ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

KIV/PIA

E-shop aplikace

Autor: Antonín NEUMANN, A14N0139P

Akademický rok: 2014/2015

1 Zadání

Zadání převzato z Courseware stránek předmětu.

Princip a základní funkce elektronického internetového obchodu (e-shopu) patří dnes již k dobře známým a běžným vědomostem díky neustále se zvětšujícímu počtu instancí těchto obchodů napříč nejrůznějšími druhy zboží a služeb i objemu odchodů touto cestou uzavíraných. V semestrální práci si proto letos můžete zvolit jakýkoli vašemu srdci blízký sortiment a vybudovat internetovou aplikaci sloužící právě jako e-shop vaše zboží nabízející.

Při návrhu a implementaci přemýšlejte nad uživatelskou přívětivostí a vhodným ošetřením chybových vstupů.

Práci můžete libovolně rozšířit o další funkčnost, maximální počet bodů, který můžete ze semestrální práce získat, je 15 bodů za prototyp a 35 bodů za aplikaci.

1.1 Uživatelské role a povinné funkce

- nepřihlášený uživatel
  - může se registrovat
  - může se přihlásit
  - může prohlížet seznam zboží
  - může zobrazit detail výrobku
  - může přidávat věci do košíku (počet kusů, velikost, atd.)
  - může zobrazit obsah košíku
  - může objednat zboží z košíku (zadat způsob přepravy, způsob platby, dodací adresu, atd.)
- přihlášený uživatel (krom možností nepřihlášeného uživatele)
  - může editovat svoje údaje
  - může se odhlásit
- administrátor (kromě možností běžného přihlášeného a nepřihlášeného uživatele)
  - může přidávat, upravovat a mazat zboží v nabízeném sortimentu
  - může sledovat provedené objednávky všech uživatelů

1.2 Požadavky na registrační formulář

- některé požadavky jsou pouze z pedagogických důvodů, abychom zkontrolovali, jak jste si poradili s ověřením uživatelských vstupů, např. s ošetřením HTML vstupů nebo vložením nesmyslných údajů
- použít checkbox (např. souhlas s podmínkami) nebo radio buttony (např.: volba pohlaví)
- žádný uživatelský vstup nesmí vyvolat neošetřenou výjimku (shodit aplikaci, chyba 500)
- jako nepovinný údaj bude datum narození, ze kterého se v profilu vypočítá věk
- používání bude pohodlné, tj. při nevhodných datech bude uživateli zobrazena chybová zpráva a vstupní pole se vyplní zadanými (chybnými) údaji
- ochrana proti automatické registraci roboty, postačí co nejjednodušší řešení, čili nemusíte používat CAPTCHA, ale stačí reverzní Turingův test formou např.:
  - Napište číslici čtyři
  - Kolikátý je nyní měsíc?

1.3 Požadavky na formulář přidávání zboží

- možnost nahrát obrázek

1.4 Požadavky na práci

- validní HTML 4.01 Strict + validní CSS 2.1 nebo XHTML 1.0 Strict + validní CSS 2.1 nebo validní (X)HTML5 + validní CSS 3
- pokud použijete HTML5, nezapomeňte využít sémantických elementů, které přináší (header, footer, apod...)
- v případě starších verzí standardů je zcela validní přístup s označením daných sekcí pomocí elementů div a patřičného id;
- ze cvičných důvodů musí mít alespoň jedna ze stránek webu dvousloupcový (nebo třísloupcový) layout
- ve všech přidávacích a editačních formulářích lze používat tabelátor, je jasné jaká data vkládat do jakého políčka, vhodné políčko má focus
- alespoň 3 neprázdné kategorie zboží (např.: notebooky, monitory, příslušenství / trika, kalhoty, boty / kytary, komba, příslušenství / apod.)
- sortiment alespoň o 10 kusech zboží, alespoň 3 s fotografií

1.5 Požadavek na uživatelská jména

Uživatelské jméno	**Heslo**
user1	User@1234
user2	User@1234
admin	Admin@1234

Uživatelů může být více, ale user1, user2 a admin jsou povinní.

