může ve svém klientovi (v našem případě ve webovém prohlížeči) úplně vypnout
2. Prohlížeče (převážně mobilní) často vypnutí JavaScriptu provádějí také automaticky, pokud je pomalé internetové připojení nebo pokud vědí, že dané zařízení (například mobilní telefon) nebude schopno tento kód správně interpretovat
3. Podpora jeho jednotlivých funkcí napříč prohlížeči se velmi liší a ani v nejnovějších prohlížečích nefungují funkce z několika let starých specifikací. Dalo by se říci, že webové prohlížeče si JavaScript interpretují každý po svém.
4. JavaScript je díky podmínkám jeho vzniku (vznikl v době označované jako „válka prohlížečů“ a byl jejím hlavním předmětem sporů, každý prohlížeč totiž vyvíjel vlastní skriptovací jazyk pro tyto účely a JavaScript byl ten, který zvítězil) velice programátorsky neintuitivní a obtížně použivatelný s mnoha chybami a nejasnými funkcemi
5. Jeho funkce jsou také velmi zastaralé nebo zastarale interpretované, mluví se o jeho nahrazení v dohledné době, to však pomalu způsobuje možné další pokračování války prohlížečů, protože každý prohlížeč se připravuje na jeho nahrazení jiným jazykem (často jazykem z vlastních dílen).

## SQL a MySQL

Další důležitý jazyk pro naplnění našich představ a potřeb pro uložení daného formuláře je jazyk SQL (zkr. Structured Query Language, česky Strukturovaný dotazovací jazyk). Jako takový je vhodný pro mnoho účelů, často při komunikaci mezi dvěma zařízeními. Jeho nejčastější využití však spočívá v ukládání dat do databází a jejich vyvolávání zpět. Na jazyku SQL je postaveno mnoho databázových systémů. Velmi rozšířenou mezi webhostingy a také velmi populární mezi programátory je databáze MySQL (2). Databázový systém MySQL je navíc velice rychlý a spolehlivý, proto jej využijeme ke správě naší databáze vyplněných a odeslaných formulářů. Je nezbytně nutné, aby bylo rozhraní MySQL na serveru nainstalováno jakožto CGI skript (jakožto rozšíření Apache serveru), nikoliv jako spustitelný program a to alespoň ve verzi 5, nejlépe však v té nejaktuálnější.

## PHP

A konečně se na závěr dostáváme k jádru celého našeho počínání. PHP (rekurzivní zkr. PHP: Hypertext Preprocessor, česky PHP: hypertextový preprocesor). Ten jsem již představil v úvodu, proto teď nebudu zabíhat do podrobností. Je nezbytně nutné, aby bylo rozhraní PHP na serveru nainstalováno také jakožto CGI skript a také alespoň ve verzi 5, nejlépe však v té nejaktuálnější.

## Server

Pro naše účely bude důležité, aby webové stránky běžely na tzv. Apache httpd serveru. Podle Wikipedie v současné době dodává prohlížečům na celém světě většinu internetových stránek (3), proto si myslím, že by s jeho zajištěním na straně webhostingu neměl být problém. Budeme si na něm ukazovat některá drobná nastavení. Samozřejmě ta jdou provést také v dalších serverech (jako např. Nginx), ovšem vyžaduje to pokročilou orientaci v takovém serveru a převedení všech příkazů a nastavení z Apache.

## Editor

Jednou z důležitých pomůcek je bezesporu také kvalitní textový editor. Na výběr je mnoho možností od těch nejzákladnějších, jako například poznámkový blok nebo tzv. Notepad2, až po ty pokročilejší, mezi kterými bych vybral například český PSPad. O výběr rozhodně není nouze, je však důležité, aby na něj byl programátor zvyklý a aby se mu v editoru dobře pracovalo. Pro jednoduchou a přehlednou správu projektů je doporučován i editor Brackets, který použiji k ukázkám kódu.

## Písmo

Najít vhodné písmo pro webový formulář nebo čistě pro webovou stránku je velmi složitý oříšek. Naštěstí je pro tyto potřeby na internetu k nalezení mnoho zajímavých článků o „typeface“, čili o využití různých druhů písem při různých příležitostech. Těmi se však mnoho zabývat nebudeme, protože to není předmětem této práce. Nejjednodušší pro nás bude využít písmo z databáze Google Fonts, která nabízí možnost vybrat si hezké a designově propracované písmo a vložit jej přímo do webové stránky. Doporučuji ještě využít aplikaci českého vývojáře zvanou „Fontokuk“ (4), která z písem dostupných v databázi Google Fonts vybere pouze ty, které zobrazují českou diakritiku. Pro naše účely jsem jako příklad vybral uživatelsky příjemné a velice oblíbené a rozšířené písmo Ubuntu.

## Ikonové písmo

Poslední důležitou roli v navigaci ve formuláři hrají moderní ikonová písma. Jde o klasická písma pro zobrazení textu, namísto písmen textu se však zobrazí ikonky. Písma se dají buď stahovat a používat jednotlivě jako celek (tak jako klasická písma) nebo si lze pomocí různých online služeb namixovat z dostupných ikonek písmo dle vlastního výběru. My to však nebudeme komplikovat a pro názorný příklad využijeme jedno z již přednastavených písem MFG Labs icon set, které je dostupné ke stažení zdarma.

# Uživatelská část

## Základní struktura formuláře

I pro nováčky v HTML by tato kapitola měla být spíš opakováním. Začneme nejprve základní strukturou webové stránky.



Ukázka : Základní struktura HTML stránky

Každá HTML stránka by měla obsahovat nejprve typ dokumentu, neboli tzv. *doctype*. V něm je deklarován formát dokumentu. Samotný obsah stránky ohraničují párové značky *<html>*. Obsah se dělí na hlavičku (párové značky *<head>*) a tělo (párové značky *<body>*). Do hlavičky se zapisují informace, které nejsou uživateli viditelné a slouží pro zpracování strojově. Naopak do těla se píše přímo samostatný obsah stránky. Do hlavičky ještě doplníme titulek stránky (párové značky *<title>*), který se zobrazí v názvu otevřené záložky v prohlížeči.

Nesmíme ještě zapomenout na nepárovou značku *<meta>* s vlastností *charset*, čili sadou znaků. Základní sada znaků je v českém prostředí většinou *windows-1250*, pokročilá zase třeba *iso-8859-2*, ale nejuniverzálnější pro náš účel bude mezinárodní sada znaků Unicode (*utf-8*). Každý editor by měl nějakým způsobem nabídnout alespoň dvě sady znaků k výběru ať už při psaní, nebo při ukládání dokumentu. Sadu znaků později už měnit nemůžeme, proto si ji nadefinujeme teď. Zvolenou sadu znaků musíme poté použít naprosto stejnou všude, kde bude potřeba.

Takto připravenou šablonu zapíšeme tedy do editoru a uložíme s příponou *.php*. Úvodní stránka by se na všech serverech Apache měla jmenovat *index.xyz*, proto náš první soubor bude mít výsledný název *index.php*. Pokud se náš formulář nevejde na jednu stránku, další stránky můžeme buď sloučit do jednoho dokumentu a pomocí skriptovacích jazyků upravit přecházení mezi nimi, nebo mnohem jednodušeji rozdělit formulář do více stránek. Pro každou další stránku poté můžeme zvolit název, jaký se nám zlíbí. Avšak opatrně, nejlépe bez diakritiky a samozřejmě tak, aby se název vztahoval k obsahu.

