TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: M 2612 – Elektrotechnika a informatika
Studijní obor: 3902T005 – Automatické řízení a inženýrská informatika

Použití objektových forem ve webových aplikacích

Using Design Patterns in web aplications

Diplomová práce

Autor: **Ondřej Drobník**
Vedoucí práce: Ing. Martin Vlasák
Konzultant: Ing. Igor Kopetschke

V Jablonci nad Nisou 1. 4. 2007

Toto CD je součástí diplomové práce Použití objektových forem ve webových aplikacích. V adresáři Eshop jsou zdrojové kódy internetového obchodu. V adresáři Admin jsou zdrojové kódy správcovské aplikace. Adresář software obsahuje potřebný software pro běh aplikací. Adresář Db obsahuje databázi internetového obchodu.

Postup při instalaci:

1. Nainstalovat webový HTTP server s podporou MySQL (soubor software/wamp5_1.7.1a.exe).
2. Nahrát do adresáře www webového HTTP serveru adresáře Admin a Eshop.
3. Nahrát do php adresáře webového HTTP serveru inicializační soubor software/php.ini.
4. Restartovat server.
5. Nainstalovat aplikaci pro správu databáze (software/SQLyog523.exe)
6. Spustit aplikaci a vytvořit připojení. Nastavit MySQL Host address na localhost, port 3306, user name root.
7. Vytvořit novou databázi s názven dronshop (DB->Create Database).
8. Nainstalovat databázi. V levém okně s batabázemi vybrat dronshop a pravým tlačítkem vyvolat nabídku. Zvolit Restore FROM SQL Dump a nastavit cestu k souboru s databází (db/databaze.sql);

Možné problémy:

1. Špatné kódování při výstupu z ajaxu. Důvod, máte špatně nastavené kódování prohlížeče.
2. Nefunguje ajax: Pravděpodobně máte vypnutou podporu JavaScriptu.
3. Nejde odeslat mail: Prosím nastavte v php.ini správné SMTP a port.
4. Jiné problémy: Pravděpodobně požíváte internetový prohlížeč Internet Explorer, vyzkoušejte Firefox nebo Operu.

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

Poděkování

Rád bych na tomto místě poděkoval Vojtěchu Kurkovi za čas, který mi věnoval ohledně databáze a Ing. Romanovi Špánkovi za pomoc s UML diagramy. Dále bych chtěl poděkovat přítelkyni Petře a dalším za morální podporu.

Anotace

Cílem této diplomové práce je obeznámit čtenáře s problematikou návrhu objektově orientovaných webových aplikací pomocí návrhových vzorů. Návrh aplikací je podložen ukázkovou aplikací internetový obchod a doprovodnou správcovskou aplikací tohoto obchodu. Jedná se o internetový obchod s počítačovými komponentami a elektronikou. Pro aplikaci jsou vhodně zvoleny návrhové vzory odpovídající konstrukci aplikace tohoto druhu a těmi jsou návrhové vzory Singleton, Tovární metoda, Strategie, Prototyp a další. Tato diplomová práce také popisuje vhodné nástroje pro tvorbu internetových objektových aplikací.

Klíčová slova: Návrhové vzory, internetový obchod.

Annotation

The goal of this thesis is to familiarize its readers with the problems of projecting object-oriented web applications using design patterns. The design is verified with the demonstration application, internet shop, and its administrational application. This internet shop is focused on computer components and electronics. Design patterns Singleton, Factory method, Strategy, Prototype and others are properly chosen for the construction of this application type. This thesis also describes suitable tools for creating the internet object applciations.

Key words: Design patterns, e-commerce.

Obsah

Použité zkratky

AJAX - Asynchronous JavaScript and XML

APACHE - softwarový webový HTTP server

CSS - Cascading Style Sheets, jazyk pro formátování internetových stránek

HTML - HyperText Transfer Protocol, protokol pro www službu

HTTP - HyperText Transfer Protocol, protokol pro www službu

IP - Internet Protocol, pomocí kterého komunikují zařízení v Internetu

MySQL - relační databázový systém

OOP – Objektově orientované programování

PEAR - PHP Extension and Application Repository, balík skriptů pro PHP

PEAR::MDB2 – balík skriptů pro práci s databází

PHP - skriptovací programovací jazyk

SESSION - relace umožňuje přesnou identifikaci uživatele, sloužící k ukládání dat na webovém serveru

SQL - Structured Query Language, je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích

UML - Unified Modeling Language, jazyk pro návrh a vizualizaci aplikací

Úvod

S Internetem se dnes setkáváme téměř na každém kroku. Primárně se asi člověk dostane k Internetu kvůli elektronické poště, e-mailu. Internet, to jsou hlavně webové stránky, stovky milionů stránek s tím nejrozličnějším obsahem. Na Internetu najdeme opravdu úplně všechno, všechno spojené s lidskou činností, je to největší, nejdostupnější a nejrychleji se rozvíjející virtuální knihovna světa. Nabízí širokou škálu informací i zábavy.

Rozmach Internetu ale přináší určitá úskalí. Na Internetu se totiž objevuje spousta nekvalitních aplikací. Je to dáno jednak z velké části tím, že většina těchto nekvalitních webů je dílem amatérských programátorů, ale objevují se i nekvalitní weby z dílen webových softwarových firem. Tyto firmy mají většinou chybný nebo špatný návrh struktury svých aplikací. Základem práce na jakémkoliv projektu je mít dobře promyšlený návrh aplikace.

Kvalitní objektově orientovaný návrh aplikace je nezbytným předpokladem úspěšné implementace. Požadavky na aplikaci, kterou bude považovat za kvalitní zákazník i dodavatel, jsou shoda původního zadání zadavatele s koncovým funkčním výsledkem dodavatele, dobrá spravovatelnost aplikace, opakovatelná použitelnost, spolehlivost a robustnost aplikace.

Tato diplomová práce se zabývá kvalitním zpracováním a návrhem softwarového produktu. Architektura systému softwarového produktu představuje poměrně náročný problém a na jejím správném navržení většinou závisí úspěch celého systému. Architektura musí být jasná a srozumitelná. Musí obsahovat sdělení, jakým způsobem budou realizovány požadavky od zadavatelů systému. Je určitě zajímavé a užitečné při návrhu architektury systému využívat standardních postupů, které ji zpřehledňují a usnadňují její implementaci.

1. Návrhové vzory

Právě kvalitní návrh softwarového díla je téměř nejpodstatnější při tvorbě celého projektu, jak již bylo zmíněno v úvodu, a právě návrhem objektově orientovaných aplikací se zabývají návrhové vzory.

Design patterns, přeloženo do češtiny návrhové vzory, představují nyní velmi moderní téma v oblasti vývoje objektově orientovaného softwaru. Jako design patterns jsou označovány vzory a modely, které se používají při opakovaném řešení stejné situace, zejména při potřebě rozvržení stálých (konstantních) prvků.

Návrhové vzory představují tedy optimální řešení nejčastěji řešených problémů. Termín design pattern (návrhový vzor) se používá v celé řadě lidských činností. Nejhojněji jsou návrhové vzory využívány v počítačové grafice a to jak v klasických aplikacích, tak ve webových.

Velký rozmach zaznamenalo využití návrhových vzorů v posledních dvou letech, kdy se především softwarové firmy zaměřily na stále se opakující problémy při strukturování obsahu projektů. Od nich převzali tento postup webdesigneři, kteří začali tyto vzory hojně využívat. Pomocí návrhových vzorů lze snáze, levněji a rychleji řešit projekty, které jsou standardizované, nebo velmi podobné již projektům někdy v minulosti realizovaným. Návrhové vzory webových aplikací vycházejí především z výsledků uživatelského testování, kterému jsou dnes a denně na Internetu podrobovány.

Příkladem použití návrhových vzorů je aplikace internetového obchodu, které je zadáním této diplomové práce. Práce chce ukázat na aplikaci internetového obchodu a doprovodné správcovské aplikaci, jak by měl vypadat správný návrh aplikací tohoto druhu. Správný návrh poslouží k sestavení vzoru, který pak následně můžeme lehce využít při tvorbě dalšího nového internetového ochodu. Že vytvoříme vzor pro internetový obchod vůbec neznamená stejnou výchozí grafickou podobu všech internetových obchodů, ba právě naopak, návrhové vzory nám tuto grafickou změnu ještě ulehčí.

Návrhové vzory tedy řeší problém přehlednosti a strukturování navigace, obsahu a dalších důležitých prvků a informací. Jedná se o řešení, které do značné míry eliminuje opakující se problémy při tvorbě typově podobných projektů.

1.1 Historie návrhových vzorů

Pojem návrhový vzor se objevil v informačních technologiích roku 1987 v Orlandu ve Spojených Státech Amerických na konferenci OOPSLA (Object-Oriented Programmin, Systems, Languages and Applications), kde pánové Ward Cunningham a Kent Beck přednášeli na téma objektů a budoucnosti objektově orientovaného programování. Ward Cunningham a Kent Beck vytvořili na základě problémů s designem aplikací několik základních návrhových vzorů. Tyto návrhové vzory měly za úkol vést začínající programátory. Vzory byly sepsány v dokumentu "Using Pattern Languages for Object-Oreinted Programs".

Na počátku devadesátých let minulého století se začal utvářet Gang of Four (GoF), který tvořili pánové Erich Gamma, Richard Helm, Ralph Johnson a John Vlissides. V roce 1991 představil Erich Gamma a Richard Helm několik návrhových vzorů jako jsou Composite, Decider, Observer atd..

Základním a stěžejním dílem pro návrhové vzory v oblasti informačních technologií se stalo dílo Design Patterns: Elements of Reusable Object-Oriented Software GAM95. Autorem byl Gang of Four (Gamma, Helm, Johnson a Vlissides) a hned po vydání roku 1994 se setkalo s velkým úspěchem. Dílo je nazýváno knihou knih v oblasti návrhových vzorů a dodnes se jedná o velmi uznávané dílo. Většina dnešních děl o návrhových vzorech vychází z rozdělení, které je v knize uvedeno.

1.2 Rozdělení návrhových vzorů

Rozdělení návrhových vzorů vychází z knihy GAM95. V tomto díle autoři rozdělili návrhové vzory do tří skupin podle problémů, které vzory řeší. Skupiny návrhových vzoru jsou:

- *Creational Patterns (vytvářející návrhové vzory)*
- *Structural Patterns (strukturální návrhové vzory)*
- *Behavioral Patterns (návrhové vzory zabývající se chováním)*

1.3 Creational Patterns

Creational Patterns (vytvářející návrhové vzory) souvisí s vytvářením objektů v systému. Snahou vytvářejících návrhových vzorů je popsat postup výběru třídy nového objektu a zajistit správný počet nových objektů. Většinou se jedná o dynamická rozhodnutí učiněná za běhu programu. Mezi tyto návrhové vzory patří:

- *Factory Method Pattern*
- *Abstract Factory Method Pattern*
- *Builder Pattern*
- *Prototype Pattern*
- *Singleton Pattern*

1.3.1 Singleton

Návrhový vzor Singleton (jedináček) lze aplikovat všude tam, kde je požadován výskyt jediné instance třídy v rámci aplikace a kde tato instance má být klientům snadno dostupná pomocí známého přístupového bodu.

Vzor se skládá z jediné součásti - třídy jedináčka. Třída implementující vzor Singleton zapouzdřuje přístup ke sdílené instanci a je také zodpovědná za vytváření této instance. Klient je od procesu vytváření instance odstíněn, pouze k ní přistupuje pomocí již zmíněného přístupového bodu.

V aplikaci internetový obchod se návrhový vzor Singleton používá především pro centralizované části aplikace, jako jsou centralizované parametry stránky, připojení k databázi atd.. Názorný příklad jsou např. centralizované parametry stránky. Při neexistenci instance třídy parametry vytvoříme instanci novou. Pokud poté chceme znovu vytvořit instanci třídy parametry, objekt se nevytváří, ale dostáváme zpět již existující instanci třídy parametry a není vytvořena instance nová. To se děje na základě statické proměnné, která je společná pro všechny instance dané třídy.

1.3.2 Factory Method Pattern

Factory Method Pattern neboli tovární metoda je nejpoužívanější návrhový vzor v ukázkové aplikaci internetový obchod této diplomové práce. Úkolem tovární metody je rozhodnout až v průběhu programu o vytvoření instance konkrétní třídy.

Princip tovární metody je postaven na předpokladu existence několika tříd, které mají obvykle společného předka, který implementuje maximum společných vlastností potomků, kde základem je potomky vytvářející metoda factory. Potomci poskytují různé služby nad různými daty a tovární metoda potom dovoluje vybrat v průběhu programu vytvoření instance některé z těchto tříd.

V průběhu chodu aplikace je nutné rozhodnout o principu výběru třídy, na jejímž základě bude objekt vytvořen.

Návrhový vzor tovární metoda je v aplikaci internetový obchod založen na třídě Krabice, která je základním tvůrcem instancí tříd. Třída Krabice implementuje statickou metodu factory, která se stará o vytváření instancí podřízených tříd např. instance třídy pro vyhledávání.

1.3.3 Abstract factory Method Pattern

Abstraktní tovární metoda (Abstract Factory Method Pattern) se zabývá úkolem vytvořit na základě rozhodnutí při běhu programu instanci třídy, která dále vytváří instance souvisejících nebo závislých tříd.

Abstract Factory Pattern úzce souvisí s návrhovým vzorem Factory Method Pattern. Factory Method Pattern je vhodné použít pokud chceme vytvořit individuální objekt, aniž by klient specifikoval třídu nového objektu. Jestliže je nutné vytvořit více objektů, které spolu nějakým způsobem souvisí, je vhodné využít Abstract Factory Method Pattern.

Tento návrhový vzor není v diplomové práci použit. Lépe se do struktury aplikace hodí související vzor tovární metoda kvůli množství tříd, které pracují nad různým typem dat.

1.3.4 Builder Pattern

Návrhový vzor Builder Pattern má za cíl oddělit konstrukci složitých objektů od jejich prezentace. Umožňuje stejným postupem vytvářet různé prezentace.

Builder Pattern lze použít, pokud máme různé objekty, které mají podobný proces konstrukce např. návrhový vzor Builder se používá pro převody souboru z určitého formátu na formát jiný, který předem neznáme.

1.3.5 Prototype Pattern

Návrhový vzor Prototype Pattern řeší, jak vytvořit kopii existujícího objektu místo vytváření nového objektu.

Pro účely vytvoření kopie už existujícího objektu je implementována ve většině programovacích jazycích metoda __clone() nebo metoda serializable() a její inverzní metoda unserializable(). Programovací jazyk PHP, který je použit k vypracování vzorového příkladu internetového obchodu této diplomové práce, má implementovány všechny zmíněné metody a to od verze PHP 5.0, která přinesla podporu objektového programování.