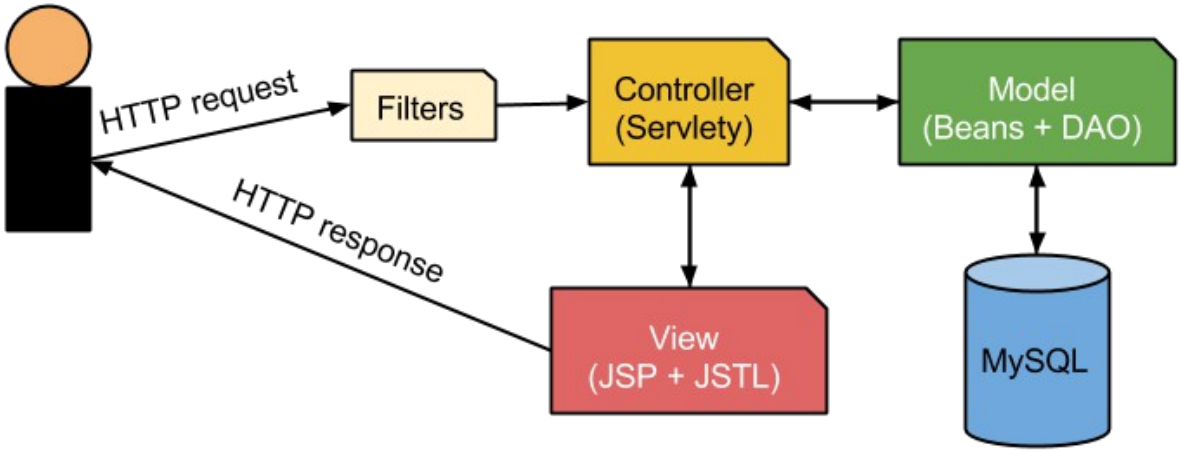
1.6 Ostatní

- veškerá vylepšení či nepříliš zřejmou funkcionalitu popište body v přiloženém README souboru (umí to tohle a tamto a najdete to zde a tamhle). Pokud při opravě práce něco nenajdeme, nebude to zahrnuto do hodnocení (což platí i o případných chybách :o)).
- uživatelské rozhraní, komentáře, dokumentace a kód může být v českém, anglickém nebo slovenském jazyce
- přesto doporučujeme psát alespoň kód a komentáře v anglickém jazyce, neb je to pro praxi dosti užitečná schopnost
- kvalita angličtiny není předmětem hodnocení, tak se toho nebojte :)
- bez ohledu na jazyk UI, aplikace bude testována na vstup českých znaků
- k získání bodů za přenositelnost musíte podporovat další 3 různé prohlížeče, každý s jiným renderovacím jádrem, z toho právě 1 textový + se musí validně zobrazovat na mobilním zařízení (smartphone, tablet, stačí otestovat na jednom OS a nativním prohlížeči).
    - např.: Opera verze < 15(Presto), Chrome (WebKit), Safari (WebKit), Links, Lynx, W3m, ...
- pokud použijete CSS3, bude při validaci nastaveno: „Vendor Extensions: Warnings“
- JavaScript není povinný
- očekává se, že po přihlášení se schovají položky z menu potřebné pro přihlášení a registraci

2 Architektura

Celá moje aplikace je postavená na návrhovém vzoru MVC (Model-View-Controller), jedná se třívrstvou architekturu oddělující od sebe přístup k datům, „bussines“ logiku a vizualizaci.

Jak vypadá implementace MVC vzoru v mé aplikace je možné vidět na obrázku 1.