Ještě podotknu, že vše, co jsme zatím do souboru napsali, může být tzv. „case insensitive“, čili může být zapsáno jak velkými, tak malými písmeny, a může být zapsáno také kombinovaně. V ukázkách používám první písmena velká pouze pro přehlednost.

## Hlavička

Do hlavičky bychom měli zapsat ještě další informace o dokumentu. Jednak by měla obsahovat zadání písem a stylů, také bychom měli uvést tzv. SEO údaje o stránce pro webové vyhledávače, případně deklarovat další aplikace a polyfilly, které použijeme.



Ukázka : Hlavička stránky

Dalšími *<meta>* značkami a jejich vlastnostmi (viz Ukázka 2) doplníme tedy libovolné popisky pro webové vyhledávače. K dispozici máme název, autora, popis, generátor (jinými slovy editor) a klíčová slova (oddělují se čárkou). Další *<meta>* značka s vlastností *robots* udává, jakým způsobem se mají vyhledávače k obsahu chovat. *follow* znamená, že mají následovat odkazy a ukládat také stránky, na které odkazujeme. *index* zase určuje, že se ukládá stránka samotná, a *archive* určuje, zda se má stránka archivovat (ukládat její kopie vyhledávači, či archivátory stránek). Případnou negaci provedeme přidáním předpony *no* (př.: *nofollow*).

Následují nepárové značky *<link>* pro vkládání obsahu do stránky. První vloží ikonu stránky, která se zobrazí u jejího titulku. Ikona by se měla kvůli zpětné kompatibilitě jmenovat *favicon.ico*. Další jsou písma, jejichž vkládání je popsáno v dokumentaci u každého písma, případně písmo vložíme přímo do našich stylů (viz kapitola 3.4 Vzhled). Na závěr vložíme naše styly buď v externím souboru, nebo přímo.

## Tělo

### Struktura stránky

Základní rozvržení stránky samotné pomocí HTML bude v našem případě sestávat ze dvou navigačních menu a samotného formuláře.



Ukázka : Tělo stránky

Navigační menu jsem ohraničil párovými značkami *<div>* a označil třídou, abychom na ně mohli později aplikovat různé styly vzhledu. K formuláři (párové značky *<form>*) patří vlastnosti *action* a *method*. Vlastnost *action* určí soubor nebo skript, který zpracuje zadaná data formuláře. Vlastnost *method* zase určuje způsob předávání zadaných dat ke zpracování. Metoda *GET* předává data přímo v adrese url, ta jsou tedy viditelná návštěvníkovi stránky a jsou jednodušeji zpracovatelné různými druhy jazyků. Oproti tomu metoda *POST* předává data skrytě, na pozadí. Taková data jsou sice čitelná užším výběrem jazyků, lépe se však hodí pro větší objem dat a zaručují vyšší bezpečnost, proto jsem se rozhodl metodu *POST* použít. Na závěr do formuláře dopíšeme ještě vlastnost *id*, kterou formulář pojmenujeme pro případ, kdy budeme chtít některý z prvků formuláře (například odesílací tlačítko) umístit mimo samotný formulář (například do jednoho z menu – viz kapitola 3.3.3 Navigace).

### Formulářová pole

Nejdůležitější součástí formuláře jsou pro nás jednotlivá formulářová pole. Vkládají se pomocí prvků *<input>*, *<textarea>*, *<select>*, *<option>* a *<button>*. Více prvků můžeme rozdělit do skupin párovými značkami *<fieldset>*, do kterých se nepovinně uvádí také párové značky *<legend>*, jejíž úlohou je nadřazené skupině přidělit název. Obecně se doporučuje značku *fieldset* používat, jelikož tato značka fakticky zpřehlední a zjednoduší orientaci uživatele ve formuláři.



Ukázka : Jednotlivé prvky formuláře

Do jednotlivých skupin poté vkládáme prvky uvedené výše. Nepárovými značkami *<input>* vkládáme samotná vstupní formulářová pole. Vlastností *type* poté určíme, jakého typu bude toto pole. Na výběr máme například nejzákladnější textové, zaškrtávací, přepínací a číselné pole, dále pole pro vkládání data, času, barvy, telefonu nebo emailu. Jenom upozorním, že pole pro vkládání data, času a barvy nejsou příliš bezpečná k používání, protože je starší prohlížeče příliš nepodporují. Pro náš záměr však příliš potřeba nebudou, proto se nemusíme příliš bát. Jejich vložení by kvůli zpětné kompatibilitě vyžadovalo doplnění o nějaký JavaScriptový polyfill (viz kapitola 2.2.3 Polyfilly).

Párovými značkami *<textarea>* můžeme vložit pole ke psaní delších textů. Párové značky *<select>* zase vkládají pole pro výběr hodnoty ze seznamu. Jednotlivé hodnoty se vkládají pomocí párových značek *<option>*. Posledním často používaným prvkem HTML formulářů je tlačítko. Zapisuje se párovými značkami *<button>* a vlastnost *type* určuje, jakým způsobem se bude tlačítko chovat. *reset* vynuluje veškerá zadaná data, *submit* data odešle a *button* plní funkci byzvýznamového tlačítka, kterému můžeme přiřadit funkci skriptovacími jazyky. Text, který se zobrazuje v tlačítku, pak můžeme nastavit mezi párové značky.

Všechny formulářové prvky dále obsahují dvojici vlastností *name* a *value* (viz Ukázka 4), které určují název a přednastavenou hodnotu předávaných dat. Pro jednodušší orientaci handicapovaných uživatelů ve formuláři je také důležité nastavit vlastnost *tabindex*, která určuje pořadí prvků při přepínání mezi nimi pomocí klávesy TAB (například u nevidomých je tato vlastnost klíčová). Ke každému poli se ještě přidává párovými značkami označený prvek *<label>* sloužící jako jeho popisek. Používá se s vlastností *for*, do které se vepíše id formulářového pole, ke kterému se prvek váže.

### Navigace

Součástí kvalitního formuláře by měla být také kvalitní navigace. Navigací myslím stránkování a intuitivní přecházení mezi jednotlivými stránkami, možnosti uložení anebo odeslání formuláře v průběhu i na konci vyplňování. Velkou roli v tomto ohledu hrají právě tlačítka a skupiny polí, která jsme si připomenuli v předchozí kapitole. Veškerou práci v našem formuláři přenecháme souboru *zpracovani.php*, proto můžeme u všech tlačítek nastavit vlastnost *type* na hodnotu *submit*. Pro tlačítka v navigačních menu (tj. mimo formulář samotný) musíme ještě dodat vlastnost *form*, jejíž hodnotu nastavíme na id formuláře, ke kterému se tlačítko váže.

## Vzhled

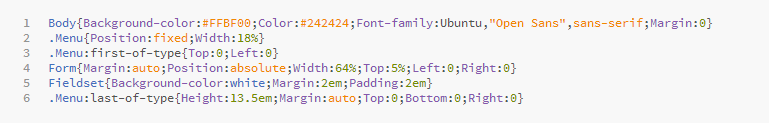
Chceme-li, aby byl formulář „user-friendly“, čili uživatelsky příjemný k vyplnění, nesmíme jej zanechat v přednastavených stylech. Vytvoříme si tedy soubor *styly.css*, do nějž pomocí kaskádových stylů (viz kapitola 2.1.2 CSS) vepíšeme naše parametry.



