

Ministry of Education and Science of Ukraine
Ternopil Ivan Puluj National Technical University

Faculty of Computer Information System and Software Engineering

(full name of faculty)

Department of Computer Science

(full name of department)

QUALIFYING PAPER

For the degree of

Bachelor

(degree name)

topic: Creating an information website for a trading company

Submitted by: fourth year student _____, group ICH-43

specialty 122 Computer science

(code and name of specialty)

Rifat Md Shakil
Mahamud

(signature)

(surname and initials)

Supervisor

Strutynska I.V.

(signature)

(surname and initials)

Standards verified by

Matsiuk O.V.

(signature)

(surname and initials)

Head of Department

Bodnarchuk I.O.

(signature)

(surname and initials)

Reviewer

(signature)

(surname and initials)

Ternopil
2022

6. Advisors of paper chapters

Chapter	Advisor's surname, initials and position	Signature, date	
		assignment was given by	assignment was received by
Occupational health and emergency safety			

7. Date of receiving the assignment 06.09.2021

TIME SCHEDULE

LN	Paper stages	Paper stages deadlines	Notes
1	Analysis of the task for qualifying work. Selection and work with literary sources.	06.09.2021	<i>Completed</i>
2	Writing chapter 1		<i>Completed</i>
3	Writing chapter 2		<i>Completed</i>
4	Writing chapter 3		<i>Completed</i>
5	Writing chapter 4		<i>Completed</i>
6	Standartization control		<i>Completed</i>
7	Plagiarism check		<i>Completed</i>
8	Preliminary defense of qualifying paper		<i>Completed</i>
9	Defense of qualifying paper	27.01.2022	

Student

(signature)

Rifat Md Shakil Mahamud

(surname and initials)

Paper supervisor

(signature)

Strutynska I.V.

(surname and initials)

ANNOTATION

Creating an information website for a trading company // Diploma thesis Bachelor degree // Rifat Md Shakil Mahamud // Ternopil' Ivan Puluj National Technical University, Faculty of Computer Information System and Software Engineering, Department of Computer Science // Ternopil', 2022 // P. ____, Fig. – ____, Tables – ____, Annexes – ____, References – ____.

Keywords: «magento», content management system, system management content, internet-shop, electronic commerce, entity-attribute-value model, model-view-controller.

The aim of the work is to study the methods and means of creating a Web-site for online store using the content management system "«Magento»" that automates and accelerates the flow of major business processes in this area.

The practical value of the work lies in the possibility of using it results in the development and study of e-commerce systems, namely organization of online stores.

The results of the thesis. The studied technology provided there are opportunities to develop a ready-to-use e-commerce system flexible enough for further development and improvement, contains features that have not yet been implemented in other software products, and implemented using modern technologies that allow to optimize and improve the process of software development in this area.

LIST OF SYMBOLS, UNITS, ABBREVIATIONS AND TERMS

- DB - database
- DBMS - Database management system
- SQL - Structured Query Language
- CRUD - (Create, Read, Update, Delete) 4 basic data management functions
- OS - operating system
- ORM - Object-Relational Mapping
- CMS - Content Management System
- HTML - HyperText Markup Language
- PHP - Hypertext Preprocessor
- CSS - Cascading Style Sheets
- EAV - Entity-Attribute-Value - the value of the entity attribute
- MVC - Model-View -Controller
- EDP - (Event-Driven Programming) - event-oriented programming
- URL - Uniform Resource Locator - the only pointer to the resource
- XML - Extensible Markup Language
- WAMP - "Windows, Apache, MySQL and PHP" - short for describes a set of software packages for the corresponding OS, indicated by the first letter

CONTENTS

INTRODUCTION.....	7
1 RESEARCH OF CONTENT MANAGEMENT SYSTEMS IN THE FIELD OF E-COMMERCE	9
1.1 Web content management system.....	9
1.2. Main functions of CMS.....	11
1.3. Ways of work.....	11
1.4. Review of existing solutions.....	12
2 DEVELOPMENT OF A WEBSITE DESIGN BASED ON THE CONSTITUENT ELEMENTS «MAGENTO» ARCHITECTURE.....	28
2.1 General overview	28
2.2. Technological base.....	30
2.3. System requirements	30
2.5. Store website structure	32
2.6. Design themes	33
2.7. Packages and themes in «Magento» directory structure.....	34
3 CREATING AN ONLINE STORE WEBSITE AND YOUR OWN MODULE BASED ON CMS «MAGENTO»	52
3.1 Installing the system on a local server	52
3.2. Overview of implementation of MVC scheme.....	53
3.3. Rules for creating modules	55
3.4. Pattern of factory.....	56
3.5. Events.....	60

