FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestrální práce z předmětu

Webové aplikace

Online obchod T-space

20. prosince 2023

Autor:

Adam Míka
A22B0319P
mikaa@students.zcu.cz

Cvičící:
Ing. Michal Nykl Ph.D.
nyklm@ntis.zcu.cz

Obsah

1 Zadání

1.1 Nutné požadavky na všechny samostatné práce

1. Technologie - povinně HTML5, CSS, PHP a SQL (MySQL nebo jiná databáze), volitelně šablony, JavaScript, AJAX, Bootstrap apod.
2. Aplikace musí dodržovat MVC architekturu a využívat OOP (min. controllery a model).
3. Web má jeden vstupní soubor (obvykle index.php), který na základě parametrů URL adresy provede požadovanou akci (tj. zavolá příslušný controller) a vypíše výstup uživateli.
4. Pro práci s databází musí být využito PDO nebo jeho ekvivalent.
5. Web musí být chráněn proti útokům typu XSS a SQL Injection.
6. V databázi musí být hesla hashována.
7. Web musí využívat upload souborů.
8. Web musí mít responzivní design (alespoň pro PC a mobil).
9. Web musí mít alespoň 3 uživatelské role (po přihlášení v systému provádí příslušné činnosti, např. autor, recenzent, admin).
10. K aplikaci musí být dodána dokumentace (viz dále) a skripty pro instalaci databáze (např. získané exportem databáze).
11. Práce musí být osobně předvedena cvičícímu a po schválení odevzdána na CourseWare či Portál.
12. Aplikaci není možné realizovat s využitím ucelených PHP frameworků (zakázáno např. Nette, Symfony atd.). Použití jejich komponent je možné pouze po schválení vyučujícím.
13. Pro front-end je vhodné využít framework Bootstrap (getbootstrap.com) nebo jeho ekvivalent.

1.2 Dokumentace

1. Vaše jméno, email, datum vytvoření, název předmětu a název aplikace/tématu.
2. URL vytvořených stránek (pokud jsou zveřejněny na serveru students.kiv.zcu.cz či jinde)
3. Popis použitých technologií - uveďte hlavně, ve které části jste kterou technologii použili.

4. Popis adresářové struktury aplikace - co je ve kterých adresářích a souborech.
5. Popis architektury aplikace - co mají na starosti které třídy (popř. lze využít i UML diagramy).
6. Seznam defaultních uživatelů včetně loginů a hesel.
7. U alternativního zadání uveďte celé, cvičícím schválené zadání práce.
8. Dokumentaci netiskněte, ale odevzdejte ji ve formátu PDF spolu s aplikací.

1.3 Hodnocení samostatné práce a získání zápočtu

1. Hodnocení práce je rozděleno na povinné požadavky a volitelná rozšíření, přičemž detailní podmínky sdělí a vysvětlí každý vyučující na prvním cvičení.
2. Pro cvičení, která vede M.Nykl, je hodnocení uvedeno v záložce menu Hodnocení SP (Nykl).
3. Pro získání zápočtu je nezbytné splnit nutné požadavky a získat ze semestrální práce minimálně 20 bodů (max. 60).

Dodatečné poznámky k hodnocení:

1. Zákaz plagiátorství - práci musí student vytvořit samostatně. Nelze ji tedy zkopírovat např. z Githubu a ani nelze "udat"starou práci ze střední školy (obvykle nesplňuje nutné požadavky). Plagiátorství je odhalováno automatizovaným systémem a hodnoceno odebráním zápočtu.
2. Odevzdání práce v průběhu cvičení v ZS (tzv. do Vánoc) je ohodnoceno bonusovými body.
3. Bootstrap či ekvivalent - obvykle umožňují snadno zajistit responzivitu a pěkný design (dohromady hodně bodů).
4. Github, Bitbucket - práce je uložena v repozitáři (snadné body; výborná záloha historie kódu).

1.4 Předvedení a odevzdání SP

1. Nutnou součástí odevzdání je osobní předvedení práce vyučujícímu, které je možné pouze na cvičeních nebo ve stanovených termínech (cca 3 za zkouškové období).
2. Vytvořené webové stránky při předvádění cvičícímu poběží na počítači, ke kterému bude mít cvičící fyzický přístup (např. počítač v učebně nebo notebook, který máte s sebou). Webovým serverem bude buď localhost (127.0.0.1), nebo univerzitní server students.kiv.zcu.cz.

3. Akceptovaná aplikace (tj. zdrojové kódy včetně dokumentace ve formátu PDF) bude odevzdána do odevzdávacího portletu na Portal či Courseware. Archiv bude pojmenovaný PRIJMENI-JMENO.ZIP a v nejvyšší úrovni adresářové struktury bude soubor readme.txt s popisem obsahu jednotlivých adresářů. Práci ale odevzdejte až po jejím akceptování vyučujícím.