Nevýhoda metody __clone() je její deklarace jako protected, což znamená, že může být volána pouze z dané třídy nebo ze tříd potomků, kteří tuto metodu zdědili. Další nevýhodou je možnost klonovat pouze objekty, které deklarují a implementují metodu __clone().

Metoda klonování serializable() funguje na principu převedení objektu na posloupnost bajtů. Poté inverzní funkcí unserializable() lze tuto posloupnost bajtů opět převést do podoby objektu a tím ho klonovat do více instancí.

Návrhový vzor Prototype Pattern můžeme použít v případě, kdy vytvoření nové instance určité třídy je časově nebo zdrojově náročné, nebo když chceme uchovat instanci třídy i po skončení běhu skriptu. Používá se také pokud chceme vytvořit nový objekt pouze s několika odlišnostmi od prototypu.

Ukázková aplikace internetový obchod používá tento návrhový vzor k uchovávání objektů i po skončení běhu skriptu. Při novém započetí skriptu, lze objekty načíst, používat předešlé uložené hodnoty, a nevytvářet tak objekty nové.

1.4 Structural Patterns

Structural Patterns představují skupinu návrhových vzorů zaměřujících se na možnosti uspořádání jednotlivých tříd nebo komponent v systému. Snahou je zpřehlednit systém a využít možností strukturalizace kódu. Mezi tyto návrhové vzory patří:

- *Adapter Pattern*
- *Bridge Pattern*
- *Composite Pattern*
- *Decorator Pattern*
- *Facade Pattern*
- *Flyweight Pattern*
- *Proxy Pattern*

1.4.1 Adapter Pattern

Návrhový vzor Adapter Pattern má za úkol přizpůsobit určitou třídu tak, aby ji bylo možné využívat i jiným než požadovaným způsobem. Musíme však zajistit funkční konverzi rozhraní třídy na rozhraní třídy jiné.

Návrhový vzor Adapter Pattern se používá k přizpůsobení „cizí“ nekompatibilní interface ke klientovi, který očekává nějaký známý interface.

1.4.2 Bridge Pattern

Návrhový vzor Bridge řeší problém oddělení rozhraní třídy od její vlastní implementace. Zapříčiňuje vytváření rozhraní a implementace nezávisle na sobě. To zapříčiní, že může být změněna implementace třídy bez toho, abychom měnili kód klienta.

Návrhový vzor Bridge souvisí s návrhovým vzorem Adapter. Jsou si ve velké části podobné, Adapter řeší problém, jak zajistit komunikaci již hotových tříd, zatímco Bridge vstupuje do hry již v době návrhu. Adapter prakticky představuje způsob konverze rozhraní již existující třídy na třídy odlišné.

1.4.3 Composite Pattern

Návrhový vzor Composite představuje řešení, jak uspořádat jednoduché objekty, atomické (primitivní), a z nich složené (kompozitní) objekty. Složené objekty se rekurzivně skládají z primitivních a dalších složených objektů. Tedy složený (kompozit) objekt je objekt, který obsahuje kolekci jiných objektů, z nich každý může být buď jednoduchý objekt, nebo opět složený objekt. Jednoduchý objekt nemá referenci na žádné jiné objekty.

Použití nachází návrhový vzor Composite v reprezentaci stromové struktury.

1.4.4 Decorator Pattern

Návrhový vzor Decorator je použitelný pro řešení problému, jak změnit vlastnosti instance třídy bez nutnosti vytvářet novou odvozenou třídu. Představuje možnost jak dynamicky složit chování objektů.

Návrhový vzor Decorator se používá např. v grafický toolkitech (XWindow, Swing), smart-pointerech (boost, loki, ...) nebo v Java I/O.

1.4.5 Facade Pattern

Návrhový vzor Facade lze použít, pokud potřebujeme zjednodušit vstupní bod do systému.

Při vývoji systému se lehce může stát, že celková struktura systému se stává velmi těžce zvládnutelnou. I použitím návrhových vzorů se mnohdy dostáváme k velmi komplexnímu systému tříd s mnoha vazbami. Navíc každý systém může obsahovat i několik subsystémů, které ještě zvětšují obtížnost pochopení zákonitostí systému. Facade představuje rozhraní, které slouží k zjednodušení komunikace mezi klientem a systémem.

1.4.6 Flyweight Pattern

Návrhový vzor Flyweight řeší problém, jakým způsobem zajistit efektivní správu velkého množství objektů, které se příliš neodlišují. Tento návrhový vzor se snaží využít sdílení stejných vlastností těchto objektů.

Návrhový vzor Flyweight nachází použití např. při vytváření aplikací textových

editorů typu Word, které budou podporovat velké množství netradičních znakových sad.

1.4.7 Proxy Pattern

Návrhový vzor Proxy se používá při potřebě zajistit kontrolu nad přístupem k jinému objektu. Existují čtyři přístupy při kontrolovaní přístupu k jinému objektu. Jsou to přístupy Protective Proxy, Remote Proxy, Smart Proxy a Virtual Proxy .

Virtual proxy zajišťuje kontrolu nad přístupem k objektu až ve chvíli poptávky po objektu a to kvůli náročnosti vytvoření kontrolovaného objektu. Proxy implementuje logiku rozhodující o nutnosti vytvoření kontrolovaného objektu.

Remote proxy reprezentuje objekt umístěný na jiném počítači.

Protective proxy kontroluje přístup ke kontrolovanému objektu na základě manipulačních práv objekt.

Smart proxy nabízí před volání kontrolovaného objektu doplňkové operace např. počet referencí atd..

1.5 Behavioral Patterns

Behavioral Patterns se zajímají o chování systému. Mohou být založeny na třídách nebo objektech. U tříd využívají při návrhu řešení především principu dědičnosti. V druhém přístupu je řešena spolupráce mezi objekty a skupinami objektů, která zajišťuje dosažení požadovaného výsledku. Mezi tento typ vzorů můžeme zařadit:

- *Chain Of Responsibility Pattern*
- *Command Pattern*
- *Interpreter Pattern*
- *Iterator Pattern*
- *Mediator Pattern*
- *Memento Pattern*
- *Observer Pattern*
- *State Pattern*
- *Strategy Pattern*
- *Template Pattern*
- *Visitor Pattern*

1.5.1 Chain Of Responsibility Pattern

Návrhový vzor Chain Of Responsibility Pattern řeší způsob, jak zpracovat požadavek bez přesného určení objektu, který jej má zpracovat.

Základem je existuje několika instancí různých tříd, které mohou potenciálně zpracovat požadavek. Klient jej odesílá bez přesného určení konkrétní instance. Třídy, které jsou schopny požadavek zpracovat, musí implementovat stejné rozhraní.

Návrhový vzor vzor Chain Of Responsibility se používá při potřebě snížit závislost klienta a objektu, který bude zpracovávat daný požadavek Jeden z příkladů využití tohoto návrhového vzoru je princip dědičnosti v objektových programovacích jazycích. Jestliže je volána funkce třídy, která tuto funkci neobsahuje, dochází k hledání funkce ve třídách, z kterých je zděděna.

1.5.2 Command Pattern

Návrhový vzor Command odstíňuje klienta od procesu zpracování jeho požadavku. Klient pouze zadá požadavek a určí zpracovatele, ale už se nezabývá způsobem vykonání požadavku.

Návrhový vzor Command vytvoří požadavek jako objekt, ten je možné parametrizovat a měnit konktrétní požadavky upravením metody execute(), která je volána a požadavek zpracuje.

Tento návrhový vzor se používá hlavně pro zpětné a dopředné operace. Vzor je schopen uchovat si dostatek informací před voláním metody execute, aby byl schopen nastolit zpět výchozí stav.

1.5.3 Interpreter Pattern

Interpreter Pattern se zabývá řešením určitého problému, který se dostatečně často vyskytuje. Vytvoříme proto jednoduchý jazyk popisující problém. Následně implementujeme interpreter, který bude definovaný problém řešit

Jde o to, že klientovi nabídneme určitý vstup na definování postupů v logickém jazyce. Není nejlepší volbou nechat psát klienta postupy přímo ve složitém vývojovém jazyce. Musíme vytvořit jednoduchý a srozumitelný jazyk. Jazyk by měl obsahovat termíny obyčejného jazyka.

Praktické využití návrhový vzor Interpreter nachází v kompilátorech např. php.net, nebo v nástrojích pro parsování XML atd..

1.5.4 Iterator Pattern

Návrhový vzor Iterator se zabývá pohybováním mezi prvky, které jsou sekvenčně uspořádány, bez znalosti implementace jednotlivých prvků posloupnosti.

Iterator je jeden z nejjednodušších a nejvíce používaných návrhových vzorů. Pohyb mezi objekty je v programovacím jazyce PHP definován pomocí cyklu foreach.

1.5.5 Mediator Pattern

Návrhový vzor Mediator řeší jakým způsobem zajistit komunikaci mezi dvěma komponenty programu, aniž by byly v přímé interakci a tím musely přesně znát své poskytované metody**.**

Mediator se stará o centralizovanou komunikaci mezi třídami, která by jinak byla rozdělena mezi více objektů. Třídy se navzájem vůbec neznají, třídy pouze oznamují svou změnu mediatorovi, který poté upozorní na změny třídy závislé. Tím vytváříme závislost tříd na mediatoru a ne závislost třída na třídě.

1.5.6 Memento Pattern

Návrhový vzor Memento se stará o zachycení a uchování interního stavu objektu bez porušení jeho zapouzdření. Účelem je znovuobnovení objektu do původního, uchovávaného stavu.

Memento pattern představuje přístup, jak zajistit uchování stavu objektu, aniž by se tento stav stal veřejně přístupným a mohl být ovlivněn (změněn), jinou částí systému. Stav objektu je definován jako sada proměnných, proto zapíšeme proměnné do externího datového zdroje (souboru) nebo vytvoříme speciálně pro účely uchování nový objekt s definovanými právy na přístup.Verze se speciálním novým objektem je bezpečnější, protože v souboru se stavy stávají veřejně přístupné

1.5.7 Observer Pattern

Observer Pattern definuje závislost objektu na více objektech. Při nastalé události informuje závislé objekty (pozorovatele).

Observer je možné použít, když je definována závislost jednoho objektu na jiném objektu (pozorovateli). Závislost znamená při změně objektu informovat pozorovatele o změně. Nad objekty, závislým a pozorovatelem, je definován nezávislý objekt, který musí informovat závislé objekty o událostech, které je mohou ovlivnit.

Tento návrhový vzor se používá na komunikaci mezi komponentami např. v programovacím jazyku Java.

1.5.8 State Pattern

Návrhový vzor State řeší problém při změně vnitřního stavu objektu. Po změně vnitřního stavu se objekt jeví jako instance jiné třídy.

Chování objektu se může měnit v různých situacích. Nejčastěji se mění při přechodu do jiného stavu nebo při změně argumentů. Obvyklé způsoby řešení těchto stavů je na základě podmínek. Vzor State řeší problém elegantněji vytvořením tříd, které zapouzdřují rozdílné chování objektu v závislosti na jeho stavu.

1.5.9 Strategy Pattern

Problém, který řeší návrhový vzor Strategie je určení skupiny algoritmů a zapouzdření každého algoritmu do samostatného objektu. Poté dává možnost záměny těchto algoritmů. Řeší problém změny algoritmu nezávisle na klientovi, který jej využívá.

Základní podmínkou návrhového vzoru je předpoklad, že existuje více podobných řešení stejného problému. Tento návrhový vzor uvažuje o skupině algoritmů, které řeší stejnou funkčnost, ale rozdílným způsobem.

Používá se např. při rozhodnutí, jaký operační systém klient používá, a na základě tohoto rozhodnutí nabídnou různé soubory ke stažení, čehož využívá i ukázková aplikace internetového obchodu.

1.5.10 Template Pattern

Návrhový vzor Template řeší problém, jak mohou zděděné třídy ovlivnit kroky algoritmu, který je implementován ve třídě předka.

Ve třídě je definován neměnný algoritmus. Algoritmus je složen z několika kroků a musí existovat více možností, jak změnou těchto kroků modifikovat algoritmus. Pak se rozhodneme, které z kroků jsou neměnné a je možné je nechat jako stálou součást kostry algoritmu a které naopak mají více možností implementace.

Když se rozdělí algoritmus na neměnné a variabilní části, je dosaženo jeho lepší využitelnosti. Změnou jednotlivých kroků, můžeme dosáhnout rozdílný výsledek stejného postupu. Jelikož je tento postup udržován pouze na jednom místě, eliminují se problémy s duplicitou kódu. Template Method vzor je základní součástí objektově orientovaného programování.

1.5.11 Visitor Pattern

Návrhový vzor Visitor má za úkol vytvořit metodu, která bude pracovat s více objekty, jež jsou uspořádány do objektové struktury. Při vytvoření nové metody by nemělo být potřeba měnit objekty, s nimiž pracujeme.

Je definována struktura objektů a má být implementováno více odlišných operací využívající objekty v této struktuře. Příkladem může být určitý výpočet, který chceme vytvořit z dat zapouzdřených v objektech. Je možné přidat do každého objektu novou metodu, která bude poskytovat potřebná data, ale pokud existuje mnoho metod, které pracují na uvedeném principu, může být nevhodné „znečisťovat“ použité třídy metodami tohoto typu. Dále se chceme vyhnout dotazování se na třídu u každého objektu ve struktuře a provádět přetypování.

2 Použité technologie v aplikaci internetový obchod

2.1 Objektově orientované PHP

Objektově orientované programování, dále už jen OOP, je metoda tvorby programů, která se snaží napodobovat způsob zacházení s věcmi v reálném životě. Přináší to své výhody v přehlednosti, lepší struktuře kódu a modulárnosti, oproti klasickému procedurálnímu programování. V programovacím jazyce PHP (Hypertext preprocesor) je implementována plná podpora OOP od verze PHP 5.0.

Objekt je názvem jednoznačně identifikovatelná entita, která má své atributy vlastnosti a metody. Klient je při práci s objektem oddělen od vnitřní implementace rozhraním, klient pracuje pouze s názvy metod, tomu se říká zapouzdření.

Třída je šablonou objektu. Definujeme ji pomocí příkazu class, za kterým je jednoznačně identifikující název třídy např. class Krabice. Konstruktor třídy je metoda, jež má stejné jméno jako třída nebo má výchozí název __construct(), a jejímž úkolem je prvotní inicializace objektu. Destruktor třídy je metoda, která se provede těsně před odstraněním objektu z paměti a lze ji využít např. pro uložení objektu funkcí serializable() do souboru. Metody konstruktoru i destruktoru začínají dvěma podtržítky z důvodu možných budoucích problémů s kompatibilitou.