	6
3.6. Database model	62
3.7. Addressing and controllers	68
3.8. Creating a module	71
3.9. Creating themes.....	76
4 OCCUPATIONAL HEALTH AND EMERGENCY SAFETY.....	78
4.1 Effects of electromagnetic radiation on the human body	78
4.2 Types of hazards	81
4.3 Road Transport Safety	84
4.4 Conclusions.....	85
CONCLUSION	86
REFERENCES.....	87

ANNEXES

INTRODUCTION

The business owner has to deal with the business sooner or later the problem of creating an online store. You can find a huge amount of information about ways to solve this problem from your own trade platform on the ready service to a full-fledged online store on the Internet. Having chosen the first option will still have to continue the search in the direction real online store. Problem is editing capabilities the appearance, functionality of trading floor is very limited, your business partners may not take an "online store" very seriously.

Online shopping is a part of e-commerce, usually B2C (business-to-consumer). B2C includes the collection of info from customers; take physical things or info / electronic good; for information good, receipt of the goods (programs, games, eBooks, other works art) on electronic network. Examples of B2C website are Network Company's retail like "Amazon.com", "Drugstore.com" and "Beyond.com". B2C electronic commerce reduces price for transactions (especially the search price), increasing access, as rule, consumer to information and allowing, as rule, consumer to find the most competitive, as rule, price for a product/service. B2C e-commerce as well as reduces market border to entry because cost of creation/promotion website is much smaller than setting the structure of a firm. In the case of goods information, B2C e-commerce is more attractive because it saves a firm from factor of added value of the physical-distribution-network. In addition, countries with the growing number of Internet users, the supply of products information is nowadays becoming more and much affordable.

Last 5 years Bangladeshi magazine on internet showed annual progress of 50% regardless of permanent economic fluctuations. At the same time, e-commerce market place is serious potential. Indeed, in Germany, the capacity of e-commerce market is about \$ 35 billion. In Bangladesh, a similar figure is not more than \$ 500-\$ 600 million.

The main factors that hinder the development of the Bangladeshi market e-commerce are:

Insufficient legal framework to regulate processes □ buy / sell on online, make electronic payments of paid □ goods or services and establish uniform transparent rules of game in the market (for □ sellers/buyers);

Poorly developed national system for electronic payments;

Rather low efficiency of majority of existing Bangladeshi □ trading platforms;

Insufficient level of Internet penetration.

Websites are often created on the basis of “content management systems.” This approach allows us to significantly reduce development time, because maximum required functionality has already been implemented.

Content management system is a tool designed for convenience filling, further content management of the website. Generally, all CMS include ability to manage a website template, in the end result will be your website. For popular CMS there are more ready-made templates and it is possible to make our own, individual site templates.

1 RESEARCH OF CONTENT MANAGEMENT SYSTEMS IN THE FIELD OF E-COMMERCE

Content management system (CMS) is software for organizing websites or other information resources on Internet or individual computer networks.

CMS greatly simplifies creation and maintenance of websites.

1.1 Web content management system

Web Content Management System is software that gives administrative tools for organization authorizations, collaborations and allow users with minor knowledge in programming to make and manage site content with relative simplicity.

Presentation Layer (Template Engine) shows website content to visitors based on a group of templates.

Maximum systems usage server-side caching for improve performance.

Administration is usually done by a web-interface, but a few system require use of a "thick" client.

Those systems allow technically untrained users make changes to website after the training course. To configure or adding new features to the site usually requires a system administrator or web developer.

We can find hundreds of CMS. Thanks to their functionality, CMS is possible use in different companies/organization despite the wide selection.

Tools and hardware available in CMS are same to maximum type of characteristic systems.

Most modern CMS have a modular architecture that allow us the administrator to select or configure components it needs.

Typical modules: dynamic menu, blog, news, polls, site search, visit statistics, guestbook and others.

Sites organized by content management systems based on the following technologies: web server, data warehouse (often DBMS, example MySQL or PostgreSQL, but there are also NoSQL-CMS), web application to ensure operation of the system, visual (WYSIWYG - what you see is what you get) page editor, web file manager interface for site file management.