Obrázek : Jednoduchý formulář bez nastavených stylů

### Stránka

Pro základní rozestavení a barevné schéma stránky budeme nyní pracovat s předpřipraveným HTML dokumentem. Znovu musím dodat, že u všech vlastností, pokud neuvedu jinak, nezáleží na velikosti písmen, pouze já jsem použil první velká pro přehlednost.



Ukázka : Styly stránky a formuláře

Kaskádové styly sestávají vždy z označení HTML prvku, za nímž následují složené závorky. Do těch se poté vkládají jednotlivé styly oddělené středníkem. Každý jednotlivý styl je označen názvem, následuje dvojtečka a za ní hodnota. Styly vkládáme buď do HTML dokumentu přímo (do hlavičky párovými značkami *<style>*) nebo do odděleného souboru s příponou *.css*, který do HTML stránky vložíme nepárovou značkou *<link>*.

Nejprve si nastavíme společné styly pro tělo stránky *body*. Do složených závorek jsem zadal barvu pozadí a barvu textu (parametry *background-color* a *color*). Jejich hodnotami může být buď anglický název barvy, nebo RGB hodnota barvy, kterou můžeme zadat buď v šestnáctkové soustavě (křížek a dvě číslice pro každou z barev) nebo v desítkové soustavě (heslo rgb a kulaté závorky, v nichž jsou hodnoty pro každou z barev odděleny čárkami). Dále jsem zadal typ písma (viz kapitola 3.4.4 Písmo) a na závěr jsem vynuloval přednastavené okraje stránky.

Navigaci jsem označil třídou *Menu*, proto styl pro navigaci můžeme nyní označit tečkou a názvem třídy. Ale pozor, u tříd záleží na velikosti písma a je tedy rozdíl mezi *.MENU* a *.menu*. Pro označení prvního a posledního menu jsem k *.menu* přidal takzvanou pseudotřídu. Ta se označuje dvojtečkou a zadaným heslem, v našem případě tedy *:first-of-type*, *:last-of-type* nebo třeba *:nth-of-type(n)*, přičemž místo *n* doplníme pořadové číslo prvku. Jelikož chceme, aby navigace zůstala uživateli viditelná vždy, i když uživatel odroluje na stránce níž, nastavíme parametr *position* na hodnotu *fixed*. Nyní ještě doplníme pozici těchto menu parametry vzdáleností od okrajů stránky *top*, *left*, *bottom* a *right*. Také můžeme doplnit parametry výšky a šířky *height* a *width*. Všechny tyto vlastnosti se mohou nastavit v různých jednotkách. K dispozici máme pixely (px), centimetry a milimetry (cm, mm), relativní výšku základního velkého písma M (em) nebo malého písma x (ex), případně i procenta (%) a dalších asi dvanáct jednotek. Nejčastější jsou však ty výše vypsané a s nimi budeme pracovat.

Chceme-li nějaký prvek vycentrovat (ať už horizontálně, či vertikálně), použijeme drobný „hack“ ve formě automatických okrajů. Pro tento prvek nastavíme *margin*, prostor mezi prvkem a jeho obalem, na hodnotu *auto* a dále musíme nastavit vzdálenosti od okraje stránky ve směru, ve kterém chceme prvek vycentrovat, na nulovou hodnotu. Nesmíme zapomenout také nastavit rozměr prvku v daném směru (*height* nebo *width*). Ještě doplním, že chceme-li určitý parametr nastavit na nulovou hodnotu, nemusíme uvést žádnou jednotku vzdálenosti. Ještě je tu skrytá podmínka, že prvek musí mít také parametr *position* nastavený na jakoukoliv hodnotu kromě *relative*. Prvek vycentrovaný v horizontálním i vertikálním směru by tedy v kaskádových stylech měl nastaveno toto:



Ukázka : Absolutně vycentrovaný prvek

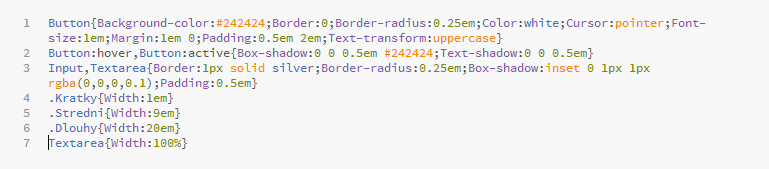
Formulář by měl být v oku příjemných barvách a s dostatečnými meziprostory kvůli uživatelské přístupnosti. Jednotlivá pole by neměla být natlačena jedno na druhé a barvy by neměly být příliš křiklavé. Nejjednodušší je tedy černobílý vzhled formuláře. O dodatečný prostor mezi prvkem a jeho obsahem se stará parametr *padding*, o dodatečný prostor mezi prvkem a jeho obalem zase výše zmíněný *margin*. Ten ovšem není určen pro formulářové prvky, proto u nich *margin* použít nemůžeme.

### Formulářové prvky



Obrázek : Jednoduchý formulář s nastavenými styly

Pro formulářová pole je časté zaoblování rohů, a stínování. Těchto efektů dosáhneme pomocí parametrů *border-radius* pro zaoblené rohy a *box-shadow* (případně *text-shadow*) pro stíny. Parametr pro stínování obsahuje posunutí ve vertikálním směru, posunutí v horizontálním směru a velikost rozmazání stínu. Může také obsahovat heslo *inset* na začátku, pokud potřebujeme nastavit vnitřní stín. Barva stínu se bere z barvy textu, můžeme ji však změnit přidáním libovolné barvy na závěr parametru. Můžeme si tedy všimnout stínu pro text i samotný prvek například u stylu pro *button:hover* a *button:active*. Přípona začínající dvojtečkou v tomto případě znamená stav daného prvku. Přípona *:hover* znamená, že se nad prvkem nachází myš (styly se použijí při najetí myši), *:active* zase znamená, že je prvek aktivován (je na něj kliknuto).



Ukázka : Různé styly formulářových prvků

Když už jsem zmínil vzhled tlačítek, bylo by dobré jejich vzhled ještě vylepšit. Můžeme například vynulovat nebo nastavit okrajovou linii tlačítka pomocí parametru *border*. Hodnota tohoto parametru sestává z tloušťky, stylu (plný, čerchovaný, apod.) a barvy. Jednotlivé hodnoty musíme oddělit mezerami. Všimněte si také parametrů *font-size* a *text-transform*. První určuje velikost písma, druhá zase velikost písmen. U *text-transform* máme k dispozici hodnoty *uppercase* pro velká písmena, *lowercase* pro malá písmena nebo *capitalize* pro první písmena velká. Nebo můžeme hodnotu vynulovat heslem *none*.

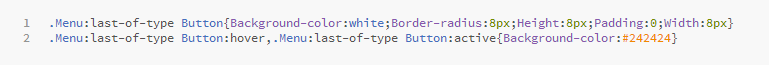
Můžete si ještě všimnout, že jsem tlačítku doplnil parametr *cursor* s hodnotou *pointer*, která zajistí, že kurzor myši nad tlačítkem bude „ukazující ruka“. Možná by mi někdo vytknul, že nad tlačítky se kurzor mění na ruku vždy. Pokud ovšem změníme vzhled tlačítka kaskádovými styly, musíme znovu nastavit také chování kurzoru. Jde pravděpodobně o nějakou historicky podloženou chybu prohlížečů, raději se proto podřídíme.

Dále si povšimněte, že jsem nastavil délku textových polí. Ke každému vstupnímu textovému poli bude zapotřebí přiřadit jednu z nastavených tříd. Tímto jsem využil kaskádovosti jazyka CSS, který mi umožňuje nastavit jeden vzhled pro více prvků. Zároveň si můžete všimnout více prvků před složenou závorkou. Tyto prvky jsou od sebe odděleny čárkou a pro všechny taktéž platí uvedené styly.