Obrázek 1: MVC model

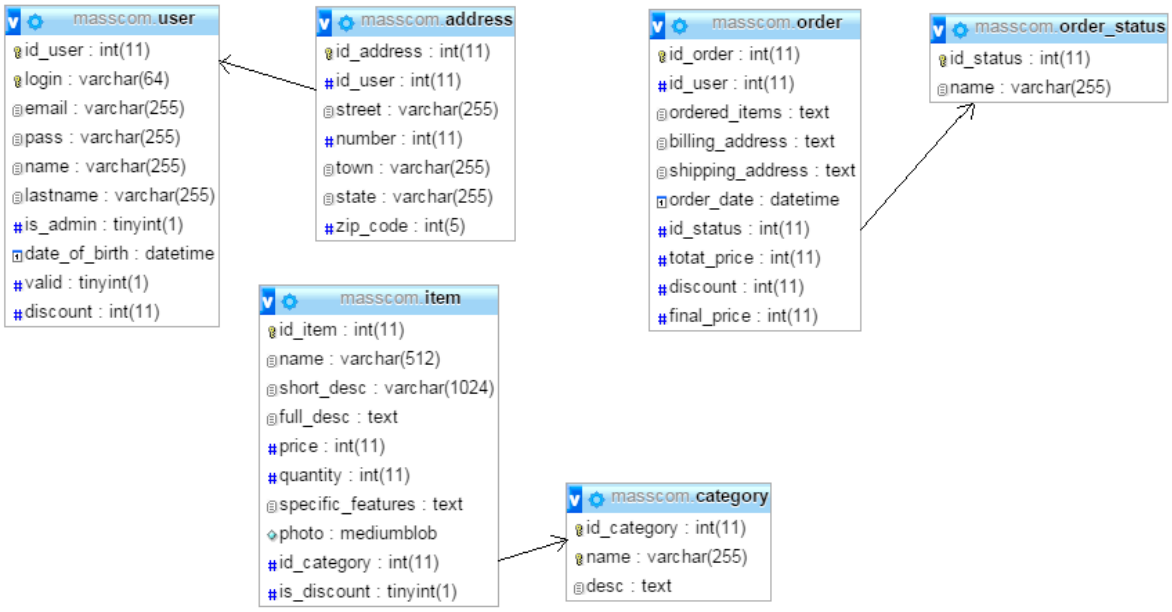
2.1 Model

Model je část aplikace starající se o přístup k datům. V mém případě jsou data uložena v MySQL databázi, schéma tabulek viz obrázek 2.

Pokud tedy controller (servlet) potřebuje pro svojí potřebu nebo pro potřebu vykreslení, tedy vrstvu view, požádá o data modelové třídy DAO (Data Access Object). Tyto třídy mu data předají ve formě objektů tříd Beans.

V celé aplikaci pouze třídy s přívlastkem DAO mají přístup k úložisti dat. Pokud by tedy došlo ke změně typu úložiště ze současného MySQL na jiný, stačí implementovat nové třídy DAO pro práci s tímto úložištěm a celá aplikace bude fungovat stále stejně.

Třídy modelu jsou v aplikaci umístěny v balíčcích `java.dao` a `java.beans`.



Obrázek 2: ERA model databáze

2.2 View

Tato vrstva zobrazuje data uživateli ve formě HTML stránek. Jednotlivé HTML stránky, které uživatel vidí jsou složené z několika JSP souborů. Pro práci s daty a řídícími strukturami je využíván „jazyk“ JSTL (JavaServer Pages Standard Tag Library).

Všechny typy souborů (JSP, CSS, JS a obrázky) používané pro zobrazování jsou umístěné v adresáři `webapp/`.