Most common technological platforms used as the basis of a web application that implements CMS: PHP, Perl, .NET.

Many modern SLE are distributed as free and easy to install programs developed by groups of enthusiasts under the free licenses such as GNU / GPL, MIT, or Apache. There are many companies that provide website creation and support services based on such systems.

There is a term content manager that reflects the kind of professional activity, i.e. the employee who is responsible for filling (content) site. CMS allow a content manager without programming skills.

Most modern content management systems have a visual editor that gives users to create and edit pages using simplified mark-up.

The first SLE was developed in large corporations to organize work with documentation. In 1995, a separate company separated from CNET

Vignette, who started the market for commercial SLE. Over time the range products expanded and increasingly integrated into modern network solutions to popular web portals.

1.2. Main functions of CMS

Providing ability to manage the website with a minimum level of training even possible to create a website from ready-made blocks.

Adding, editing, deleting information from the website.

Organization of collaboration with content.

Manage user access mode.

Creating web pages.

1.3. Ways of work

Create pages on demand. Methods of this type of work based on link <"editing module → database → presentation module">. When requested, presentation module creates a page with content on based on information from database. We can change information in database with editing module. For each requests pages are re-generate by server, and this creates a load on server. But this load can be minimized many times by using different caching methods.

Generate pages when editing. This type of system changes the content of the website and makes a group of static pages.

Mixed type. This type of system combines advantages of the first two system. It can be done by caching - module the view creates a page once, then it can boot from the cache several times faster. Cache can be updated as automatically, after some times when making changes to particular parts of the website, manually at order of administrator. Second way is preserve particular information blocks at the time of editing the website and collecting page from the blocks when user requests for corresponding page.

1.4. Review of existing solutions

Consider some of popular content management systems which used in e-commerce.

“VirtueMart”

“VirtueMart”, formerly known as mambo-phpShop, is a free software for creating an online-store, created to complement such website content management systems like “Mambo” and “Joomla”. “VirtueMart” is written on PHP, uses MySQL database. It is good for websites with low or medium load. VirtueMart is protected by a GNU-license (GPL). VirtueMart first appeared as a standalone phpShop program for creating the online store. After Mambo branched out into “Joomla”, the developer make a change of the mambo-phpShop brand to VirtueMart, which it officially supports newer Joomla site management system. Current versions of VirtueMart can to operate with Mambo.

VirtueMart supports unlimited number of products/categories, unlimited number of currencies for a product, purpose of many products of categories, the efficiency to sell products for download, also gives us power to disable sales feature, use VirtueMart's directory method. VirtueMart allows us to have several prices for the same product, based on number or affiliation of buyer to a particular group, provides us possibility of using various payment systems. “VirtueMart” has system of coupons, discounts, and also a large selection payment systems or delivery methods.

When migrating “VirtueMart” from Mambo to “Joomla”, the developers did not pay attention to differences between “Mambo” and “Joomla”, resulting in VirtueMart

branches 1.0.x contains a lot of old procedural code and components that already exist in Joomla, but for some reason not used.

Admin panel:

Marketing-tools

Multilevel pricing.

Coupons.

Pricing rules.

Determination of the tax rates

Define rules for a region/country.

Creating tax classes, such as normal or wholesale.

Setting VAT rate.

Products and catalogue

Efficiency to specify different attributes for the product.

Limitless nesting of categories.

Configured products with selective properties (for example: color/size)

Sorting goods by predefined attributes (performed by using an additional module).

Administration

Groups of buyers (simple, VIP, etc.)

Import / Export (in the form of third-party developments)

Access control system

Currency selection

Choice of countries

Sales reports

Choice of delivery methods

Choice of payment methods

Front end features:

Offer to buyers of related products

Offer to buyers of recently viewed products

Search engine optimization (SEO) - so far in the form of third-party developments

Contact the buyer

Email notifications

RSS subscription to a category or all product groups

Search and advanced search

Comparison of products - in the form of third-party developments

Product tags - in the form of third-party developments

Customer reviews of products

Making an order

Shopping cart

Customer accounts

Recommendation of goods by one buyer to another

Top 10 products

Display special product offers

Import and export data:

The standard build does not include import / export modules, only in the form of third-party developments.