### Navigace

Jestliže chceme uživateli nabídnout plný uživatelský komfort, musíme mu umožnit se po našem formuláři jednoduše a rychle pohybovat. Proto bychom měli přidat nejen tlačítko na další stránku formuláře, ale také tlačítko zpět, menu pro přechod na libovolnou stránku a možnost kdykoliv odeslat formulář ke schválení nebo si data průběžně uložit. Na tohle vše by nám měly posloužit tlačítka (*<button>*), případně odkazy (*<a>*).

Oblíbená a moderní jsou v dnešní době také tzv. „bullets“, čili kulatá tlačítka ve tvaru puntíku. Jejich vytvoření není nijak složité. V HTML místo názvu vepíšeme *&nbsp;*, což je zástupný ASCII symbol pro nezlomitelnou mezeru (používanou v případech, kdy nechceme, aby se mezi slovy zalomoval řádek). Kdybychom tam vepsali klasickou mezeru nebo nechali text tlačítka prázdný, tlačítko by se vůbec nezobrazilo. Takto jsme si vytvořili tedy prázdné tlačítko a nyní nám chybí zaoblit jeho okraje dokulata.



Ukázka :Kulatá tlačítka

Další funkcí Kaskádových stylů je možnost vybrat prvek, který je uvnitř prvku jiného. Používá se k tomu jednoduše mezera. V příkladu tedy vybírám všechna tlačítka z druhého menu a přiděluji jim styly. Chci-li dosáhnout kulatého tlačítka, musím mu nejprve nastavit stejnou šířku i výšku, a poté nastavit zaoblené rohy tak, aby bylo zaoblení minimálně stejně veliké, jako šířka a výška. Nakonec už jenom vynuluji vnitřní okraje v tlačítku a přidám barvu pro samotné tlačítko i barvu pro aktivní tlačítko. Výsledek poté vypadá zhruba takto:



Obrázek : Kulatá tlačítka

### Písmo

Slíbil jsem, že se na písmo a jeho vkládání podíváme blíže. Slib dodržím, ale upozorňuji, že je to jedna z náročnějších funkcí našeho formuláře. Možností pro vložení písma do stránky je samozřejmě více. Záleží na tom, v jakém formátu písmo máme uložené. Nejjednodušší k vložení jsou tzv. systémová písma (např.: Times New Roman nebo Arial). Ta jsou součástí všech operačních systémů a můžeme je tedy rovnou použít v našich stylech pomocí parametru *font-family*:



Ukázka : Vložení písma do souboru

Jednotlivá použitá písma se oddělují čárkami, víceslovné názvy písem se uzavírají do uvozovek. Platí zde pravidlo, že prohlížeč pokusí načíst první písmo, a pokud se mu nepovede jej načíst, přejde na další v řadě. Na závěr je tedy vhodné vložit jedno z klíčových slov *serif* (patkové), *sans-serif* (bezpatkové), *cursive* (stočené, psací), *fantasy* (dekorativní, expresivní) a *monospace* (se stejnou šířkou písmen, počítačové). Prohlížeč pak za dané klíčové slovo dosadí nejběžnější systémové písmo pro danou kategorii (např.: Times New Roman pro klíčové slovo *serif*).

Horší jsou však další případy, kdy chceme použít nesystémové písmo. Jestliže jsme takové písmo získali z online databází (například z Google Fonts), pak je u každého písma doplněný většinou i návod k použití, často i s kódem. Ten pak stačí už jen vložit na příslušné místo a vše je hotovo.

Největší problém nám však bude dělat písmo, které máme k dispozici v souboru na našem počítači. Takové písmo je většinou ve formátu *.ttf* nebo *.otf*. Zde nejprve musíme písmo zkonvertovat do dalších formátů pomocí nějakého konvertoru. Písmo kvůli zpětné kompatibilitě totiž budeme potřebovat ve formátech *.eot*, *.svg*, *.woff* a *.ttf* nebo *.otf*. Takto připravené čtyři soubory poté musíme vepsat do našich stylů přesně podle zadané formule:



Ukázka : Formule pro vložení písma do stránky

### Prefixy

Na závěr uživatelské části se ještě musím zmínit o tzv. „prefixes“, čili předponách u jednotlivých stylů. Některé styly, které jsme použili výše, totiž nejsou zcela zpětně kompatibilní. Jedná se například o zaoblené rohy. U takových si poté musíme zjistit, jaké předpony máme doplnit těmto stylům, aby fungovali i na některých starších prohlížečích. K tomu nám může posloužit například internetová stránka CanIUse.com, která obsahuje databázi všech parametrů Kaskádových stylů, které je nějakým způsobem třeba opravit pro zpětnou kompatibilitu. Taková předpona sestává z pomlčky, jména prohlížeče, který tuto funkci podporuje ve starších verzích, pomlčky a poté již následuje klasický zápis stylu. Pro zaoblené rohy tedy plný prefixovaný kód bude vypadat následovně:



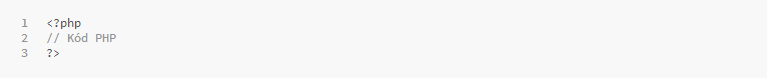
Ukázka : Prefixované zaoblené rohy u tlačítka

# Strojová část

Dosud jsme se zabývali strukturou a vzhledem formuláře. Ten by nyní měl vypadat uživatelsky i designově perfektně, stále nám však nefunguje ani jedno tlačítko. Nyní tedy přichází na scénu ta nejdůležitější část zajišťující jeho funkčnost.

## PHP v HTML kódu

Abychom mohli do souboru umístit kód PHP, musíme jej od ostatního obsahu souboru (například od kódu HTML nebo od Kaskádových stylů) oddělit. K tomu nám slouží otevírací *<?php* a zavírací *?>* značky. Mezi tyto značky poté umístíme náš kód. Vždy jej ale od těchto značek musíme oddělit alespoň jednou mezerou nebo novým řádkem. Samotný kód pak obsahuje jednotlivé příkazy, které oddělujeme středníkem.



Ukázka : vložení PHP kódu do HTML dokumentu

## Zpracování odeslaných dat do sessions

Funkce *sessions* nám v PHP umožňují pracovat s daty mezi více stránkami. Jejich využití je tedy například u vícestránkového formuláře více než jasné. *Session* nám umožní průběžně ukládat data, dále je zpracovávat a doplňovat již zadaná data, dále je předávat mezi stránkami, zapisovat je do souboru, či s nimi jinak nakládat. Každá *session* má navíc svoje vlastní unikátní ID, jehož pomocí budeme moct zadaná data třídit a ukládat.

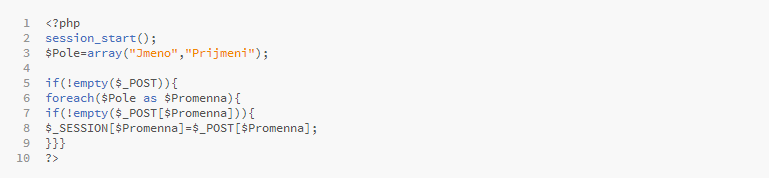
Pokud chceme s daty pracovat pomocí *sessions*, musíme nejprve *session* spustit. K tomu slouží příkaz *session\_start()*, který nejenže spustí v daném souboru *session*, ale také získá *session* data z předchozí stránky. Je potřeba jej proto uvést v každém souboru, kde chceme s daty v *session* pracovat. Data odeslaná z formuláře se ale neposílají skrz *session*, nýbrž pomocí polí *POST* nebo *GET*. Proto je z těchto polí musíme nejprve do *session* přepsat. To uděláme pomocí příkazů *$\_POST[‘jmeno‘]* nebo *$\_GET[‘jmeno‘]*. Do proměnné poté data z takových polí přepíšeme takto: *$Jmeno=$\_POST[‘jmeno‘]*. Obsah těchto proměnných se ale při odchodu z dané stránky smaže, proto je přepíšeme do speciálních *session* proměnných. Náš kód tedy bude vypadat takto: *$\_SESSION[‘jmeno‘]=$\_POST[‘jmeno‘]*.