2.3 Controller

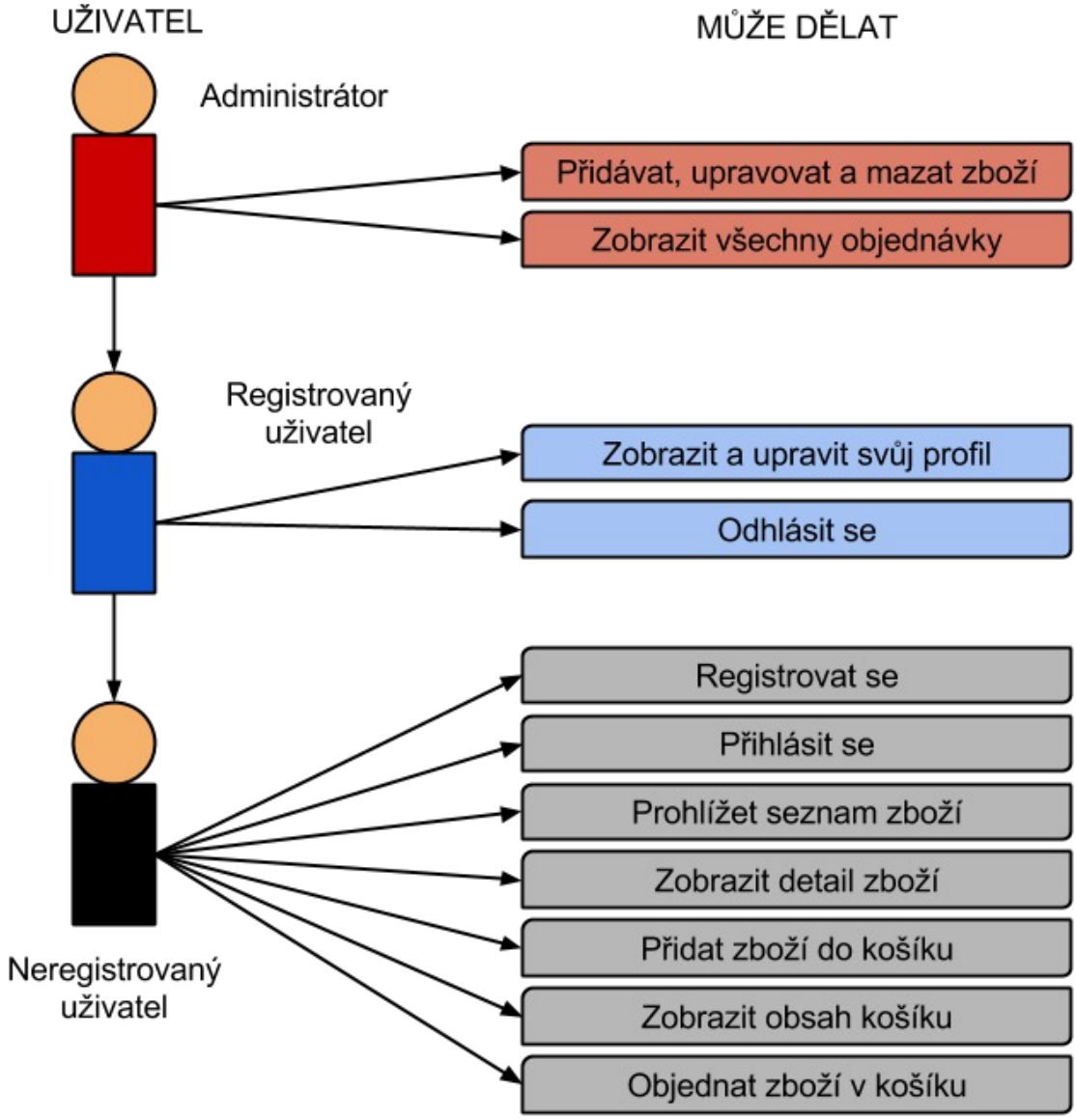
Controller obsahuje třídy starající se o chod a řízení celé aplikace, přijímají HTTP requesty, získávají data, rozhodují které stránky a s jakými daty budou vráceny a zobrazeny uživateli.

Rovněž přijímají data z formulářů, ověřují jejich platnost a případně data předávají k uložení do databáze.

3 Případy užití

Případy užití popisují co mohou jednotlivý uživatelé v aplikaci dělat. Ze zadání vyplývá, že moje aplikace má celkem 3 typy (úrovně) uživatelů – administrátor, registrovaný uživatele a neregistrovaný uživatel.

Každá úroveň může dělat činnosti úrovně nižší. Všechny případy užití jsou dobře patrné z obrázku 3.



Obrázek 3: Případy užití aplikace

4 Implementace

4.1 Připojení k databázi

Pro připojení k databázi a získávání dat slouží, jak již bylo zmíněno, třídy DAO. Každá tato třída dědí od abstraktní třídy *AMysqlDao* a implementuje rozhraní *IGenericDao*.

Třída *AMysqlDao* se stará o připojení k databázi, potřebné údaje jako adresu serveru, port nebo přihlašovací údaje získává ze souboru *db.properties* (v případě potíží s properties souborem existuje v aplikaci ještě náhradní možnost získání dat zapsaných přímo v této třídě – tento postup byl zvolen pouze pro potřeby semestrální práce).