Modifikátory proměnných a metod umožňují úpravy přístupu k vlastnostem a metodám uvnitř objektu. Jedná se o modifikátory private, protected, static a abstract, jež fungují obdobně jako v jiných programovacích jazycích podporujících OOP.

Pomocí modifikátoru private lze vlastnost či třídu označit jako soukromou, což znamená, že ji nelze použít vně příslušné třídy. Modifikátor protected funguje podobně jako modifikátor private, ovšem použití vlastnosti či metody není omezeno pouze na objekty příslušné třídy, ale také na objekty tříd od ní odvozených. Modifikátor static umožňuje deklarovat metody i vlastnosti jako statické, tj. společné pro všechny instance téže třídy. Pomocí modifikátoru abstract lze deklarovat metody třídy jako abstraktní, tedy bez konkrétní implementace, která se doimplementuje ve třídě potomka.

Dědičnost je v programovacím jazyce PHP označována klíčovým slovem extends. Původně definovaná třída se nazývá předek a nově vytvořená třída se nazývá potomek. Potomci dědí všechny vlastnosti a metody od předka a mohou tyto vlastnosti a metody rozšiřovat.

2.2 MySQL

MySQL je relační databázový systém typu DBMS (database managment system), kdy každá databáze databázového systému MySQL je tvořena z jedné nebo více tabulek, které mají řádky a sloupce. V řádcích rozeznáváme jednotlivé záznamy, sloupce mají jméno a uvozují datový typ jednotlivých polí záznamu. Práce s databázemi, tabulkami a daty se provádí pomocí příkazů, respektive dotazů vycházejí z deklarativního programovacího jazyka SQL (Structured Query Language)

MySQL je velmi snadno implementovatelný (lze jej instalovat v operačních systémech Unix , MS Windows a dalších), má vysoký výkon a jedná se o volně šiřitelný software. MySQL je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí. Právě tyto klady přispívají k oblibě MySQL. Technologie používané pro vývoj ukázkové aplikace internetového obchodu je kombinací MySQL, PHP a Apache jako základního software webového serveru.

Systém MySQL byl od počátku optimalizován na rychlost, a to i za cenu některých zjednodušení. Zjednodušení představovalo donedávna neimplementaci některých standardních prvků databázových systémů, jako jsou pohledy, triggery, a uložené procedury a funkce, které přinesla až verze MySQL 5.0. Tyto vlastnosti jsou právě hojně využívány v databázi, kterou používá ukázková aplikace internetový obchod této diplomové práce.

2.3 CSS

CSS (Cascading Style Sheets) neboli kaskádové styly jsou kolekcí metod pro grafickou úpravu webových stránek. Kaskádové styly jsou kaskádové, protože se na sebe mohou vrstvit definice stylu, mohou se přepisovat a pak platí ta poslední.

CSS souvisí velmi těsně s HTML. Styl se může nadeklarovat třemi způsoby a to přímo v textu, interním a externím stylopisem (stylesheet).

Aplikace internetový obchod používá externí stylesheet umístěný v souboru direct/css/style.css. Stránka se na tento soubor odkazuje tagem <link>. Výhody je možnost linkovat extrní stylesheet na více stránek a společné umístění všech stylů v jednom centralizovaném souboru.

2.4 HTML

HyperText Markup Language (HTML), značkovací jazyk pro hypertext, je jedním z jazyků pro vytváření stránek pro WWW (World Wide Web), který publikuje stránky na Internetu.

Jazyk HTML je charakterizován množinou značek (tagů) a jejich atributů. Tagy se uzavírají mezi úhlové závorky ("<" a „>“). Tagy rozlišujeme párové a nepárové. Párové tagy jsou složeny z počátečního a koncového znaku, kdy koncový znak se liší od počátečního přidáním znaku “/”. Mezi tagy se uzavírají části textu dokumentu a tím se určuje význam (sémantika) obsaženého textu. Část dokumentu uzavřená mezi párové tagy vytváří tzv. element (prvek) dokumentu. Součástí obsahu elementu mohou být další vnořené elementy. Atributy jsou doplňující informace, které upřesňují vlastnosti elementu.

2.5 UML

Unified Modeling Language (UML) je jazyk, který usnadňuje návrh a vizualizaci různých typů aplikací. UML umožňuje modelovat jednoduché i složité aplikace pomocí stejné formální syntaxe a tím umožňuje čitelnost pro každého se znalostí této syntaxe.

2.6 JavaScript

JavaScript je klientský objektově orientovaný skriptovací jazyk. Zpravidla se používá jako interpretovaný programovací jazyk pro WWW stránky, vkládaný přímo do HTML kódu stránky. Jsou jím obvykle ovládány různé interaktivní prvky grafického uživatelského rozhraní (GUI), jako jsou tlačítka, textová políčka nebo tvořeny animace a efekty obrázků. Pro jeho funkčnost je zapotřebí a nutná podpora JavaScriptu internetovým prohlížečem.

2.7 AJAX

AJAX (Asynchronous JavaScript and XML) je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů.

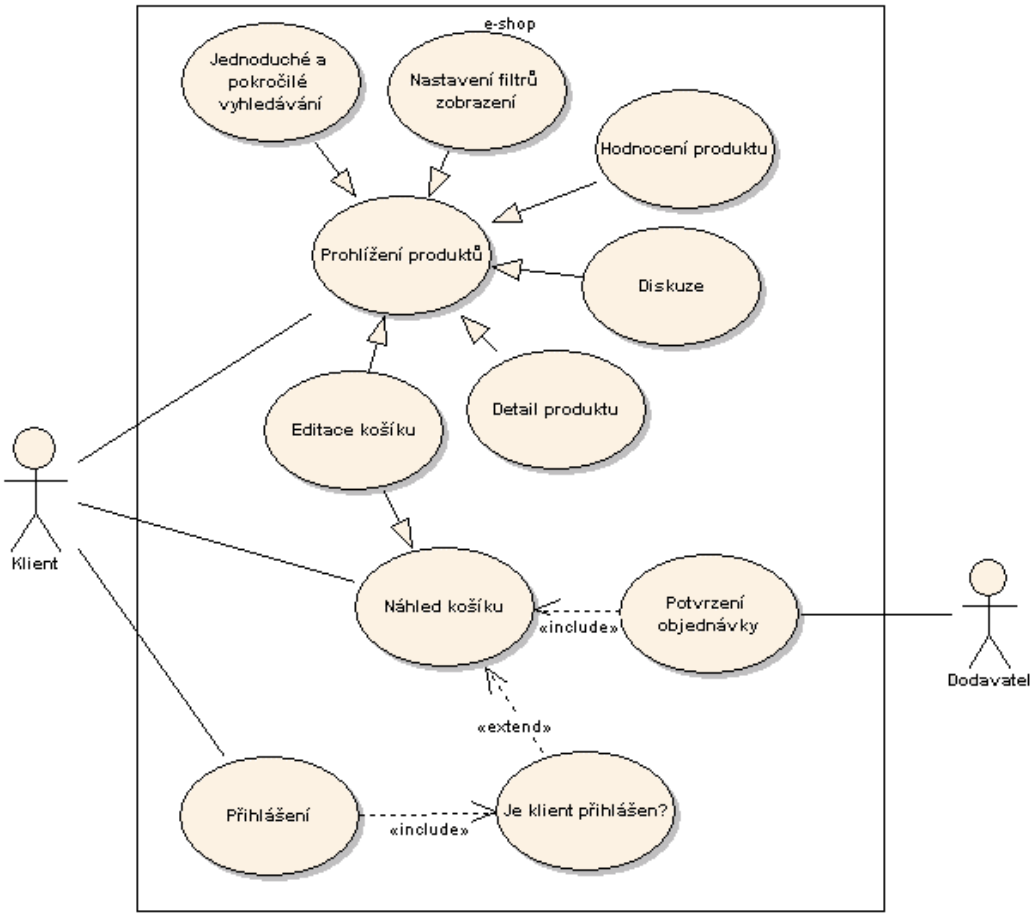
Mezi velké výhody ajax technologie patří velké snížení zátěže serveru díky nutnosti nenačítání stránky celé, ale jen její části.

3 Aplikace internetový obchod

3.1 Úvod

Aplikace vypracovaná v rámci této diplomové práce je internetový obchod zabývající se prodejem počítačových komponent a spotřební elektroniky. Klient má možnost vybrat si mezi sto-třiceti produkty.

Internetový obchod umožňuje klientovi listovat v produktech, řadit produkty podle různých kritérií, fulltextově vyhledávat, diskutovat o produktech, hlasovat o jejich kvalitě atd. viz obr. 3-1.



Obr. 3-1 – UML use case diagram internetového obchodu.

UML diagram posisuje jednoduše, co všechno může klient v obchodě dělat. Základní možnosti klienta jsou přihlášení, náhled košíku a prohlížení produktů. Další možnosti rozšiřují (generalizují) základní možnosti klienta např. detail produktu rozšiřuje prohlížení produktů.

3.2 Souborová a adresářová struktura

Základní element programového kódu aplikace internetový obchod je nazván krabice. Tyto krabice jsou koncipovány tak, aby se chovaly jako autonomní části zdrojového kódu, jakési moduly. Každá z krabic představuje jednu třídu a každá z krabic je uložena v souboru, který má příponu (*.class.php). Přehled krabic a jejich funkce viz. tabulka 3-1.

jméno krabice (*.class.php)	**funkce**
DetailDoruceni	Detail o zákazníkovi, způsobu platby a dopravy.
DetailKosik	Detailní náhled košíku.
DetailObjednavka	Detailní přehled objednávky.
DetailProdukt	Detailní informace o produktu, hodnocení a diskuze.
DetailZakaznik	Registrační formulář, změna kontaktních údajů.
FormularDiskuze	Formulář na přidání příspěvku do diskuze.
HtmlFooter	Patička HTML stránky.
HtmlHeader	Hlavička HTML stránky.
Info	Zobrazení nápovědy, servisu, reklamace atd.
Menu	Hlavní menu obchodu.
PokrocileVyhledavani	Pokročilé vyhledávání.
PravyBox	Box přihlášení, košík, filtry a jednoduché vyhledávání
PrazdnaKrabice	Předek všech krabic.
SeznamProduktu	Listování produkty.
Tree	Strom sekcí.
Vyhledavani	Vyhledávání.

Tab. 3-1 – Přehled krabic a jejich funkce

Adresářová struktura projektu internetový obchod je rozdělena do adresářů viz tabulka 3-2.

Adresář	Funkce
cnf	Obsahuje soubor (base.const.php) s definicemi konstant.
common	Obsahuje společné třídy pro celou aplikaci (kosik, page, parametry, zakaznik atd.), soubor se společnými funkcemi (common.func.php) a soubor pro připojením k databázi (doDb.php).
direct	Obsahuje soubory kaskádových stylů (style.css) a JavaScriptové soubory (function.js) včetně souborů pro práci s AJAX technologií (direct\js\Ajax).
external	Obsahuje cizí doinstalované části programu, jako je balík PEAR atd.
file	Obsahuje soubory, které si klient obchodu může stáhnout (příkladem je soubor ceníku).
gf	Obsahuje grafickou prezentaci obchodu (obrázky používané na elementy nebo v elementech HTML kódu).
info	Obsahuje textové include soubory (*.inc) s nápovědou jak si v obchodě nakoupit, kontaktem, servisem a reklamací.
krabice	Obsahuje soubory krabic, které reprezentují jednotlivé třídy.
src	Obsahuje obrázky produktů prodávaných v internetovém obchodě. Jsou pojmenovány podle svého identifikačního čísla, které jim náleží v databázi internetového obchodu.
template	Obsahuje grafické vzory jednotlivých stránek.

Tab. 3-2 – Přehled adresářů projektu internetový obchod

3.3 Základní struktura

Základnou pro strukturu internetového obchodu jsou třídy umístěné ve společném adresáři common.

V souboru common/class/krabice.class.php je umístěna třída krabice, která implementuje metodu factory(). Tato metoda zajišťuje základ návrhového vzoru tovární metoda, který má až za běhu skriptu rozhodnout o vytvoření instance konkrétní třídy.

Metoda factory() je implementována jako statická funkce, které jsou předávány parametry jméno nově vytvářené krabice, titulek krabice a parametry potřebné pro tvorbu krabice.

V těle samotné factory metody nejdříve testujeme, zda bylo zadáno jméno krabice a zda náhodou již daná třída nebyla v průběhu skriptu načtena a existuje. Pokud obě tyto podmínky proběhnou s pozitivním výsledkem (true), dochází k připojení obsahu souboru (include_once) krabice k vykonávanému skriptu. Po připojení souboru ke skriptu vytvoříme novou instanci třídy krabice s danými parametry a vytvořenou instanci vracíme jako návratovou hodnotu z této statické funkce zpět na místo jejího volání.

Volání metody factory probíhá za běhu programu viz. programový kód na obrázeku 3-1.

```
7 // vytvorime HTML hlavicku
8 $HTMLhlavicka = Krabice::factory('HtmlHeader', '',array());
9 $HTMLhlavicka = $HTMLhlavicka->vystup();
```

Obr. 3-2 –Ukázka volání metody factory()

3.4 Připojení k databázi

Připojení k databázi je realizováno pomocí balíku PEAR, konkrétně balíčku na správu databázového připojení PEAR::MDB2. Balík skriptů PEAR::MDB2 je umístěn v adresáři external. Samotné připojení k databázi je implementováno v souboru common/DoDB.php, který obsahuje připojení klíčového souboru balíčku PEAR::MDB2, kterým je soubor external/mdb2.php. Skript na připojení k databázi dále obsahuje konstanty pro připojení. Konstanty jsou adresa serveru s databází, jméno databáze, jméno uživatele a heslo uživatele databáze. Pro připojení k databázi je využíván návrhový vzor Singleton, který zajistí, že objekt připojení k databázi se nevyskytne v duplicitách. Po vytvoření připojení k databázi SQL dotazem zadáváme používanou znakovou sadu databáze, která je v případě databáze internetového obchodu windows 1250.

3.5 Zobrazování stránek

Rozlišení, na jakou stránku se má aplikace internetového obchodu směrovat, se usuzuje podle url odkazu nebo podle dodaných parametrů od klienta. Zpracování skriptů vždy začíná ve skriptu index.php. V tomto skriptu se přidávají (require_once) skripty nutné k běhu internetového obchodu, jako je skript na připojení k databázi atd.. Seznam všech nutných skriptů a jejich funkce viz. tabulka 3-3.