Problém však nastává, pokud má uživatel vypnutá *cookies* ve svém prohlížeči. *Sessions* jsou totiž jinými slovy *cookies* s nulovou platností, což znamená, že jsou data v *sessions* uchovávána pouze do doby, než je *session* ukončena nebo je vypnut prohlížeč. Jelikož však *sessions* díky svému krátkodobému charakteru nejsou pravé *cookies*, jedná se tak o chybu. Na tuto chybu naštěstí PHP pamatuje, proto pokud je v prohlížeči vypnuto používání *cookies*, lze předávání dat v *session* opravit tím, že se v adrese url za znakem otazníku u všech odkazů, které vedou na stránku, kde ještě budeme s daty pracovat, uvede tzv. *session id*. Uděláme tak pomocí funkce *echo()*, který jednoduše vypíše hodnotu proměnné nebo text v uvozovkách. V našem případě právě vypíše PHP proměnnou *session\_id()*.



Ukázka : Doplnění *session id* do adresy url

Pokud máme velmi mnoho proměnných, můžeme použít PHP pole. Názvy jednotlivých formulářových prvků můžeme přepsat do jednoho pole (*array*), a poté je zpracovat cyklem *foreach*. Kód v souboru *zpracovani.php* tedy zatím bude vypadat zhruba takto:



Ukázka : Zpracování velkého množství dat pomocí pole a foreach cyklu

V ukázce máme pole v proměnné *$Pole*, v němž jsme vypsali jména všech formulářových prvků, které chceme zpracovat. Ta ještě nesmíme zapomenout obalit uvozovkami a oddělit čárkami. Poté následuje *foreach* cyklus, který nám z pole vybere veškeré hodnoty a pro každou hodnotu přepíše data z *POST* do *session*. Nesmíme se ještě zapomenout ujistit, že pole *POST* není prázdné, tedy že uživatel do formuláře zadal nějaké hodnoty, obdobně také u jednotlivých prvků formuláře, abychom zbytečně nezapisovali prázdná data.

## Zápis dat do cookies

Nyní již máme data přepsaná v *session* proměnných. Naším dalším cílem je tato data uložit a zapsat. Na výběr máme více cest. Data můžeme zapsat jak na server, tak do paměti prohlížeče. Pro zápis dat na server používáme nejčastěji *databáze*, v prohlížeči zase využijeme *cookies*.

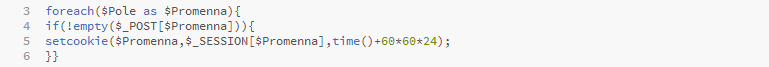
*Cookies* zapisují data do paměti prohlížeče. Uživatel je tím pádem může omezit, či úplně zakázat, navíc je může kdykoliv smazat, což nám neumožňuje s nimi pracovat jako s hlavním úložným zařízením. Z pozice sběratele formulářů je také nemůžeme nijak číst. Proto do nich zapíšeme pouze dočasná data, která chceme, aby vydržela v paměti déle, než *session* data, a jejichž uplatnění bude sloužit jako záložní například v případě, že v průběhu vyplňování formuláře dojde k nějaké chybě, ať už na straně uživatele nebo serveru.

Zápis se provádí příkazem *setcookie()*. Ten má v závorce tři parametry oddělené čárkou. Prvním parametrem je jméno *cookie*, druhým její hodnota a třetím čas ve vteřinách od 1. ledna 1970, do kdy bude k dispozici. Pro třetí parametr udávající čas můžeme použít PHP proměnnou *time()*, která vypíše aktuální čas. K ní pak můžeme přičíst samotnou dobu, po kterou chceme data zachovat. Zápis hodnoty by pak mohl vypadat například takto:



Ukázka : Uložení jednoduché hodnoty do *cookie* na dobu jednoho dne

Větší množství dat zapíšeme podobně, jako jsme je zpracovali. Také využijeme *foreach* cyklu, *$Promenna* stále zastává jednotlivá jména formulářových polí a do *cookies* se tedy zapíše jméno i hodnota.



Ukázka : Zápis většího množství dat hodnot do *cookies*

## Zápis dat do databáze

Nejbezpečnější možností zápisu dat jsou bezpochyby serverové databáze. Důležité je mít na serveru databázi přednastavenou a mít připravenou tabulku s kolonkami, do kterých budeme data vkládat. Vše můžeme připravit například pomocí nástroje *phpMyAdmin*.

S databází MySQL pak můžeme v PHP pracovat třemi různými způsoby. Prvním z nich je procedurální rozšíření PHP MySQL. Je to původní a nejstarší možnost, jak s databází pomocí PHP pracovat. Má to ale háček. Toto rozšíření od verze MySQL 4.1.3 neumožňuje používat některé nové funkce tohoto jazyka, proto není doporučeno jej používat (5). Druhou možností je tzv. *PDO*, což je objektově orientovaná vrstva PHP pro práci s databázemi. Je vyvíjena tak, aby pomocí této vrstvy bylo možno pracovat s více druhy databází, nejenom s MySQL, avšak pro začátečníky není kvůli jejímu objektově orientovanému přístupu zcela vhodná. Poslední možností je rozšíření PHP MySQLi (MySQL Improved), což je vlastně původní rozšíření MySQL doplněné o nové funkce tohoto jazyka. Navíc umožňuje jak objektově orientovaný, tak procedurální přístup k databázi a v podstatě se mnoho neliší od původního rozšíření PHP MySQL.

Abychom mohli s naší databází pracovat, musíme se k ní nejprve připojit. Toho docílíme pomocí funkce *mysqli\_connect()*, který má čtyři parametry. Tím prvním je adresa serveru, kde databáze běží a následuje uživatelské jméno a heslo k připojení do databáze a jméno samotné databáze. Celou funkci je ještě vhodné vložit do proměnné, protože pomocí ní budeme dále k databázi přistupovat. Tato proměnná se nyní bude vkládat do prvního parametru všech příkazů souvisejících s naší databází.



Ukázka : Připojení k databázi

Dále musíme stanovit sadu znaků pro přístup k databázi. Toho dosáhneme pomocí funkce *mysqli\_set\_charset()*, jejímž prvním parametrem je právě funkce *mysqli\_connect()*, kterou jsme si již dříve uložili do proměnné, a druhým parametrem pak je samotná sada znaků. Zápis sady znaků se zde ale řídí podle jazyka SQL, proto musíme sadu *windows-1250* změnit na *cp1250*, sadu *iso-8859-2* na *latin2* a *utf-8* na *utf8*.