Import and export of goods

Import and export of goods balances in warehouses

Import and export of buyers

Ability to create import / export templates for different purposes. There are third-party development of export modules in Index. Market and others.

Payment methods:

Credit card

Payment on delivery

PayPal

eCheck.net

MerchantWarrior

Third-party development of payment modules, including WebMoney, Yandex.Money, RBKMoney, Qiwi Visa Wallet, PayKeeper, Assist and others.

Delivery methods:

Auspost

Canada Message

DHL

FedEXdc

Flex

InterShipper

Shipvalue

International Postal Union

USPS

Standard delivery module

Delivery depending on the region or country

Self-pickup

PrestaShop

PrestaShop is an open source e-commerce web application.

Written in PHP, Smarty is used to write templates, designed for MySQL. This engine is designed for small and medium business and has over 310 standard features for quick creation functional store. In 2010-2014. Recognized as the best web

application for online stores, winning the Best Open-source Business Application award.

Translated into almost 60 languages, the community numbers more than 500,000 man.

There is paid and free support. The company's headquarters are located in Paris and Miami. The team consists of more than 100 people and consists of developers, designers and IT specialists in e-commerce.

PrestaShop is sometimes compared to «Magento» and OpenCart. Yields the first size of the community and the number of ready-made solutions, but easier in development. OpenCart like PrestaShop is easy to learn, and is gaining popularity with every year.

Functionality

Catalog

Payment

Delivery of goods

Statistics

Translations

Localization

Security

Management

WYSIWYG text editor

Ability to add additional modules

Database backup (partial or complete)

Automatic generation of hatches files

Automatic generation of robots.txt files

Send e-mail via SMTP (SSL and TLS supported) or using the PHP mail function.

Font management in PDF files
Indexing of goods to optimize search
Inventory management
SMS (inventory, new order ...)
No e-mail notifications available
Store modules
Ajax – basket
Cloud of labels
Ajax Search
Alias –search
Customizable pages (text, photos)
Stage products (product presentation)
Product tips: date of purchase or add to cart
On-Off modules
Products recommended on the main page
Ability to insert ads
Show new products
Ability to integrate with Google AdSense
Bookmark in one click
Choice of currency
Language selection
Products RSS feed
Show other products in the same category
Show the best sellers
Display category products
Display an additional link
Show manufacturers

Show suppliers

Block "My account"

Full customer account display (information, orders, slips ...)

Subscribe to the newsletter block

RSS feed for subscription

Emphasizing available means of payment

The third block is the RSS feed

Quick search

Ubercart

Ubercart is also a free software for opening trades platforms, online-stores, auctions, created for supplement such content-management-systems like “CMF Drupal”. We can install Ubercart module on any server that supports “PHP and MySQL” and distributed under the “GNU GPL license”.

Ubercart is fully integrated with “Drupal”, that means 100% compatibility with every website that use Drupal. The module can be use both in cases of sale of goods and services, and downloading all online products. In addition, we can use it for hiring and employment, creation of significant events. Main purpose Ubercart has ability to open a limitless number of different payment gateways electronic world currencies and banking systems. Ubercart allows us to spend a number of comprehensive actions within Drupal CMF, namely: adding a new one product, re-sell goods, make discounts and issue coupons, exchange currency, make any domestic and international payments, support affiliate programs, pay commissions, issue coupons on licensed goods and services, to conduct a comprehensive classification of products enterprises.

Like Drupal, Ubercart can run on variety of language platforms.

“Drupal” is a popular open modular content-management-system (CMS) with open source, written in the PHP programming language.

The task of content-management-systems is to facilitate the creation, filling, website update. Drupal can run on popular systems such as

“Windows, Mac OS X, Linux.” In fact, on any platform that supports

“Apache, Nginx, Lighttpd or Microsoft IIS web servers”; also required database management systems MySQL / MariaDB, “PostgreSQL”, SQLite/others commercial databases. Full Drupal system requirements are listed on the official website.

Critics of “Drupal” blame developers for poor use of PHP object capabilities. The Drupal API create little use of the existing ones in PHP OOP capabilities. Developers argue, this is weak implementation of PLO language (especially before PHP version 5). The object model in “Drupal” is present, but little unconventional for PHP.