Soubor	funkce
cnf/base.const.php	Definice konstant
class/Page.class.php	Obsahuje třídu pro práci se stránkou, která parsuje url a ukládá nové parametry stránky do objektu parametry. Nové parametry jsou získávány z polí $_HTTP_GET_VARS, $_HTTP_POST_VARS. Třída je Singleton.
class/Parametry.class.php	Soubor obsahuje třídu parametry. Třída uchovává klientské nastavení. Objekt je na počátku zpracování skriptu serializován, pokud existoval v minulosti. Třída je Singleton.
class/Zakaznik.class.php	Soubor obsahuje třídu zákaznik. Třída zapouzdřuje práci nad zákazníkem. Uchovává jeho stav (přihlášen/ nepřihlášen), jeho kontaktní údaje (titul, jméno, příjmení, adresu, město atd.). Objekt je na počátku zpracování skriptu serializován, pokud již existoval v minulosti. Třída je Singleton.
class/Kosik.class.php	Soubor obsahuje třídu kosik. Třída kosik umožnuje práci s nákupním košíkem v aplikaci. Objekt je na počátku zpracování skriptu serializován, pokud již v minuloasti existoval. Třída je Singleton.
class/Strategie.class.php	Soubor se třídou strategie. Demonstrace návrhového vzoru strategie.
doDb.php	Soubor s připojením k databázi.

class/Krabice.class.php	Soubor se třídou krabice.Třída krabice je základním tvůrcem objektů, protože implementuje statickou metodu factory().
krabice/PrazdnaKrabice.class.php	Předek všech krabic, snaží se implementovat maximum společných vlastností všech tříd.
common.func.php	Tento soubor obsahuje společné funkce pro aplikaci.

Tab. 3-3 – Přehled skriptů nutných pro běh aplikace

Ve skriptu index.php jsou dále testovací podmínky, zda existují session proměnné parametry, zakaznik a kosik. Na základě výsledku těchto podmínek buď objekty parametry, zakaznik a kosik vytvoříme nové, nebo pokud session proměnné existují, unserializujeme objekty tzn. načteme session proměnné a převede je do podoby objektů uskutečněním návrhového vzoru Prototype.

Výhoda serializace a unserializace je zřejmá, dokážeme si uchovat vlastnosti objektů i po skončení běhu skriptu.

Ve skriptu se poté testuje, zda dodané parametry, zpracované ve třídě Page, odpovídají podmínkám pro vstup na určitou stránku. Ve třídě Page dochází k testování, zda hodnota proměnné sekce, přijatá v poli $_HTTP_GET_VARS se rovná nějaké položce z pole statickeCesty, které je pevně definované v souboru base.const.php.

V poli statickeCesty jsou definovány všechny přípustné možnosti směrování v obchodě, jedná se tedy o jakousi pojistku proti směrování na jinou než dovolenou stránku. Při obdržení neplatného parametru sekce, dojde k jeho anulování, je provedeno unset($sekce). Pokud je k proměnné sekce nalezen odpovídající oponent v poli statickeCesty, dojde k nastavení vlastnosti sekce objektu parametry a s touto vlastností se dále pracuje.

Právě na základě vlastnosti sekce objektu parametry testujeme, jakou stránku chce klient vidět. Testování probíhá v podmínkách jdoucích za sebou v návaznosti if (podmínka) elseif (podmínka), to pro co možná nejmenší počet testování. Pokud vlastnost sekce objektu parametry nevyhoví žádné podmínce, je zobrazena domovská stránka obchodu.

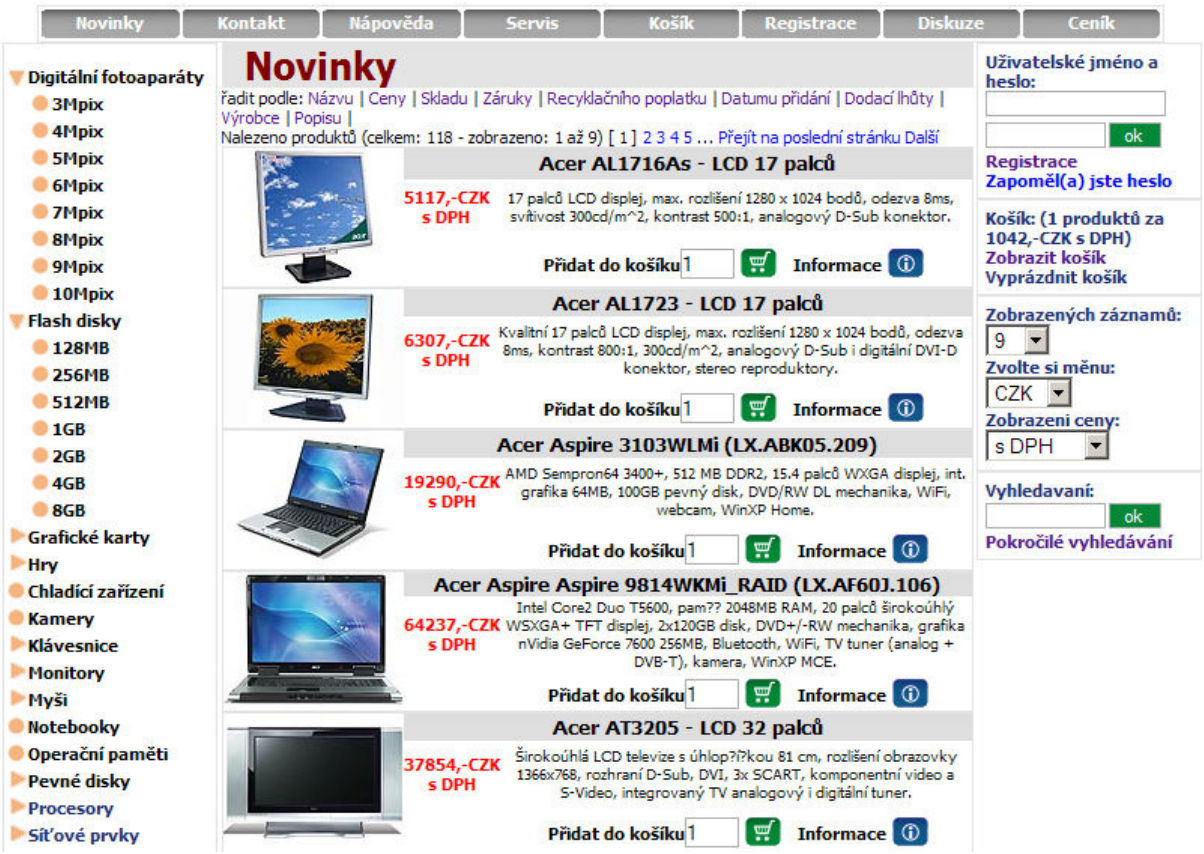
Pokud vlastnost sekce objektu parametry odpovídá některé hodnotě z pole

statických cest, jsou do probíhajícího skriptu přidány (require_once) soubory příslušející chtěné stránce z adresáře template, které se starají o stavbu stránky a její grafické rozvržení.

Template soubory jsou vždy pro danou stránku dva. Soubor s příponou (*.krabice.php) se stará o tvorbu nových objektů tovární metodou, a soubor (*.tpl.php) se stará o rozdělení jednotlivých částí kódu, získaných jako výstup od jednotlivých krabic, mezi blokové elementy HTML. Blokovým elementům kódu přiřazuje vlastnost class, jejíž definice je v souboru kaskádových stylů.

3.6 Domovská stránka

Domovská stránka aplikace internetového obchodu je výchozí stránkou. Jsou zde zobrazovány všechny produkty podávané v tomto obchodě. Grafický výstup stránky je znázorněn na obr. 3-3.



Obr. 3-3 – Ukázka grafického výstupu internetového obchodu, domovská stránka.

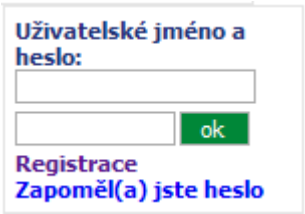
3.7 Menu

Menu internetového obchodu je implementováno v souboru krabice/Menu.class.php. Implementace obsahuje odkazy na stránku novinek (domovská stránka), stránky kontaktu, zobrazení košíku, odkaz na stáhnutí ceníku a další. Odkaz košík, který ukáže klientovi našeho obchodu obsah jeho košíku, je podmíněn existencí objektu kosik a nenulovostí počtu produktů v košíku. Při neexistenci objektu kosik a nulovosti počtu produktů se odkaz stává neaktivním tzn. neodkazuje na žádnou stránku.

V liště menu je dobře demonstrován návrhový vzor Strategie, který řeší problém určení skupiny algoritmů a zapouzdření každého algoritmu do samostatného objektu. Řeší problém změny algoritmu nezávisle na klientovi, který jej využívá. Návrhový vzor strategie se ukrývá pod odkazem ceník. Skupinu algoritmu v této situaci představuje problém nabídnutí stáhnutelného souboru s ceníkem našeho obchodu klientovi na základě klientova operačního systému. Pokud přijde klient z operačního systému Windows nebo Unix, skript to pozná na základě hodnoty HTTP_USER_AGENT z pole $_SERVER a vytvoří tomu příslušný objekt spravující stáhnutí souboru pod daným operačním systémem. Pro případ operačního systému Windows je vytvořen objekt pro stáhnutí souboru cenik.zip a pro případ operačního systému Unix objekt pro stáhnutí souboru cenik.tar.gz.

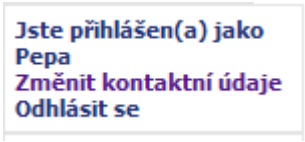
3.8 Přihlášení klienta

Klient se může přihlásit do systému internetového obchodu prostřednictvím boxu na přihlášení viz. obrázek 3-4.



Obr. 3-4 – Box přihlášení zákazníka

Box obsahuje textový vstup na přihlašovací jméno, maskovací vstup pro heslo, odesílací tlačítko pro přihlášení a odkazy registrace a zapomněl jste heslo. Pokud je již klient registrován (má již účet viz. registrace), a zná své přihlašovací jméno a heslo a zadá ho do textových vstupních polí a potvrdí, systém klienta přihlásí viz. obrázek 3-5.



Obr. 3-5 –Přihlašovací box aplikace po úspěšném přihlášení zákazníka

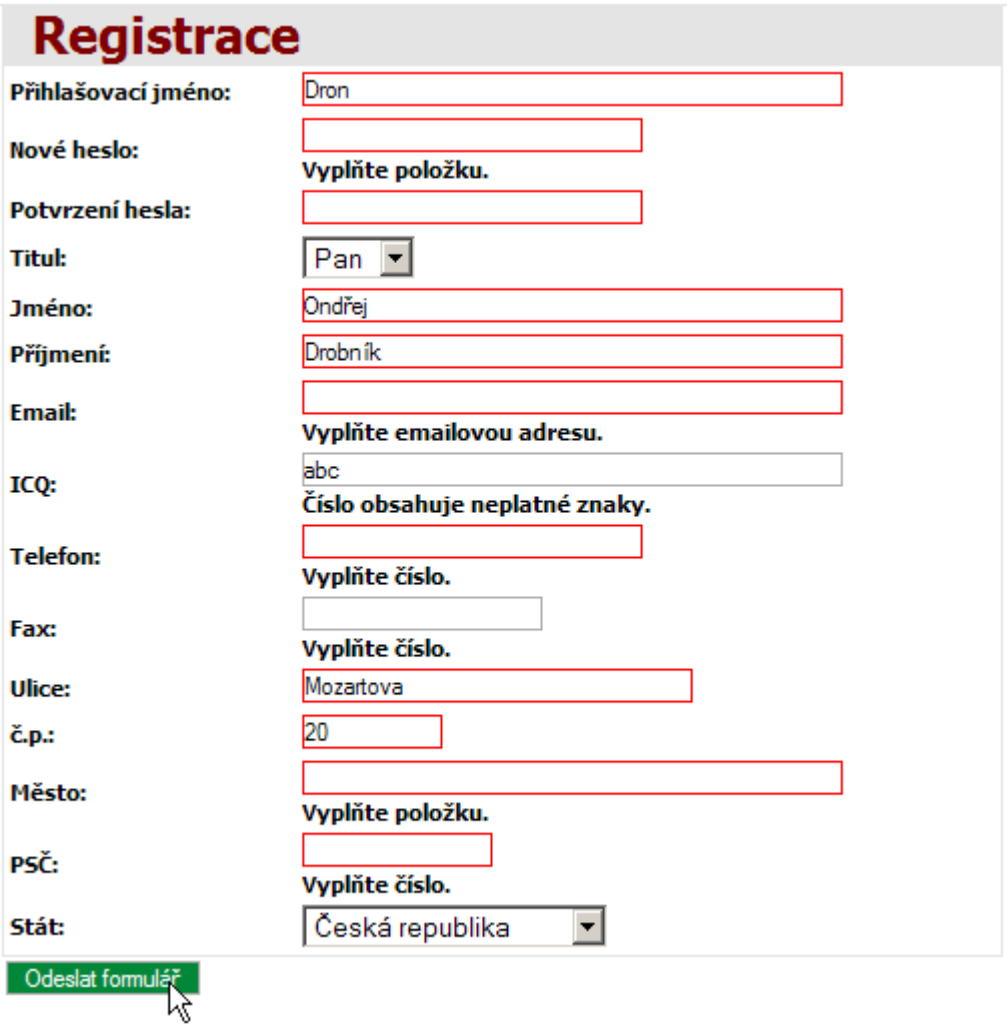
Při přihlášení se klientovi rozšíří možnosti o nakupování produktů, kdy bez přihlášení si klient může produkty pouze přidávat do košíku a procházet košíkem, po přihlášení se mu povolí možnost udělat nákup. Box přihlášení uživatele je těsně svázán s objektem zákazník, který obsahuje informace o klientovi (zda je přihlášen do systému internetového obchodu nebo ne, jméno, příjmení, titul, město, psč ,). Vlastnost prihlasen objektu zakaznik je definována jako protected, pro možnou modifikaci pouze z těla třídy zákazník, což umožňuje bezpečné zacházení s proměnou sloužící k přihlášení zákazníka. Objekt zakaznik je implementován jako Singleton tzn. pro běh skriptu existuje pouze v jedné instanci.