Ukázka : Nastavení sady znaků pro práci s databází

Teď již můžeme začít pracovat s databází. Nejprve sestavíme příkaz SQL obsahující data, která chceme do databáze zapsat, a poté tento příkaz zpracujeme pomocí funkce *mysqli\_query()*. Příkaz SQL pro zapsání dat do tabulky databáze je *insert into*, za nímž následuje název tabulky. Následuje příkaz *set*, za nímž následují zapisovaná jednotlivá data. Ta se vkládají ve formátu *Sloupec=‘Hodnota‘* a oddělují se čárkou. Celý samotný příkaz jazyka SQL by pak mohl vypadat nějak takto:



Ukázka : Příkaz jazyka SQL pro zápis do databáze

Ještě, než příkaz odešleme, musíme ale veškeré hodnoty zadané uživatelem přepsat do strojově čitelné hodnoty. Musíme tedy znaky, které jsou příkazy jazyka SQL nebo které nejsou zahrnuty v nastavené sadě znaků tzv. *escapovat* (čti „iskejpovat“), k čemuž slouží funkce *mysqli\_real\_escape\_string()*, jejímž prvním parametrem je stále příkaz *mysqli\_connect()* a druhým parametrem je SQL příkaz, který chceme přepsat.



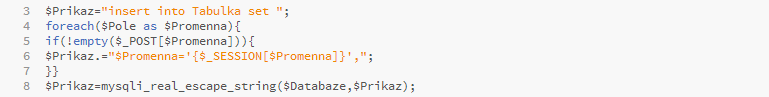
Ukázka : Zpracování SQL příkazu

Takto připravený SQL příkaz pak již můžeme zapsat do funkce *mysqli\_query()*, jejíž parametry jsou totožné s funkcí *mysqli\_real\_escape\_string()*. Na závěr ukončíme spojení s databází pomocí příkazu *mysqli\_close()*. Ani zde nesmíme zapomenout na první parametr.



Ukázka : Ukončení práce s databází

Pokud máme větší množství dat, která chceme zapsat, pak data do příkazu zpracujeme podobně, jako jsme to udělali u *cookies*. Příprava takového příkazu bude poté vypadat zhruba takto:



Ukázka : Příprava příkazu pro zápis většího množství dat do databáze

## Zápis dat do souboru

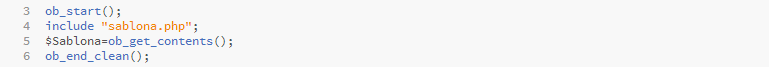
Další možností, jak pracovat se zadanými daty uživatele, může být na serveru zápis do souboru. Můžeme jej využít jak k samotnému uložení dat, tak k vygenerování vyplněného formuláře pro uživatele ke stažení nebo vytištění. Budeme potřebovat funkce *fopen()*, *fwrite()* a *fclose()*, které k danému účelu slouží. Pomocí *fopen()* vytvoříme soubor nebo jej otevřeme, pokud již byl vytvořen, pomocí *fwrite()* do něj zapíšeme data a pomocí *fclose()* ukončíme práci se souborem.

*fopen()* má v závorce dva hlavní parametry. Tím prvním je jméno souboru, případně cesta k němu, tím druhým parametr určující, k jakým operacím budeme chtít soubor použít. Na výběr máme z více možností. Ke čtení souboru používáme nejčastěji parametrů *r* pro otevření stávajícího souboru a nastavení kurzoru na jeho začátek, *w* pro vytvoření nového souboru nebo vymazání obsahu stávajícího a *a* pro vytvoření nového souboru nebo načtení stávajícího, přičemž kurzor se nastaví na konec souboru. Ke každému parametru se ještě přidává *+*, pokud chceme soubor otevřít i k zápisu. K uložení dat je tedy vhodné použít *a+*, k vygenerování unikátního formuláře pro každého uživatele pak stačí *w+* v kombinaci s unikátním jménem souboru (zde můžeme například využít *session id*). Oba dva parametry *fopen()* znovu nesmíme zapomenout obalit uvozovkami a oddělit je čárkami. Funkce *fwrite()* má také dva parametry. Prvním z nich je funkce *fopen()*. Jelikož jsme si ji již předem nadefinovaly, nejjednodušší je si ji uložit do proměnné a tu poté do parametru jenom doplnit. Druhým parametrem bude text, který chceme zapsat. *fclose()* má jenom jeden parametr totožný s prvním parametrem u *fwrite()*.



Ukázka : Jednoduchý zápis do souboru

Pokud chceme do souboru data pouze uložit, pak nám postačí jenom textový vstup obsahující zapisované proměnné a obalený v uvozovkách. Pokud chceme v zapisovaném textu vytvořit nový řádek, použijeme PHP proměnnou *PHP\_EOL*. Pokud však chceme uživateli vygenerovat vyplněný formulář k vytisknutí, pak bude jednodušší nahrát do *fwrite()* předpřipravenou šablonu uloženou v externím souboru. Abychom zpracovali proměnné, vkládané do této šablony, a zároveň nahráli obsah souboru do *fwrite()*, musíme použít funkci *include()*, jejímž parametrem je jméno souboru šablony, případně cesta k němu. Funkci *include()* jako takovou však nemůžeme nahrát do proměnné, proto použijeme tzv. *output buffering*. Pomocí této funkce jazyka PHP vytvoříme dočasně druhý paralelní běh PHP kódu uvnitř souboru, jakousi vyrovnávací paměť, která je nezávislá na zbytku kódu, ale dovede s okolím pracovat. Vytvoříme si tak jakousi „oddělenou místnost“. V ní nahrajeme obsah šablony pomocí *include()*, čímž jej automaticky zpracujeme (proměnné se nahradí jejich hodnotami) a poté celý obsah naší oddělené místnosti uložíme do proměnné.



Ukázka : Nahrání obsahu šablony do proměnné pomocí *output bufferingu*

Oddělenou místnost uvedeme pomocí konstruktoru *ob\_start()*. Následující kód až do uvedení konstruktoru *ob\_end\_\*()* nebo do konce samotného souboru už bude uvnitř oddělené místnosti. Nyní tedy můžeme doplnit funkci *include()*. Pomocí konstruktoru *ob\_get\_contents()* nahrajeme obsah oddělené místnosti, čili obsah vložené a zpracované šablony do proměnné a pomocí konstruktoru *ob\_end\_clean()* ukončíme oddělenou místnost a vymažeme dočasná data v ní. Proměnnou pak už můžeme vložit do příslušného parametru funkce *fwrite()*, případně s ní dále pracovat.

## Odeslání potvrzujícího emailu

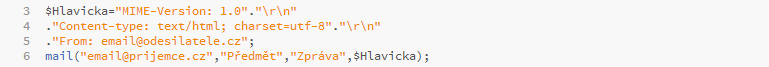
Obdobně připravíme a odešleme potvrzovací email. Znovu můžeme použít *output buffering* společně s předpřipravenou šablonou, pokud chceme vytvořit složitější email pomocí HTML a Kaskádových stylů. Při přípravě šablony emailu ale máme omezenější možnosti. Email by měl být napsán v jednoduché formě HTML s použitím co nejjednodušších stylů a nejlépe bez použití skriptovacích jazyků. Veškeré vložené soubory, jako například obrázky nebo videa, by měly být vloženy do emailu přímo, například pomocí konstrukce *data*. Pro naše účely však postačí i jednoduchý email bez obrázků.