2 Popis použitých technologií

2.1 JavaScript

JavaScript je skriptovací programovací jazyk, který umožňuje interaktivitu na stránkách. Používá se pro manipulaci s obsahem HTML stránek, zpracování událostí a komunikaci s webovým serverem.

```
function limitT(a) {
    if (a.length > 32) return a.substring(0, 90)+"...";
}
$("#content p").text(function() {
    return limitT(this.innerHTML)
});
```

JavaScript je použit pro klientovou validaci vstupních formulářů, interaktivní prvky na stránce a reakci na uživatelské události. Například pro omezení počtu znaků v popisu produktu.

2.2 jQuery

jQuery je rychlá, malá a bohatá knihovna pro JavaScript, která zjednodušuje manipulaci s dokumentem HTML, obsluhu událostí, animace a práci s AJAXem.

jQuery byl použit pro usnadnění a zjednodušení některých operací v JavaScriptu, jako například selekce elementů, manipulace s DOM a práce s AJAXem.

2.3 MVC Architektura

MVC (Model-View-Controller) je architektonický vzor, který rozděluje webovou aplikaci do tří základních komponent - modelu, pohledu a řadiče. To pomáhá oddělit logiku aplikace od prezentační vrstvy a usnadňuje údržbu a rozšiřitelnost.

MVC architektura byla použita pro organizaci a strukturování kódu semestrální práce. Třída WhiteTeaController slouží jako řadič, který ovládá logiku pro stránku O nás, MyDatabase zastává roli modelu pro práci s databází a pohled je oddělen do samostatného souboru whitetea.php.

3 Popis adresářové struktury aplikace

1. assets: Obsahuje statické soubory, jako CSS styly (style.css) a JavaScriptové skripty (script.js), které jsou používány ve webové aplikaci.
2. controllers: Obsahuje kontroléry aplikace. IController.interface.php může být rozhraním, které implementují všechny kontroléry, a whiteTeaController.php je konkrétní implementací kontroléru pro stránku O nás.
3. models: Obsahuje modely aplikace. MyDatabase.class.php může obsahovat třídu pro práci s databází a userManage.php může zahrnovat třídu pro správu uživatelů.
4. views: Obsahuje šablony pohledů aplikace. whitetea.php může obsahovat HTML a PHP kód pro zobrazení stránky O nás.
5. index.php: Hlavní vstupní bod webové aplikace. Zde může být definována logika pro směrování a spouštění aplikace.

4 Popis architektury aplikace

WhiteTeaController

1. Zajišťuje inicializaci a správu objektů pro práci s databází a uživateli.
2. Obsahuje metodu show, která zpracovává požadavky na zobrazení stránky O nás, manipuluje s daty a výsledek vrací ve formě řetězce HTML.

MyDatabase

1. getTea: Získává data o čaji podle zadané kategorie.
2. addToCart: Přidává produkt do košíku uživatele.
3. decrease: Sníží množství produktu v databázi.
4. updateQuantity: Aktualizuje množství produktu v databázi.

UserManage

1. userLogout: Odhlašuje uživatele.
2. isUserLogged: Kontroluje, zda je uživatel přihlášen.
3. getLoggedUserData: Získává data o přihlášeném uživateli.

Popis vzájemných vztahů:

1. WhiteTeaController má vlastní instance tříd MyDatabase a UserManage, které využívá pro práci s databází a správu uživatelů.
2. MyDatabase a UserManage jsou samostatné modelové třídy pro práci s databází a uživatelskými účty.

3. WhiteTeaController komunikuje s MyDatabase a UserManage prostřednictvím jejich veřejných metod.
4. WhiteTeaController je součástí architektury MVC, kde plní roli řadiče (Controller), MyDatabase je model pro práci s daty, a UserManage je další model pro správu uživatelů.

5 Seznam defaultních uživatelů

Administrátor

1. Login: admiam
2. Heslo: adam

Admin

1. Login: rychlikj
2. Heslo: heslo12345

Skladník

1. Login: jensísek
2. Heslo: heslo12345

Zákazník

1. Login: martinjekokos
2. Heslo: MaritinMaritin

6 Závěr

Veškerá architektura, adresářová struktura a implementace tříd webové aplikace byla navržena s ohledem na efektivitu, modularitu a snadnou údržbu. Aplikace využívá technologie jako JavaScript, jQuery a MVC architekturu k dosažení co nejlepší uživatelské zkušenosti a jednoduché správy kódu.

Adresářová struktura byla pečlivě organizována, s oddělením statických souborů, kontrolérů, modelů, pohledů a konfiguračních souborů. Tím je dosaženo čistého oddělení zodpovědností a zvyšuje se přehlednost projektu.

MVC architektura, kterou aplikace následuje, zajišťuje, že logika, data a prezentační vrstva jsou odděleny, což usnadňuje úpravy, rozšíření a testování jednotlivých částí aplikace.

Celkově lze konstatovat, že navržená webová aplikace splňuje požadavky na strukturu, organizaci a efektivitu. Využívání moderních technologií a návrhových vzorů

přispívá k optimálnímu vývoji a udržení kvalitního kódu. Projekt je připraven k dalšímu rozvoji, ať už se jedná o přidání nových funkcí, optimalizaci či rozšíření.