Samotné testování správnosti zadaných údajů probíhá technologií ajax. Tato technologie umožňuje měnit části HTML stránky, provádět určité operace nebo posílat určitá data bez nutnosti načtení celé stránky znovu. Princip se zakládá v tomto případě na události onclick odesílacího tlačítka přihlašovacího boxu. Událost onclick ukazuje na javascript funkci do souboru function.js. Ve funkci, konkrétně v tomto případě ve funkci prihlasZazaznika(), vyvářím nový ajax objekt, kterému předáváme uživatelské jméno, heslo a jméno PHP funkce, kterou má zavolat (call) a provést její obsah. Funkce pro volání ajax objektem jsou umístěny v souboru direct\js\ajax\index.php. Pro testování správnosti zadaného přihlašovacího jména a hesla je v PHP funkci proveden SQL dotaz na počet zákazníků s daným přihlašovacím jménem a heslem. Heslo je zakódováno algoritmem md5. Provedený SQL dotaz vrací počet zákazníků s daným přihlašovacím jménem a heslem. Pokud je počet jedna, funkce vrací řetězec ok a nastavuje vlastnost prihlasen objektu zakaznik na pravdu (true) a nastavuje zákazníkovi

kontaktní údaje z výsledků provedeného dotazu. Pokud je počet různý od 1, funkce vrací řetězec chyba. Tato návratová hodnota (ok nebo chyba) je poté analyzována v javascript funkci a podle ní je v případě úspěšného přihlášení uživatel přesměrován na domovskou stránku a je přihlášen, nebo v případě neúspěšného přihlášení se nestane nic. Po přihlášení dostává samozřejmě klient možnost odhlásit se. Odhlášení spočívá v nastavení protected vlastnosti prihlasen objektu zakaznik hodnotu false a znovunačtení stránky. Box přihlášení je implementován v krabici PravyBox.class.php, která seskupuje do jedné třídy přihlašovací box, box košíku, box filtry a box na vyhledávání.

3.9 Registrace a změna kontaktu

Pokud klient ještě nemá svůj účet v našem obchodě (ještě se neregistroval), může si svůj účet zařídit kliknutím na odkaz registrace v menu nebo v přihlašovacím boxu. Po kliknutí na jeden z těchto dvou odkazů dojde k přesměrování do registračního formuláře. Grafickou podobu registračního formuláře demonstruje obrázek 3-6.



Obr. 3-6 – Registrační formulář

Registrační formulář obsahuje vstupní pole, které jsou důležité pro získání účtu a následné zaslání objednávky, sem patří: jméno a příjmení, emailová adresa, telefon, ulice, číslo popisné, město, poštovní směrovací číslo a stát, tyto pole jsou označeny červeným okrajem a je bezpodmínečně nutné je při registraci vyplnit. Další pole ve formuláři jsou pro registraci nepotřebné, záleží pouze na zákazníkovi, zda pole vyplní či ne, jedná se o pole icq a fax.

Klient pro úspěšnou registraci musí dodržet určitá pravidla. Jedná se správný formát údajů zapsaných do formuláře. Při kontrole každé pole prochází vlastní validací zadaného údaje. V souboru společných funkcí (common/common.func.php) jsou implementovány funkce zajišťující validaci znakového řetězce s diakritikou i bez diakritiky, validaci obecného čísla, validaci emailové adresy a validaci telefonního čísla.

Základem validačních funkcí jsou regulární výrazy, kdy testujeme přítomnost znaků v testovaném řetězci na daných pozicích. Funkce na validaci také kontrolují maximální nebo minimální délku, kontrolují prázdnost řetězce atd.. Na základě těchto parametrů buď daný řetězec funkcí projde bez chyby (funkce vrací true a hodnota proměnné chyba vrácená odkazem není nastavena) a nebo řetězec neprojde kontrolou a funkce do hodnoty vracené odkazem chyba zapíše, o jakou chybu se jedná a funkce vrací false. Výstupní pravdivostní hodnotu z funkce true nebo false přidáváme operací AND do jedné proměnné, která má výchozí hodnotu true. Pokud po validaci má tato proměnná hodnotu true, tak všechny validace proběhly v pořádku a můžeme přidat nového klienta do databáze. Validace záznamů opět probíhá technologií AJAX tzn., že nenačítáme celou stránku znovu, pouze v případě výskytu chyb ve formuláři měníme chybové hlášky u jednotlivých elementů.

Registrace a kontakt zákazníka jsou implementovány v krabici krabice/DetailZakaznik.class.php.

Pokud se jedná o změnu kontaktních údajů klienta, pravidla jsou nastavena stejně jako v případě registrace. Při změně kontaktních údajů dostává klient ještě možnost změny hesla. Pro změnu hesla nejdříve musí zadat heslo staré a potom napsat dvakrát (podruhé pro potvrzení) nové heslo. Při změně kontaktních údajů, dochází ke změně (update) záznamu v databázi, při registraci je vložen záznam nový (insert).

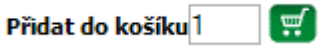
3.10 Košík

3.10.1 Třída košík

Základem nákupního košíku je třída kosik implementovaná v souboru common/kosik.class.php. Třída kosik má vlastnosti celková cena produktů v košíku, počet produktů v košíku, celkové recyklační poplatky za produkty v košíku, cena za dopravu objednávky, způsob platby za objednávku atd.. Ve třídě kosik existuje asociativní pole položky, které reprezentuje jednotlivé položky košíku. Třída kosik je pro bezkonfliktní běh skriptu provedena jako Singleton.

3.10.2 Přidání produktu do košíku

Samotné přidávání produktů do košíku je možné na každé stránce, kde je dostupná možnost přidat do košíku viz. obrázek 3-7. Tato možnost je na stránkách s prohlížením produktů, jako je stránka domovská, stránky sekcí vybraných podle stromu sekcí, stránka vyhledávání po úspěšném nalezení aspoň jednoho produktu nebo je tato možnost přidat produkt do košíku v detailu produktu.



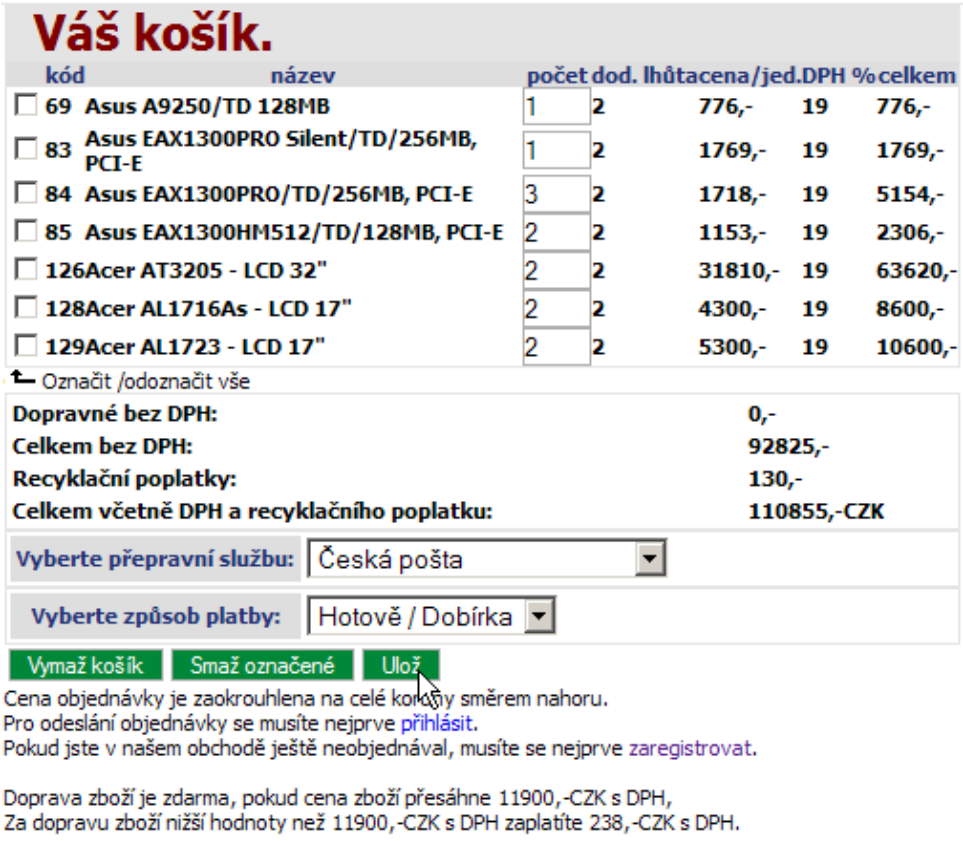
Obr. 3-7 – Přidání produktu do košíku.

Přidávání produktů do košíku je prováděno ajax technologií. Výhoda je opět v nenutnosti načítat celou stránku a tak obtěžovat klienta čekáním na načtení stránky a zatěžovat server zbytečnými požadavky. Princip spočívá v nastavení události onclick na tlačítko přidat do košíku. Po kliknutí na tlačítko přidat do košíku se zavolá funkce javascriptu v souboru function.js, která vytvoří AJAX objekt a předá mu parametry jméno funkce, identifikační číslo produktu a kvantitu, kterou si klient přeje do košíku přidat. Ajax objekt potom zavolá PHP funkci v souboru direct/js/ajax/index.php pod zadaným jménem, které jsme předali ajax objektu jako parametr. Funkce v našem případě funkce PridejProdukt() spustí SQL proceduru udp_vratCenuSklademRecyklaci, kdy proceduře předáváme identifikační číslo produktu. Zpět z této procedury dostáváme cenu, počet kusů daného produktu na skladě a recyklační poplatky. Nyní procházíme asociativní pole polozky objektu kosik a zjišťujeme, zda už daný produkt v košíku existuje nebo ne. Testování probíhá podmínkou jestli existuje klíč v poli if (array_key_exists($_id, $kosik->polozky)), kdy $_id je identifikační číslo produktu z databáze a $kosik->polozky je asociativní pole položek košíku. Pokud nalezneme položku v košíku, přidáme kvantitu k už existující položce a dále testujeme, zda nová kvantita produktu v košíku nepřesáhla počet produktů na skladě. Při překročení počtu produktů na skladě se kvantita produktu nastaví na počet produktu na skladě. Pokud položku v košíku nenalezneme, přidáme do pole novou položku a testujeme pouze chtěnou kvantitu s počtem položek na skladě. Podle rozdílu nový počet položek v košíku mínus starý počet položek v košíku na základě ceny a recyklačních poplatků vypočteme novou celkovou cenu položek v košíku a celkové recyklační poplatky za

položky v košíku. Po vypočítání všech těchto cen PHP funkce vrací hodnoty počet produktů v košíku, celková cena za položky košíku a odkazy na prohlížení košíku, do funkce javascriptu v souboru direct/function.js. Zde se přijatý řetězec (responseText) funkcí split rozdělí podle oddělovače na položky pole. Z tohoto pole se potom vytvoří nová podoba boxu košík, kdy se v boxu objeví text obsahující nový počet produktů v košíku a nová celková cenu košíku.

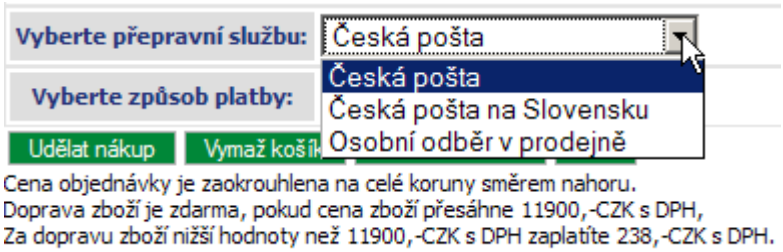
3.10.3 Náhled košíku

Náhled košíku podává celkový obraz o stavu klientova košíku viz. obr. 3-8.. V náhledu jsou zobrazeny všechny položky košíku v přehledné struktuře. Je vidět u všech položek košíku název, jednotková cena, recyklační poplatky, dodací lhůta, počet objednaných kusů a cena celková za produkt. V náhledu je také zobrazena cena za dopravu, celkové recyklační poplatky za objednávku a výsledná celková cena s daní a recyklačními poplatky.



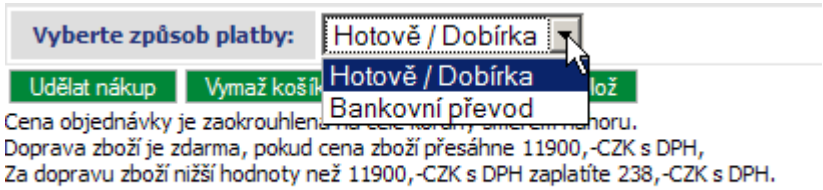
Obr. 3-8 – Náhled košíku.

V náhledu má klient možnost měnit způsob dopravy viz. obrázek 3-9. Klient si může vybrat mezi třemi způsoby dopravy zásilky. Na výběr přepravní služby je zde Česká pošta a její doprava po České republice nebo na Slovensko, kdy doprava po ČR je za 200,-Kč bez DPH a na Slovensko 500,-Kč bez DPH pokud nepřesáhne cena objednávky 10000,- Kč bez DPH. Při objednávce nad uvedený limit nebo při osobním odběru v prodejně je doprava zdarma.



Obr. 3-9 – Změna přepravní služby.

Klient dostává v náhledu košíku také možnost vybrat si způsob platby viz obr. 3-10. Žádný způsob ptatby není zvýhodněn.



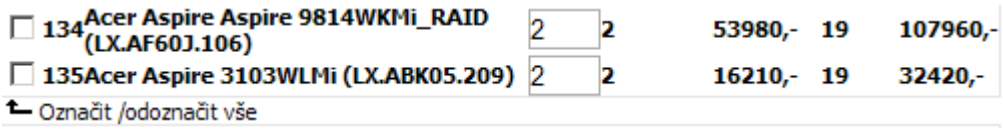
Obr. 3-10 – Změna způsobu platby.

Klient v náhledu košíku dostává možnost nakoupit vybrané zboží, měnit kvantitu položek v košíku, mazat položky košíku nebo smazat košík celý. U každého produktu je vstupní textové pole, nazvané počet, s kvantitou produktu v košíku, které může klient editovat a po kliknutí na tlačítko ulož, ve spodní části náhledu košíku, se celý košík přepočítá a ukáží se nové ceny a kvantity položek. V náhledu košíku má každý produkt své vlastní zaškrtávací políčko, které se dá využít při požadavku na smazání položky z košíku.

Samotná akce změna kvantity produktu v košíku je podobná akci přidání produktu do košíku. Smazání celého košíku je v principu jednoduché smazání pole

položek a vynulování vlastností objektu kosik.

Pokud chceme vymazat pouze některé položky z košíku, označíme zaškrtávací pole u daného produktu a klikneme na tlačítko „Smaž označené" ve spodní části náhledu košíku. Při označování zaškrtávacího políčka u jednotlivých položek v náhledu, se provádí funkce na událost onclick. Tato funkce vytvoří ajax objekt, který zapíše do pole zaškrtnuto objektu kosik pravdivostní stav zaškrtnuto zaškrtávacího pole. Potom když klikneme na tlačítko „Smaž označené" ve spodní části náhledu, script opět vytvoří ajax objekt, který zavolá funkci, ve které procházíme pole zaškrtnuto objektu kosik. Podle pravdivostní hodnoty v poli zaškrtnuto mažeme položky košíku. Pokud nastane stav, že v košíku nejsou žádné položky, funkce smaže celý košík a přesměruje na domovskou stránku. V náhledu jsou také pomocná tlačítka označ vše a odoznač vše, umístěná pod seznamem produktů v košíku, která pomáhají při označování položek v košíku, viz. obr 3-11.