Podstatnější limitací pro nás však bude, že emailová zpráva nesmí obsahovat více než 70 znaků na řádek, aby byla správně odeslána a přečtena. Užitečnou funkcí, díky které nebudeme muset při přípravě šablony emailu myslet na délku řádku, je funkce *wordwrap()*, která vždy po zadaném počtu znaků vloží zadaný oddělovač, například nový řádek. Do této funkce bychom měli zadat tři parametry. Prvním je zpráva, kterou chceme zpracovat, druhým je počet znaků a třetím je znak, který se vždy po zadaném počtu znaků vkládá do zprávy. Emailová zpráva by měla vždy obsahovat kompletní tzv. *CRLF* deklaraci, nemůžeme proto použít proměnnou *PHP\_EOL*. V našem případě by tedy mohla funkce *wordwrap()* vypadat nějak takto:



Ukázka : Zpracování emailové zprávy

Samotné odeslání emailu pak provedeme pomocí funkce *mail()*. Jejími čtyřmi nejčastěji používanými parametry jsou postupně email příjemce, předmět, samotná zpráva a hlavička. Do hlavičky bychom měli zapsat *MIME* deklaraci, použitou sadu znaků (viz kapitola 3.1 Základní struktura formuláře) a odesílatele zprávy a oddělit jednotlivé položky hlavičky odřádkováním *CRLF*. Celá funkce *mail()* by pak mohla vypadat zhruba takto:



Ukázka : Odeslání potvrzovacího emailu

## Získání zapsaných dat

Pokud uživatel odešle, uloží nebo jinak opustí formulář (například neúmyslně chybou serveru nebo klienta), měl by mu náš formulář umožnit se k těmto datům vrátit, získat je zpět, obnovit je. Ukázali jsme si ukládání do *sessions*, *cookies*, databáze nebo souboru. Jak ale tato data získat zpět?

Nejjednodušší je získat data ze *session*. Jestliže ji uživatel má spuštěnou a nevypnul v mezičase prohlížeč, pak ověříme, zda jsou data v *session* ještě dostupná a nahrajeme je zpět. Pokud tedy uživatel omylem přeruší vyplňování formuláře, ale nevypnul ještě prohlížeč, můžeme data vrátit zpět do formuláře pomocí podobného způsobu, jakým jsme je četli. Do hodnoty *value* u prvku *<input>* stačí vložit samotnou *session* proměnnou pomocí funkce *echo()*. Abychom zamezili tomu, že se nám do prvku formuláře vepíše chyba, zjistíme pomocí podmínky *if*, zda daná proměnná existuje nebo zda není prázdná. První možnost obstarává funkce *isset()*, druhou funkce *empty()*. Jejich společným jediným parametrem je proměnná, kterou chceme vyšetřit a použití té, či oné funkce je čistě na našem rozhodnutí. Poté vypíšeme buď hodnotu proměnné, nebo prázdnou hodnotu (prázdnými uvozovkami *““*). Pro takto jednoduchou podmínku stačí použít podmínkové operátory *?:* jejichž parametry vyjadřuje nejlépe toto schéma: *podmínka?platí:neplatí*. Zápis takovéto zkrácené podmínky a vypsání různých hodnot pro různé případy by v našem případě vypadalo zhruba takto:



Ukázka : Podmínka pro vypsání hodnoty uložené v *session*, pokud existuje

Podobnou podmínkou se dají vložit i data z *cookies*. Jediným rozdílem vlastně je, že místo *$\_SESSION[“Jmeno“]* použijeme *$\_COOKIE[“Jmeno“]*. Pro přepsání dat z *cookies* zpět do nové *session* můžeme použít stejného *foreach* cyklu a stejného pole, jakým jsme dříve hodnoty zapisovali do *cookies*, ovšem nyní místo funkce *setcookie()* použijeme znovu příkaz *$\_COOKIE[“Jmeno“]*.

Složitější už ale bude získání dat z databáze nebo souboru. Pro potřeby prohlédnutí si zapsaných uživatelů sběratelem formulářů stačí, pokud si data v databázi prohlédneme pomocí aplikace *phpMyAdmin*, nebo pokud si soubor s daty otevřeme. Pokud ale potřebujeme data vložit uživateli zpět do formuláře, pokud například uživatel potřebuje data po určité době aktualizovat, musíme data načíst složitými funkcemi a konstrukcemi. Každopádně, tak jako v předchozích případech, to bude velmi podobné předchozímu zápisu dat.

Postup pro získání dat z databáze bude velmi podobný tomu, jakým jsme data do databáze nahráli. Znovu bude potřeba se nejprve k databázi připojit, nastavit sadu znaků, znovu se odešle připravený příkaz a ukončí spojení. Jediný rozdíl bude spočívat v tom, že samotný příkaz bude vypadat trochu jinak. Příkaz pro vybrání určité hodnoty z tabulky je *select*. Za ním následuje výčet čárkami oddělených sloupců, ze kterých chceme získat data. Chceme-li získat data ze všech sloupců, pak výčet sloupců stačí nahradit hvězdičkou. Tu však pro naše účely nebudeme potřebovat. Dalším příkazem *from* vybereme příslušnou tabulku, ze které chceme získat data. Poslední příkaze *where Sloupec=‘Hodnota‘* nám pak vybere pouze ty řádky, na kterých se ve *Sloupci* vyskytuje zadaná *Hodnota*. S příkazem *where* v kombinaci s různými dalšími příkazy můžeme vybrat více různých řádků, které odpovídají různým zadaným kritériím, pro naše účely ale zase postačí tato jednoduchá forma. Celý samotný SQL příkaz pak bude vypadat takto:



Ukázka : SQL příkaz pro získání určité hodnoty z databáze

Data ze souboru získáme a nahrajeme do proměnné podobným způsobem, jako u šablony. Můžeme použít původní *output buffering*, ale protože nyní již nepotřebujeme zpracovat žádné proměnné, tak jako tomu bylo u šablony, postačí nám funkce *file\_get\_contents()*, jejíž parametr je stejný jako u funkce *include()*. Pomocí této funkce vložíme obsah souboru přímo do proměnné a budeme moci se souborem dále pracovat. K dalšímu zpracování poté použijeme mnoho užitečných *string* funkcí. Poslouží nám například *explode()*, *strtok()*, *parse\_str()*, *similar\_text()*, *str\_split()*, *strip\_tags()*, *strpos()*, *strstr()*, *nl2br()*, *wordwrap()*, *preg\_match()*, *substr()* a mnoho dalších.

## Vícenásobné vkládání kódu

Jazyk PHP ale nemusíme využít jenom k práci s daty ve formuláři. Můžeme si jím usnadnit například přípravu formuláře samotného. K tomuto účelu využijeme funkcí, které jsme si již představili dříve. Využijeme polí, *if* podmínek i *foreach* cyklů.