Rozhraní *IGenericDao* popisuje základní metody pro práci s objety tříd beans. Jedná se o metody *create*, *read*, *update* a *delete* často označované pouze jako CRUD.

4.2 Filtry

Filtry jsou velmi podobné klasickým servletům s tím rozdílem, že po příchodu HTTP requestu je tento před předáním příslušnému servletu předán jednomu nebo více filtrům, které jej mohou různě modifikovat.

Ve své aplikaci využívám celkem 5 filtrů:

- *EncodingFilter* sloužící pro nastavení správného kódování.
- *ShoppingCart* který získá ze session seznam zboží uložených v nákupním košíku a předá jej každému servletu.
- *AuthFilter*, *AdminFilter* a *LoginProtectFilter* starající se přihlašování uživatelů a ochranu před neoprávněným přístupem.

4.3 Statické a chybové stránky

Aby nemusel pro každou statickou nebo chybovou stránku vytvářet vlastní servlet, implementoval jsem pouze dva *servlet.Page.java* (pro statické stránky) a *servlet.Error.java* (pro chybové stránky). Každý tento servlet je namapován na více URI (v případě chybových stránek i na více HTTP kódů) a k rozhodování o zobrazení konkrétní stránky dochází až uvnitř servletu na základě zpracování URI pomocí třídy *libs.UriParser.java*.

4.4 Upload obrázků

Obrázky produktů přidané pomocí formuláře jsou v servletu zpracovány pomocí knihoven *apache.commons.fileupload* a *apache.commons.io* a uloženy do databáze jako datový typ MEDIUMBLOB (omezení cca 16MB).

V případě potřeby obrázek z databáze vypsat je potřeba zavolat servlet, který datý daný obrázek s databáze vypíše přímo jako posloupnost bytů.

5 Instalace a nasazení

Celý projekt je vyvinut v Java SDK 7 a projektovém manažeru Apache Maven. Celý projekt byl vyvíjen v IDE Eclipse Luna (4.4.1) a webovém prohlížeči Google Chrome (verze 39.0.2171.95 m).

5.1 Překlad (Compile)

Pro přeložení aplikace slouží v Maven projektech soubor *pom.xml*. Překlad se spustí příkazem **mvn compile**. Přeložené soubory jsou poté dostupné v adresáři *target/*.

5.2 Sestavení (Packaging)

Při sestavení se přeložené zdrojové soubory s dalšími potřebnými soubory zabalí do jediného souboru WAR, konkrétně *neumann.war* umístěného v adresáři *target/*. Příkaz pro packaging je **mvn package**.

5.3 Nasazení (Deploy)

Pro nasazení aplikace na server Apache Tomcat 7, který jsem používal i při vývoji aplikace, je nutné nakopírovat WAR soubor do adresáře *webapp/* a server restartovat.

Výsledná aplikace je poté dostupná na adrese *http://adresa-serveru:port/neumann*, v případě serveru Tomcat se implicitně jedná o adresu *http://localhost:8080/neumann*.

6 Závěr

Práce splňuje povinné zadání, kromě toho jsem navíc přidal vlastní ošetření výjimek, vlastní chybové stránky (400, 401, 403, 404, 405, 500 a 501) s možností snadného rozšíření o další. Další zajímavostí je podle mého implementace zobrazování statických stránek, jako například všeobecné obchodní podmínky, které jsou řešeny pouze namapováním příslušné URI na jeden servlet (`servlet.Page`) a vytvořením JSP souboru téhož názvu ve složce `WEB-INF/page/`.

Práce byla zajímavá a ukázala mi základy tvorby webových stránek v jazyce Java, dosud jsem měl zkušenosti pouze s PHP. Rovněž si myslím, že práce byla až zbytečně těžká (například upload obrázků).

Přestože jsem se aplikaci snažil navrhnout a implementovat svědomitě, některé věci by šlo zcela jistě udělat lépe a celá aplikace by šla dále hojně rozvíjet po vzoru moderních e-shopů.