Obr. 3-11 – pomocná tlačítka označ vše a odoznač vše.

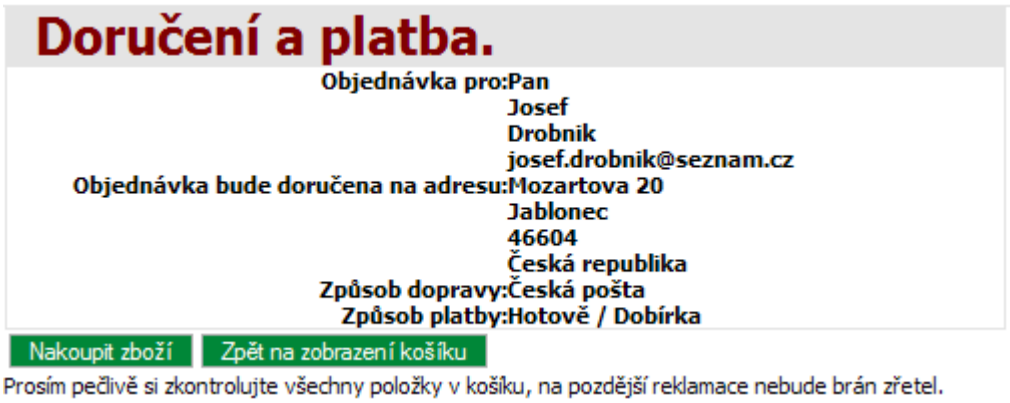
3.10.4 Shrnutí košíku

Shrnutí košíku je stránka, kde jsou vidět jednotlivé položky v košíku už bez možnosti editace viz obr.3-12. Stránka je tvořena stejně jako stránka s náhledem košíku, až na rozdíl, že neobsahuje editační prvky. Je to předposlední stránka před finálním vytvořením objednávky. Je zde odkaz zpět, kterým se lze dostat na náhled košíku a odkaz přejít k doručení a platbě, kterým se dostaneme na stránku s doručením a platbou, kde jsou shrnuty informace o klientovi (jméno, příjmení, email, adresa atd) a způsobu platby a dopravy.

Obr. 3-12 –Shrnutí objednávky.

3.10.5 Doručení a platba

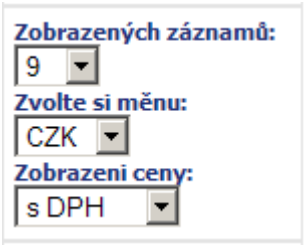
Tato stránka obsahuje info o klientovi a detaily o vybraném způsobu platby a dopravy viz. Obr 3-13. Tyto údaje jsou pouhým vypsáním vlastností objektu zakaznik a objektu kosik. Z této stránky je možné se vrátit zpět na náhled stránky (tlačítko zpět) nebo je zde možnost nakoupit zboží. Po kliknutí na tento odkaz je provedena objednávka a poslán klientovi informační email s informacemi o objednávce. Všechny položky košíku a informace o nakupujícím klientovi jsou uloženy do databáze konkrétně do tabulek objednavka2produkt a objednávky. Objednavka2produkt obsahuje propojení objednávky s produkty, kdy v tabulce je zaznamenáno u každé objednávky jaké produkty obsahuje a v jakém množství. Takulka objednávky obsahuje identifikační číslo objednávky, identifikační číslo klienta, datum provedení objednávky, celková cena objednávky a poznámky k objednávce.



Obr. 3-13 –Doručení a platba

3.11 Filtry zobrazení

Filtry pro zobrazení jsou implementovány v krabici PravyBox.class.php, která seskupuje do jedné třídy přihlašovací box, box košíku, box filtry a box na vyhledávání. Jedná se o filtry počtu zobrazených záznamů na stránce, filtr zobrazované měny a filtr zobrazované ceny bez a nebo s daní z přidané hodnoty viz obr. 3-14.



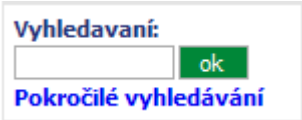
Obr. 3-14 – Filtry zobrazení

Filtr na počet zobrazených záznamů nastavuje počet zobrazených záznamů na stránce po události onchange výběrového (<select>) elementu. Při této události nastane přesměrování na stejnou stránku s parametrem limit, který je nastaven na námi zvolenou hodnotu. Při načítání stránky se parametr limit zapíše do vlastnosti limit objektu parametry a stránky ho dále používají jako své výchozí nastavení. Stejný sled událostí nastává i při nastavování zobrazované měny i při nastavování zobrazení ceny s daní z přidané hodnoty nebo bez. Nastavování těchto tří parametrů stránky neprobíhá technologií ajax kvůli velkému počtu závislostí na stránce. Plnění výběrových (<select>) elementů probíhá z polí definovaných v objektu parametry. Konkrétně jde o pole mozneMeny, kdy procházíme pole a jeho položkami plníme jednotlivé volby (<option>) ve výběrovém elementu. Pokud při plnění nalezneme právě platnou hodnotu pro stránku, nastavíme volbu jako selected, což způsobí předání focus této volbě a tím její vybrání.

3.12 Fulltextové vyhledávání

V aplikaci jsou implementovány dvě možnosti vyhledávání produktů a to vyhledávání základní, přímé napsání hledané fráze do textového pole, a nebo pokročilé vyhledávání produktů podle více parametrů, např. podle rozmezí ceny a sekce atd.. Panel základní vyhledávání je dostupný z jakékoliv stránky v obchodě viz. obr. 3-15.

Fulltextové vyhledávání se jmenuje fulltextové, protože hledání probíhá ve všech textových položkách produktů, což znamená hledání v názvu, popisu a výrobci produktu.

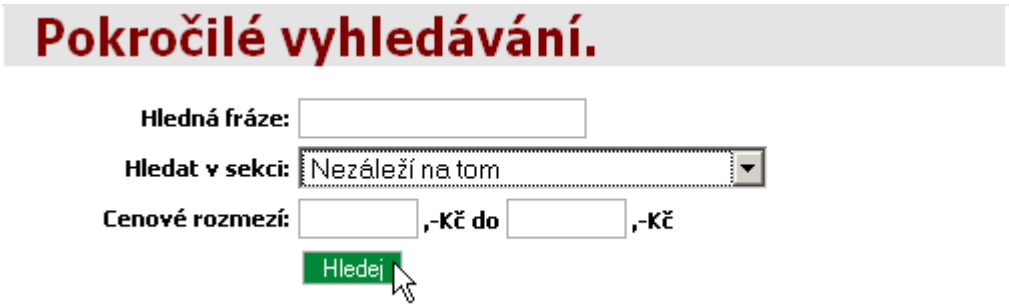


Obr. 3-15 – Box základní vyhledávání-

Box jednoduché vyhledávání obsahuje vstupní pole pro hledanou frázi, tlačítko pro hledání a odkaz na pokročilé vyhledávání. Princip vyhledávání se zakládá na tabulkách fulltextSlova a fulltext2produkt. Tabulka fulltextSlova obsahuje unikátní slova a jejich identifikační čísla. Tabulka fulltext2produkt obsahuje identifikační číslo slova z fulltextSlova tabulky a identifikační číslo produktu, u kterého se toto slovo nachází. Tvorba těchto tabulek je popsána v části administrátorské aplikace tvorba fulltextové tabulky. Vyhledání fráze probíhá po kliknutí na tlačítko "ok" v boxu vyhledávání, kdy je volána javascript funkce hledejFrazi(), ve které se testuje prázdnost řetězce. Pokud je řetězec prázdný, funkce nevykoná nic a stránka zůstane nezměněna. Pokud zákazník zadal do vstupního textového pole frázi, dochází k přesměrování do skriptu index.php s parametrem sekce rovným řetězci vyhledavani a s parametrem fraze rovným klientově zadané frázi. Dále ve skriptu probíhá testování, zda jde o pokročilé vyhledávání nebo o jednoduché vyhledávání, to se děje na základě dodaných parametrů, pokročilé vyhledávání má navíc oproti jednoduchému parametry cena od a cena do a hledáni v sekcích viz dále. Při hledání je fráze vložena do SQL dotazu a ohraničena znaky "%", pro možnost, kdy je fráze pouze část slova. SQL dotaz potom vybírá všechny produkty z tabulky produkt, kde se shodují identifikační čísla produktů s identifikačními čísly produktů v tabulce fulltext2produkt, a v tabulce fulltextSlova existuje slovo, které osahuje danou frázi a identifikační číslo produktu je id produktu v tabulce Produkt.

Pokročilé vyhledávání má oproti jednoduchému přidány možnosti vyhledávání v sekcích a vyhledávání podle rozmezí ceny viz. obr 3-16. Pokud klient nevyplní cenové rozmezí a ve výběru hledat v sekci nechá výchozí hodnotu "nezáleží natom", chová se vyhledávání jako vyhledávání jednoduché. Princip hledání podle

pokročilejších parametrů je jen otázkou přidání spojení s tabulkou Sekce v SQL dotazu a podmínky, aby cena hledaného produktu byla větší než cena od a menší než cena do.



Obr. 3-16 – Pokročilé vyhledávání

V nalezených záznamech lze jednoduše listovat a třídit, stejně jako při prohlížení produktů v jiných sekcích.

3.13 Detail produktu

3.13.1 Detail produktu

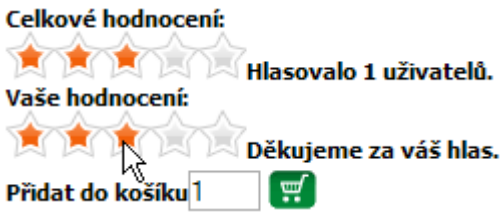
Detail produktu je implementován v krabici DetailProdukt.class.php. Jde o přehledné zobrazení všech informací o produktu, jako je jeho název, cena, záruční doba, recyklační poplatky, výrobce, popis atd. viz obr. 3-17.

V krabici detailu produktu jsou ve funkci vystup() z databáze načteny informace o produktu a k těmto informacím je přidáno hodnocení produktu a diskuze. U hodnocení má zákazník možnost jednak sám hlasovat (hvězdy pod nadpisem Vaše hodnocení), a nebo pouze shlédnout, jak hlasovali jiní uživatelé o tomto produktu (celkové hodnocení). Klient má možnost diskutovat o tomto produktu prostřednictvím diskuze.

Obr. 3-17 – Detail produktu.

3.13.2 Hodnocení

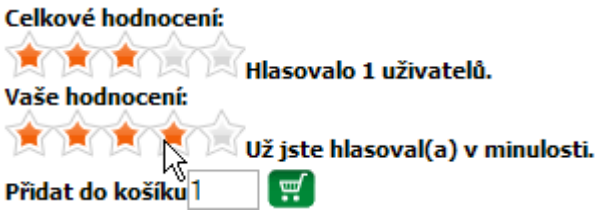
Hlasování o kvalitě produktu probíhá technologií ajax. Zákazník může ze své dosavadní zkušenosti, nebo z posouzení parametrů produktu usoudit, jak kvalitní produkt je, a přisoudit mu jedno z hodnocení, které nabízí hodnocení produktu. Hodnocení jsou, produkt je tragický, špatný, dobrý, ucházející a nebo výborný. Každému z těchto hodnocení je přisouzena známka (počet hvězdiček) a to od tragického, který má hvězdu pouze jednu, do produktu výborného, který má hvězd pět. Toho hodnocení se ihned po hodnocení promítá do celkového hodnocení (hvězdy pod nadpisem Celkové hodnocení) a je poděkováno zákazníkovi za jeho hlasování o produktu viz. obr. 3-18.



Obr. 3-18 – Hodnocení produktu.

Na hodnocení produktu je v databázi internetového obchodu vyčleněna tabulka

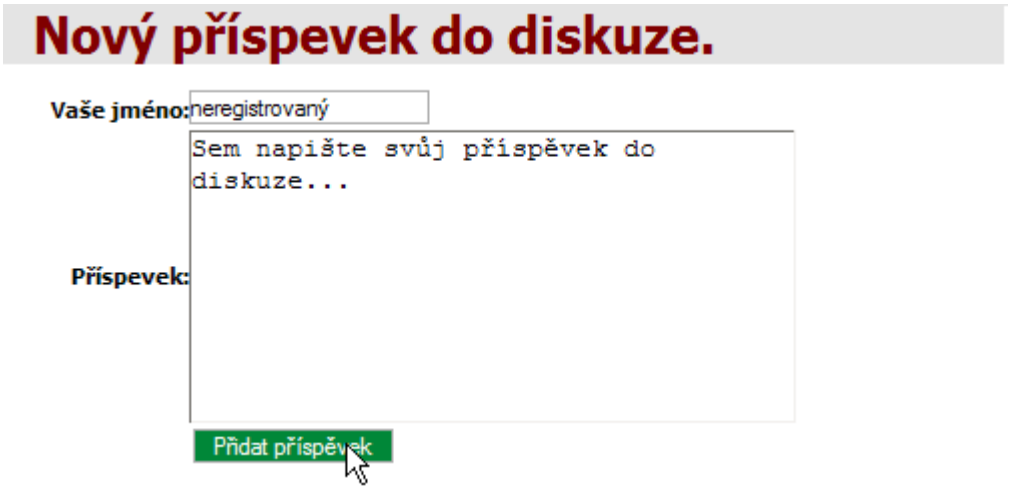
Hodnoceni, a sloupce pocetHodnoticich, sumaHodnoceni a hodnoceni v tabulce Produkt. V tabulce Hodnoceni jsou uchovávána data o hodnotícím zákazníkovi (zakaznikId) a k jakému produktu se hodnocení vztahuje (produktId). U zákazníka zaznamenáváme jeho identifikační číslo, pokud je přihlášen, tak jeho identifikační číslo přidělené v databázi a pokud jde o neregistrovaného klienta, tak je identifikační číslo rovné -1. Další uchovávaná data jsou datum hodnocení, proxy server a IP adresa ve formátu čísla, převedená funkcí ip2long a získaná z hodnoty pole $_SERVER['REMOTE_ADDR']. Hodnoty IP adresy a proxy serveru jsou testovány při novém pokusu klienta hlasovat o stejném produktu, kdy se klientovi zobrazí „Už jste hlasoval“ a jeho nový hlas není již započítán do celkového hodnocení produktu viz. obr. 3-19.