Pro příklad můžeme chtít vytvořit větší množství textových formulářových polí. Do každého pole budeme chtít ještě automaticky doplnit hodnotu, pokud již je uložena v *session*. A ke každému poli budeme chtít doplnit navíc popisek. V samotném HTML by jedno takové pole vypadalo takto:



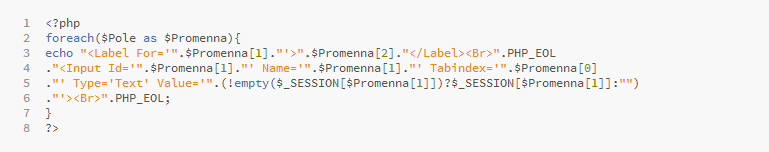
Ukázka : Kompletní formulářové textové pole v HTML

Takových polí pak ve formuláři může být nespočet. Vypisovat všechna taková pole by bylo neefektivní a náročné. PHP nám však v tomto procesu umí dobře pomoct. Začněme tedy tím, že si vytvoříme pole, do kterého vypíšeme všechny hodnoty, které se nám budou měnit. Pokud je takových hodnot více než jedna, pak můžeme do polí místo hodnot samotných vepsat další vnořená pole. V našem případě se mění tři proměnné: *id*, *tabindex* a *name*, případně další proměnné mající stejnou hodnotu jako *name*. Do každého vnořeného pole tedy vložíme tři různé hodnoty. Celé pole pak vypadá zhruba takto:



Ukázka : Pole s vnořenými poli a proměnnými

Další postup již důvěrně známe. Pomocí *foreach* cyklu a doplňkových *if* podmínek tedy vygenerujeme celý formulář a dosadíme do něj jednotlivé hodnoty. Určitou hodnotu z pole pak můžeme získat pomocí hranatých závorek, do nichž se zapíše číslo nebo název příslušné hodnoty. Například první hodnotu v poli získáme pomocí příkazu *$Pole[0]*. Nesmíme zde zapomenout na fakt, že číslování v programovacích jazycích začíná vždy od nuly, nikoliv od jedničky. Pokud chceme z ukázky 30 vybrat z vnořeného pole hodnotu *Příjmení*, pak náš příkaz napíšeme jako *$Pole[1][2]*. S pomocí *foreach* cyklu pak můžeme celý proces zautomatizovat a vypsat tolik proměnných, kolik vnořených polí s hodnotami máme. Získání hodnot *Jméno*, *Příjmení* a *Titul* uvnitř cyklu tedy provede příkaz *$Promenna[2]*. Celý cyklus by pro tento případ mohl vypadat třeba takto:



Ukázka : *foreach* cyklus pro vypsání většího množství formulářových prvků

## Detekce chyb a chybová stránka

# Závěr

# Seznam zdrojů

1. **Vaclav.Makes.** Wikipedia.org. *PHP.* [Online] 22. 2 2014. [Citace: 2. 3 2014.] http://cs.wikipedia.org/w/index.php?title=PHP&oldid=11248145.

2. **Zajíc, Petr.** Linuxsoft.cz. *MySQL (1) - pestrý svět databází.* [Online] 1. 3 2005. [Citace: 2. 3 2014.] http://www.linuxsoft.cz/article.php?id\_article=731.

3. **PastoriBot.** Wikipedie. *Apache HTTP Server.* [Online] 10. 2 2014. [Citace: 2. 3 2014.] http://cs.wikipedia.org/w/index.php?title=Apache\_HTTP\_Server&oldid=11200224.

4. **Chamurappi.** Webylon. *Fontokuk.* [Online] 2014. [Citace: 2. 3 2014.] http://fontokuk.webylon.info/.

5. **Moravec, Zdeněk.** Programujte.com. *PHP a MySQL – MySQLi – 1. díl.* [Online] 14. 1 2010. [Citace: 22. 3 2014.] http://programujte.com/clanek/2009103100-php-a-mysql-mysqli-1-dil/.

6. **SEVT, a.s.** *Přihláška ke studiu na vysoké škole v bakalářském nebo magisterském studijním programu.* Praha : SEVT, 2007.

# Seznam obrázků

[Obrázek 1: Jednoduchý formulář bez nastavených stylů 16](#_Toc383349298)

[Obrázek 2: Jednoduchý formulář s nastavenými styly 18](#_Toc383349299)

[Obrázek 3: Kulatá tlačítka 20](#_Toc383349300)

# Seznam ukázek kódu

[Ukázka 1: Základní struktura HTML stránky 12](#_Toc383349301)

[Ukázka 2: Hlavička stránky 13](#_Toc383349302)

[Ukázka 3: Tělo stránky 14](#_Toc383349303)

[Ukázka 4: Jednotlivé prvky formuláře 15](#_Toc383349304)

[Ukázka 5: Styly stránky a formuláře 16](#_Toc383349305)

[Ukázka 6: Absolutně vycentrovaný prvek 18](#_Toc383349306)

[Ukázka 7: Různé styly formulářových prvků 18](#_Toc383349307)

[Ukázka 8:Kulatá tlačítka 20](#_Toc383349308)

[Ukázka 9: Vložení písma do souboru 20](#_Toc383349309)

[Ukázka 10: Formule pro vložení písma do stránky 21](#_Toc383349310)

[Ukázka 11: Prefixované zaoblené rohy u tlačítka 21](#_Toc383349311)

[Ukázka 12: vložení PHP kódu do HTML dokumentu 22](#_Toc383349312)

[Ukázka 13: Doplnění *session id* do adresy url 23](#_Toc383349313)

[Ukázka 14: Zpracování velkého množství dat pomocí pole a foreach cyklu 23](#_Toc383349314)

[Ukázka 15: Uložení jednoduché hodnoty do *cookie* na dobu jednoho dne 24](#_Toc383349315)

[Ukázka 16: Zápis většího množství dat hodnot do *cookies* 24](#_Toc383349316)

[Ukázka 17: Připojení k databázi 25](#_Toc383349317)

[Ukázka 18: Nastavení sady znaků pro práci s databází 25](#_Toc383349318)

[Ukázka 19: Příkaz jazyka SQL pro zápis do databáze 26](#_Toc383349319)

[Ukázka 20: Zpracování SQL příkazu 26](#_Toc383349320)

[Ukázka 21: Ukončení práce s databází 26](#_Toc383349321)

[Ukázka 22: Příprava příkazu pro zápis většího množství dat do databáze 26](#_Toc383349322)

[Ukázka 23: Jednoduchý zápis do souboru 27](#_Toc383349323)

[Ukázka 24: Nahrání obsahu šablony do proměnné pomocí *output bufferingu* 28](#_Toc383349324)

[Ukázka 25: Zpracování emailové zprávy 29](#_Toc383349325)

[Ukázka 26: Odeslání potvrzovacího emailu 29](#_Toc383349326)

[Ukázka 27: Podmínka pro vypsání hodnoty uložené v *session*, pokud existuje 30](#_Toc383349327)

[Ukázka 28: SQL příkaz pro získání určité hodnoty z databáze 30](#_Toc383349328)

[Ukázka 29: Kompletní formulářové textové pole v HTML 31](#_Toc383349329)

[Ukázka 30: Pole s vnořenými poli a proměnnými 31](#_Toc383349330)

[Ukázka 31: *foreach* cyklus pro vypsání většího množství formulářových prvků 32](#_Toc383349331)

# Přílohy

Příloha 1

Příloha 2

1. Zpětná kompatibilita je terminus technikus pro stav, kdy je daný technický produkt (například kód nebo software) technicky plnohodnotně zaměnitelný se svojí starší verzí. V programátorské praxi se tak jedná o stav, kdy napsaný kód nezestárne a není potřeba jej po určité době aktualizovat, doplnit nebo úplně vyměnit [↑](#footnote-ref-1)
2. Polyfill je úsek kódu nebo samostatná knihovna vložená do stránky z důvodu zajištění funkčnosti dané funkce napříč klienty. Používá se často k záplatování chyb, doplnění nadstandardních funkcí nebo kvůli zpětné kompatibilitě. Mnoho polyfillů je však naprogramováno v JavaScriptu, proto na ně nelze příliš spoléhat (viz kapitola 2.1.3 JavaScript). [↑](#footnote-ref-2)
3. DOM (zkr. Document Object Model, česky Objektový model dokumentu) je rozhraní umožňující skriptovacím jazykům přístup ke struktuře stránky, k jejím jednotlivým prvkům a případně také k jejich důležitým vlastnostem [↑](#footnote-ref-3)