Obr. 3-15 – Situace při pokusu opět hodnotit

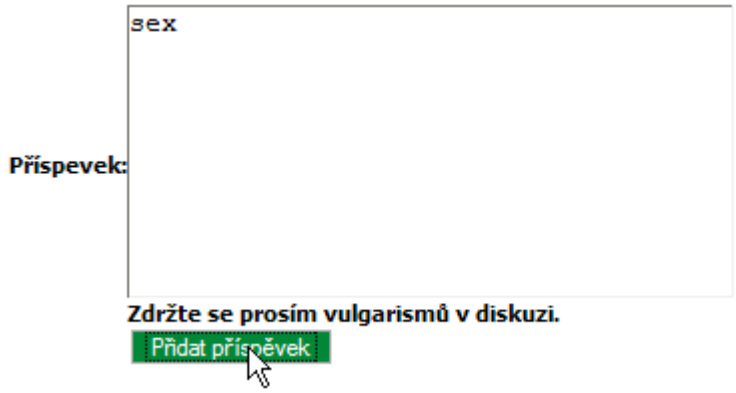
3.13.3 Diskuze produktu

O výpis diskuze produktu se stará funkce diskuzeProduktu(), implementovaná v krabici detailu produktu. Jde o výpis z tabulky diskuze, kde si uchováváme jméno autora příspěvku, text příspěvku, datum příspěvku a identifikační číslo produktu, ke kterému se diskutovaný příspěvek vztahuje. Pod příspěvky diskuze je tlačítko přidat příspěvek, které nás přesměruje na stránku s přidáním příspěvku k produktu viz. obr 3-20.



Obr. 3-20 – Přidání nového příspěvku do diskuze

Formulář pro přidání nového příspěvku do diskuze, obsahuje textové vstupní pole jméno, které pokud je zákazník přihlášen se automaticky vyplní jeho přihlašovacím jménem, a pokud zákazník přihlášen není, je vyplněn text neregistrovaný. Toto pole si může zákazník pro zachování své anonymity změnit. Pole na přidání příspěvku „příspěvek" je vstupní pole pro nový příspěvek do diskuze. Ve spodní části formuláře je tlačítko pro přidání příspěvku do diskuze. Přidání příspěvku je ošetřeno jednoduchým filtrem vulgarit. Filtr funguje po kliknutí na tlačítko přidat příspěvek, kdy se spustí funkce přiřazená k události onclick tohoto tlačítka a v této funkci se vytvoří ajax object. Ten pak v php kódu volá funkci filtrVyrazu() s parametrem text příspěvku. Ve funkci filtrVyrazu() je seznam vulgarit nepřípustných v diskutovaném příspěvku. Celý text příspěvku se porovnává s každým výrazem z pole vulgarit a v případě nalezení výrazu v příspěvku je navrácena chyba do formuláře přidání příspěvku viz. obr 3-21.

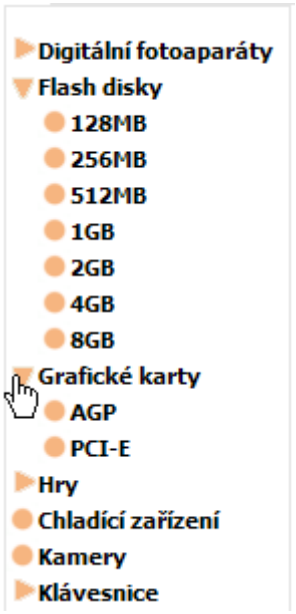


Obr. 3-21 – Filtrování nevhodných slov pro diskuzi.

Když filtr vulgarismů nenalezne v novém příspěvku žádný z nedovolených výrazů, příspěvek je přidán do databáze a zobrazován v diskuzi u daného produktu.

3.14 Strom sekcí

Strom sekcí slouží pro přehledné zobrazení závislostí sekcí v internetovém obchodě viz. obr 3-22.



Obr. 3-22 – Strom sekcí

Strom je implementován v souboru krabice/Tree.class.php a jeho struktura je načítána ze souboru direct/menu.txt. V souboru menu.txt je strom definován co řádek to jedna sekce. Na začátku každého řádku jsou tečky, určují jak hluboko je sekce ve stromu umístěna např. jedna tečka značí sekci první úrovně (sekce, která nemá předchůdce), sekce se dvěma tečkami na začátku jsou závislé na předešlé sekci s jednou tečkou atd. Po tečkách je zobrazovaný název sekce a po názvu, oddělném svislítkem, následuje odkaz na sekci. V případě sekcí první úrovně je odkaz složen pouze z odkazu na index.php a parametru identifikační číslo sekce např.. Digitální fotoaparáty|index.php?sekceId=1 a v případech sekcí druhé a vyšší úrovně je v odkazu parametr identifikační číslo předka např. ..3Mpix|index.php?sekceId=2&parentId=1. V souboru krabice krabice/Tree.class.php pak dochází k parsování textového souboru se stromem a tvoření přehledného grafického výstupu. Při kliknutí dochází k rozbalení (expanzi) stromu a mění se obrázky u jednotlivých sekcí, pokud je sekce konečná

(žádná další sekce tuto sekci nemá jako svého předka), je u ní zobrazen obrázek kruhu. Pokud sekce není konečná, má u sebe sekce znak trojuhelník a ten značí možnost rozbalení. Pokud k rozbalení sekce neposlední úrovně dojde, trojúhelník změní orientaci viz. obr 3-23.

Obr. 3-17 – Zleva, sekce nerozbalená, rozbalená, sekce poslední úrovně

3.15 Prohlížení produktů

Prohlížení produktů je implementováno v krabici SeznamProduktu. V krabici je nejdříve testováno, zda jsou platné parametry sekceId a parentId, které značí určitou sekci. ParentId a sekceId jsou sloupce v tabulce Sekce v databázi internetového obchodu. Pokud jsou tyto parametry platné, jsou zakomponovány do SQL dotazu a do vykonání dotazu dostáváme zpět produkty a jejich informace z dané sekce. Pokud parametry platné nejsou, zobrazuje se stránka domovská se všemi produkty.

Při prohlížení produktů, má klient možnost si tyto produkty řadit dle svých kritérií a listovat ve stránkách, pokud je produktů v sekci více něž je limit pro zobrazení na stránku. O řazení se stará metoda razeniListu(), které jsou předávány parametry titulek řazení, sloupec v databázi podle kterého chceme záznami řadit a směr řazení. O listování ve stránkách se stará třída PageLister umístěná v adresáři common/class. Má implementovány metody next(), previous(), current(), showListing(), které se starají o procházení v záznamech a jejich přehledné zobrazení.

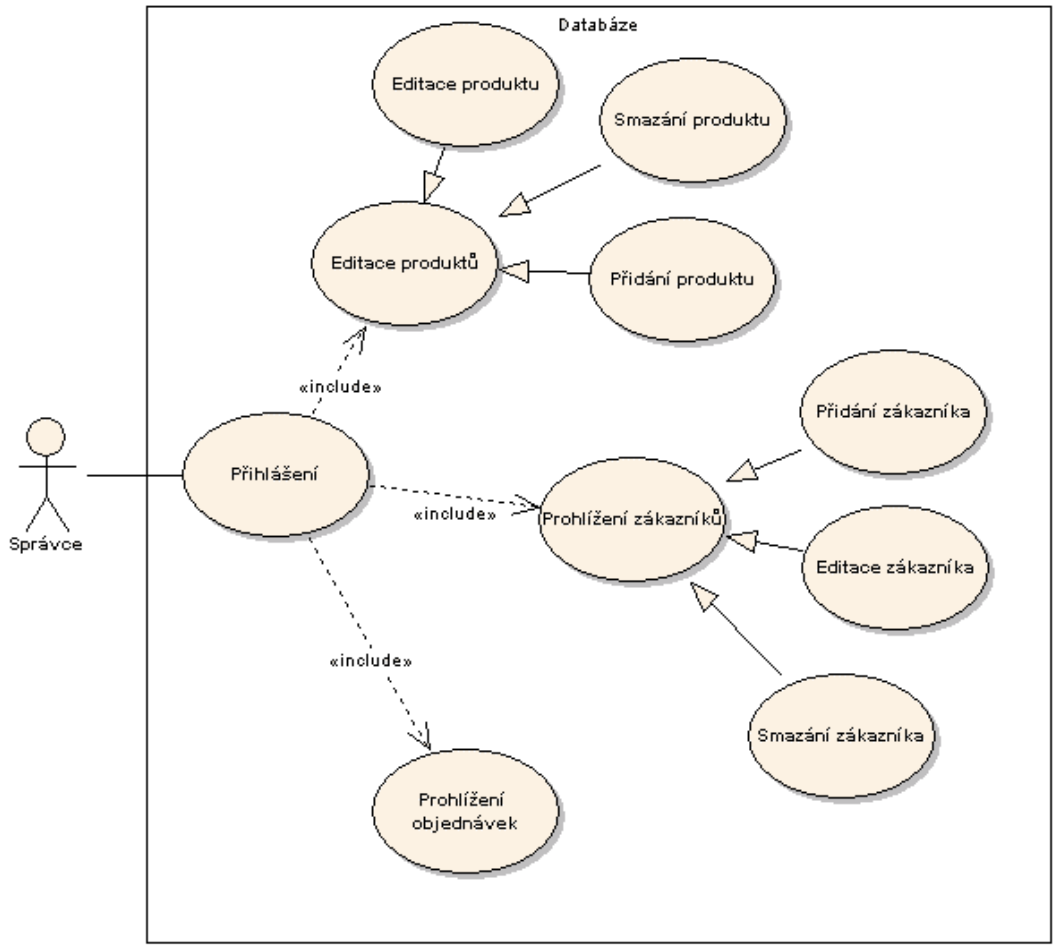
4.Správcovská aplikace

4.1 Úvod

Pro správu internetového obchodu slouží aplikace spravující zákazníky, produkty a objednávky tohoto internetového obchodu.

Do správcovské aplikace mají přístup pouze uživatelé, kteří jsou vedeni jako admin, a to po zadání svého přihlašovacího hesla a jména.

Přehled funkcí správcovské aplikace viz obr 4-1.



Obr. 4-1 – UML use case diagram správcovské aplikace

Struktura správcovské aplikace je úplně stejná jako struktura internetového obchodu. Opět základ tvoří návrhový vzor tovární metoda a jednotlivé objekty jsou Singletony. Na stejné struktuře je vidět dobrá modifikovatelnost základu obou aplikací.

4.2 Souborová a adresářová struktura

Základní element programového kódu správcovské aplikace je opět krabice. Struktura souborů a adresářů je v obou aplikacích stejná. Krabice jsou opět koncipovány tak, aby se chovaly jako autonomní části zdrojového kódu. Přehled krabic správcovské aplikace viz tab. 4-1.

jméno krabice (*.class.php)	funkce
HtmlFooter	Patička HTML stránky
HtmlHeader	Hlavička HTML stránky
Lista	Zobrazení listy s odkazy
Login	Přihlášení do systému
Ostatni	Ostatní nastavení v obchodě
PrazdnaKrabice	Předek všech krabic
PridatProdukt	Přidání nového produktu do databáze.
SeznamObjednavek	Procházení objednávkami.
SeznamProduktu	Procházení produkty
SeznamZakazniku	Procházení a editace zákazníků

Tab. 4-1 – Přehled krabic a jejich funkce

4.3 Přihlášení

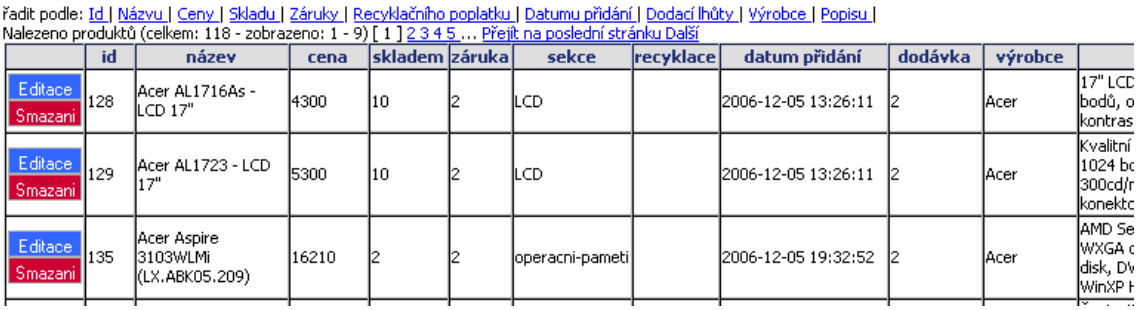
Při vstupu do správcovské aplikace se objeví přihlašovací okno viz obr 4-2. Správce zde musí pro svoje přihlášení do aplikace zadat přihlašovací jméno a heslo.

Obr. 4-2 – přihlášení do správcovské aplikace

Přihlášní správce do správcovské aplikace je založeno na testování zadaného přihlašovacího jména a hesla s údaji v databázi. Údaj hesla je zakódován algoritmem md5. U přihlašovacích údajů v databázi je také testováno zda je nastaven sloupec admin tabulky zakaznici na 1, což značí, že jde o správce aplikace.

4.4 Prohlížení produktů

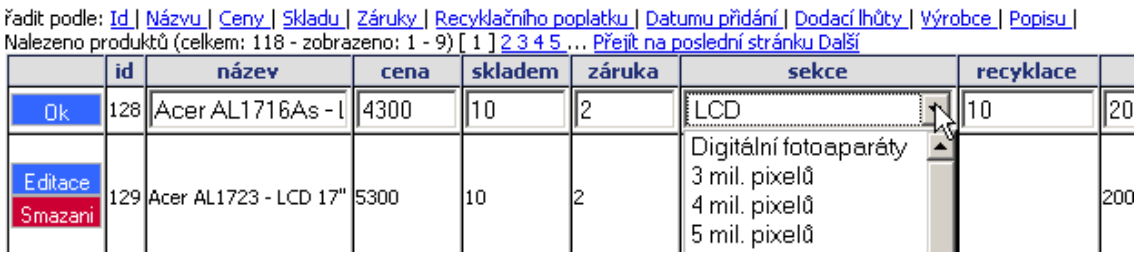
Do prohlížení produktů se může správce dotat kliknutím na odkaz seznam produktů v menu správcovské aplikace. Po kliknutí na tento odkaz dojde k načtení informací z databáze o produktech internetového obchodu viz. obr 4-3. Informace o produktech jsou načítány z tabulek produkt a sekce databáze internetového obchodu.


řadit podle: Id | Názvu | Ceny | Skladu | Záruky | Recyklačního poplatku | Datumu přidání | Dodací lhůty | Výrobce | Popisu |
Nalezeno produktů (celkem: 118 - zobrazeno: 1 - 9) [1] 2 3 4 5 ... Přejít na poslední stránku Další

	id	název	cena	skladem	záruka	sekce	recyklace	datum přidání	dodávka	výrobce	
Editace Smazani	128	Acer AL1716As - LCD 17"	4300	10	2	LCD		2006-12-05 13:26:11	2	Acer	17" LCD bodů, o kontras
Editace Smazani	129	Acer AL1723 - LCD 17"	5300	10	2	LCD		2006-12-05 13:26:11	2	Acer	Kvalitní 1024 bc 300cd/r konektc
Editace Smazani	135	Acer Aspire 3103WLMi (LX.ABK05.209)	16210	2	2	operacni-pameti		2006-12-05 19:32:52	2	Acer	AMD Se WXGA c disk, DV WinXP H

Obr. 4-3 – Zobrazené informace o produktech

Správce má na této stránce možnost záznamy řadit, listovat, editovat, mazat a přidávat nové. Řadit lze podle různých sloupců v databázi a listovat mezi jednotlivými stránkami zobrazených záznamů. Editace je dostupná po stisknutí tlačítka editace v nepojmenovaném slouci u záznamu. Po stisku tlačítka editace se zpřístupní textová pole jednotlivých sloupců pro editaci a místo tlačítek editace a smaž se objeví tlačítko ok pro potvrzení editace. V programovém kódu je událost onclick tlačítka editace ošetřena javascript funkcí, která vytvoří ajax objekt. Tento objekt poté volá php funkci, ve které se z databáze načtou záznamy o produktu, který chceme editovat. PHP funkce poté posílá jako návratovou hodnotu do javascript funkce získané informace o produktu oddělené oddělovačem tři zpětná lomítka. V javascript funkci dochází funkcí split k oddělení jednotlivých informací o produktu dle oddělovače a těmito informacemi jsou naplněny editační pole u editovaného záznamu viz. obr. 4-4.

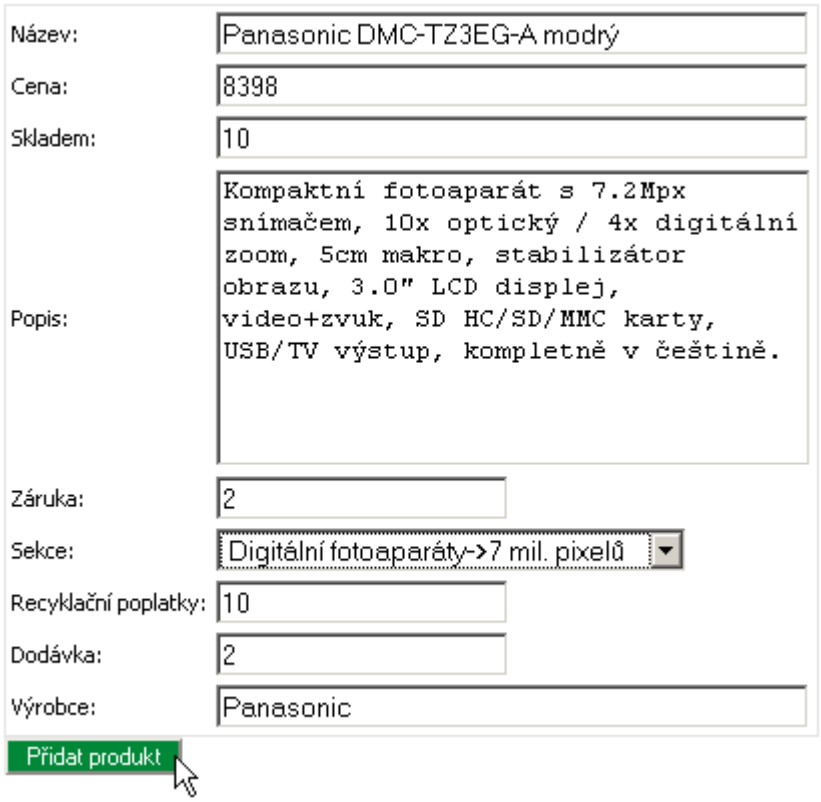


Obr. 4-4 – Editace produktu.

Po ukončení editace lze změny potvrdit stiskem tlačítka ok. Změny vejdou v platnost provedením update dotazu databáze s novými údaji produktu.

Smazání produktu probíhá po kliknutí na tlačítko smazani u záznamu. V kódu dochází opět v javascript funkci k vytvoření ajax objektu, který volá php funkci a tato funkce sql dotazem delete smaže příslušný zázam z databáze.

Správce může přidat nový produkt po stisku tlačítka „Přidat produkt". Po stisku se objeví formulář na přidání nového produktu viz. obr. 4-5.

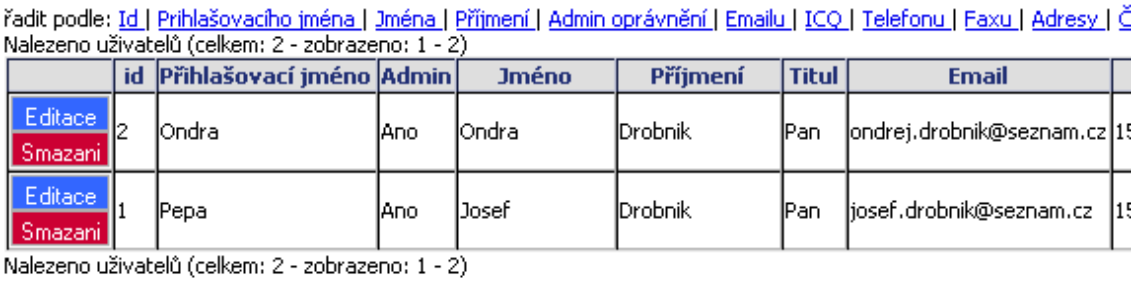


Obr. 4-5 – Formulář přidat nový produkt.

4.5 Prohlížení zákazníků

Do prohlížení zákazníků se může správce dostat kliknutím na odkaz seznam zákazníku v menu správcovské aplikace. Po kliknutí na tento odkaz dojde k načtení informací z databáze o stávajících zákaznících internetového obchodu viz. obr 4-6. Informace o zákaznících jsou načítány z tabulky zakaznici našeho internetového obchodu. Stránka zobrazení zákazníků je graficky a funkčně stejná jako stránka

s produkty, rozdíl je pouze v zobrazení jiných sloupců z databáze.


řadit podle: Id | Prihlašovacího jména | Jména | Příjmení | Admin oprávnění | Emailu | ICQ | Telefonu | Faxu | Adresy | Č
Nalezeno uživatelů (celkem: 2 - zobrazeno: 1 - 2)

	id	Přihlašovací jméno	Admin	Jméno	Příjmení	Titul	Email	
Editace Smazani	2	Ondra	Ano	Ondra	Drobnik	Pan	ondrej.drobnik@seznam.cz	1
Editace Smazani	1	Pepa	Ano	Josef	Drobnik	Pan	josef.drobnik@seznam.cz	1

Nalezeno uživatelů (celkem: 2 - zobrazeno: 1 - 2)

Obr. 4-6 – zobrazené informace o zákaznících

4.6 Ostatní nastavení

Do ostatního nastavení obchodu je zařazeno vytváření tabulek pro fulltextové vyhledávání.

Tvorba fulltextové tabulky je postavena na obsahu sloupců nazev, popis a vyrobce databázové tabulky Produkt. Ze záznamů těchto sloupců je tvořeno pole jednotlivých slov a pole jim odpovídajících identifikačních čísel produktů.

Poté jsou do tabulky FulltextSlova přidávány SQL dotazem INSERT jednotlivá slova a jim odpovídající identifikační čísla produktů. Sloupec slova má nastavený unique index, který zajistí, že při pokusu o přidání již existujícího slova se jako výsledek SQL dotazu vrací specifická chyba značící duplicitu slova a slovo není do tabulky přidáno. Pokud se nám duplicita slova objeví, identifikační číslo slova se zjišťuje dotazem na identifikační číslo daného slova. Pokud se uplicita neobjeví, stačí zjistit poslední přidaný index do tabulky a tím zjistit identifikační číslo nově přidaného slova.

Toto identifikační číslo slova poté zapíšeme s identifikačním číslem produktu do tabulky Fulltext2Produkt, která propojuje slova s produkty.

6. Závěr

Úkolem této diplomové práce bylo ověřit možnosti použití návrhových vzorů v objektově orientovaných webových aplikacích. Na základě zadání byly vytvořeny aplikace internetový obchod a přidružená správcovská aplikace tohoto obchodu. Na těchto dvou aplikacích je ukázána vhodnost použití návrhových vzorů a to především návrhových vzorů tovární metoda, singleton, strategie, prototyp a další. Nejvíce se uplatnil návrhový vzor tovární metoda jako základní stavební vzor obou aplikací. Na návrhovém vzoru tovární metoda je nejlépe ukázán jednoduchý způsob použití návrhových vzorů a jejich jednoduchá modifikovatelnost. Modifikovatelnost se zakládá na společném základě obou aplikací, ke kterému jsou přidávány samostané bloky kódu, v aplikacích nazývány „Krabice“, a které reprezentují jednotlivé autonomní objekty. Další návrhové vzory se uplatnily jako doplňky k aplikacím internetového obchodu a správcovské aplikace a opět se potvrdila jejich vhodnost k rychlejšímu řešení projektů, které jsou standardizované, nebo velmi podobné již projektům někdy v minulosti realizovaným.

V první části diplomové práce je provedeno rozdělení návrhových vzorů, které vychází z knihy GAM95, která je doposud nejuznávanější knihou v oblasti návrhových vzorů. Po rozdělení jsou popsány všechny návrhové vzory, ve stručnosti je nastíněm důvod vzniku jednotlivých návrhových vzorů a příklad jejich použití. Je také stručně popsáno, kde se jednotlivé návrhové vzory v aplikacích objevují.

V další části diplomové práce se popisují webové technologie vhodné k tvorbě objektově orientovaných webových aplikací. Základem pro vývoj aplikací zadaných v rámci této diplomové práce se staly technologie pod bezplatnými licencemi, jako jsou programovací jazyk PHP, databázový systém MySQL a server APACHE.

V poslední části se diplomová práce věnuje rozboru aplikací internetový obchod a správcovské aplikace

Tato práce dává dostatek podkladů, aby každý mohl posoudit smysluplnost návrhových vzorů. Jedním z poselství, které z většiny dobrých návrhových vzorů vyzařuje, je jejich jednoduchost. Ta může návrhářům dodat sílu potřebnou k zvládnutí složitých informačních systémů.

Literatura

[1] Andi Gutmans, Stig Saether Bakkenm, Derick Rethans: Mistrovství v PHP 5 Computer press, 2005, 655s. ISBN 80-251-0799-X.

[2] [GAM95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlisside: Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, říjen 1995,395s. ISBN 0-201-63361-2

[3] Page-Jones, Meilir. Základy objektově orientovaného návrhu v UML. Praha: Grada Publishing, 2001.389s. ISBN 80-247-0210-X.

[4] Schmuller, Joseph. Mysl9me v jazyku UML. Praha: Grada Publishing, 2001. 360s. ISBN 80-247-0029-8.

[5] Bráza, Jiří. PHP 5. Praha: Grada Publishing, 2005. 240s. ISBN 80-247-1146-X.

Elektronické zdroje

[6] http://www.enteract.com/~bradapp/docs/patterns-intro.html - Brad Appleton: Patterns and Software: Essential Concepts and Terminology.

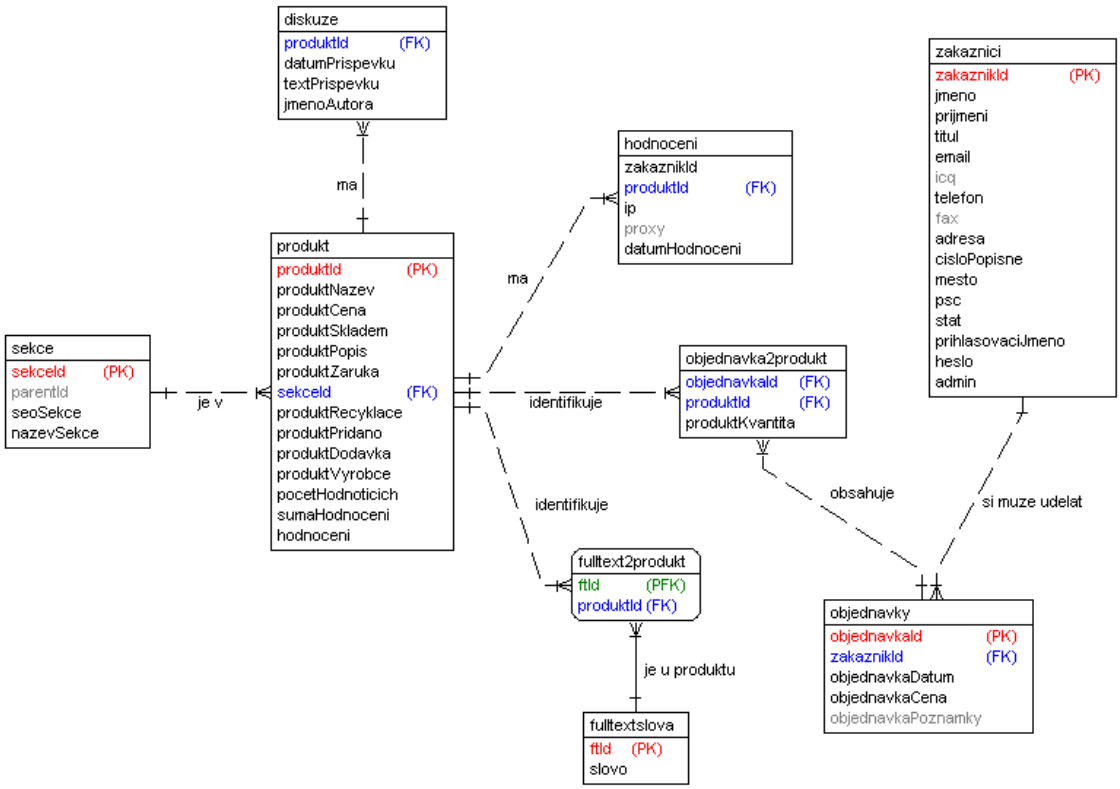
[7] http://www.cetus-links.org/oo_patterns.html

[8] http://www.objects.cz/produkty/Produkty.html

[9] http://objekty.vse.cz/

Přílohy

A - ERD Model databáze



Obr. 4-6 – ERD model databáze internetového obchodu.

B Elektronická podoba diplomové práce

Součástí diplomové práce je přiložené CD, které obsahuje:

- Soubor Read_me.doc, obsahující informace o obsahu CD
- Software APACHE, MySQL
- Zdrojové kódy aplikací
- Text diplomové práce

Výpis souboru Read_me.doc.

Toto CD je součástí diplomové práce Použití objektových forem ve webových aplikacích. V adresáři Eshop jsou zdrojové kódy internetového obchodu. V adresáři Admin jsou zdrojové kódy správcovské aplikace. Adresář software obsahuje potřebný software pro běh aplikací. Adresář Db obsahuje databázi internetového obchodu.