The disadvantages (but also advantages) of “Drupal” include the lack of backward API compatibility with fairly high dynamics of project development. API changes occur in almost every release, along with the addition new features remove some old or change call settings functions. This directs to the need for third-party module developers, adapt them to work with new versions of Drupal. However, API changes and procedure adaptations of modules to newer versions are described in documentation for each release, and an automated system kernel upgrade mechanism is always offered to new version. Plus this development scheme - no need to pull from version of software layer is suitable with old API, which facilitates current code system.

System distribution includes a group of modules that give the following features:

collection of information feeds (RSS, RDF, Atom);

blogging, bidding, forums;

creating forms for sending messages;

system localization;

rename link (assign links are clear and convenient pseudonyms);

- conducting surveys;
- customizable profiles
- search by content (content is considered as messages on forums, pages, and any other assigned items);
- keeping a journal of statistics (attendance);
- taxonomy (ordering material by category) - very "valuable" possibility;
- Inclusion of mechanism of auto regulator of control of loading and others.

OpenCart

OpenCart is one of the free source of e-commerce system that distributed by GNU <General-Public-License>.

We can install OpenCart on any Apache web-server that supports for PHP5 or above and MySQL. OpenCart built a very large community (over 46000 members) that is created over 8,500 free extensions in the form of additional-modules.

Real important advantages of OpenCart over «Magento», Virtuemart or osCommerce is modern MVC architecture, increased speed, vQmod, multifunctional administrative content-management panel, less consumption of server resources.

OpenCart has proven itself in the commercial sector as trustworthy and low-maintenance e-commerce system with support calculation of all of the most famous electronic payment systems.

Key features.

Technical advantages:

- Support for PHP 5.x and MySQL 4.x, 5.x.
- The code corresponds to the basic principles of the Model-View-Controller template, which allows to carry out works of various complexity independently of each other.
- Compared to competitors (Magento, Virtuemart, osCommerce) has the best speeds up and carries less load on the server.
- Support for many modern browsers.
- Built-in multilingualism. Ukrainian language available.
- Unlimited number of categories and products.
- Support for templates, modules and add-ons.

Administration / Base:

- Support for one or more stores.
- Unlimited number of products and categories.
- Support for physical and virtual (downloadable) products.
- Ease of backup and recovery of the database.
- Statistics of goods, orders and sales.
- Multilingualism.
- Support for currencies and exchange rates.

Client part:

- Buyer registration.
- All orders are stored in a database for efficient history retrieval shopping.
- Customers can view the history and status of their orders.
- Temporary basket for guests and permanent for registered customers.
- Fast and convenient search tool.
- SSL (Secure Sockets Layer) support.
- Convenient site navigation.
- The customer can have several delivery addresses in the personal address book.

Payment and delivery system:

- Support for many types of payments (checks, payment orders).
- Support for many payment systems using modules (2Checkout, PayPal, Authorize.Net, iPayment, RuPay, Webmoney).
- Set up payment methods for different regions.
- Calculation of delivery based on weight and price of goods and delivery area.
- Many delivery calculation modules.
- Calculation of taxes taking into account the region.

It's worth noting:

PHP-based stenciling.

SEO is not optimized at this time.

Fast version development, which requires frequent kernel updates.

osCommerce

osCommerce <"Open Source Commerce"> is an online store engine.

We can install it on any web server which supports PHP and MySQL. This is a free software along GNU General Public License.

A big community has built around osCommerce (more than 140,000 participants), there are more than 4,000 contributions (various modules) for osCommerce, that allow us to change and supplement the functions of the store in very different ways. There are thousands of stores around the world osCommerce database (14,063 stores are officially registered in the catalog).

Key features:

- Compatible with PHP 4.x, 5.x and MySQL 4.x, 5.x.
- Compatible with all major browsers.
- Built-in multilingual, default English, German, Spanish, Russian, Ukrainian and many others are available.
- Installation wizard (wizard).
- Unlimited number of sections and products.

Administration / Base:

- Supports an unlimited number of products and category sections.
- Support for physical and virtual (downloadable) products.
- Ease of data backup and recovery.
- Statistics of goods and customers.
- Multilingual support.
- Support for multiple currencies.

Client part:

- Buyer registration.
- All orders are stored in a database for fast and efficient search (shopping history for buyers).
- Customers can view the history and status of their orders.
- Temporary basket for guests and permanent for customers.
- Fast and friendly search interface.
- Security with SSL (Secure Sockets Layer) support.
- The customer can have several delivery addresses in his address book.

Payment and delivery system:

- Support for numerous types of payments (checks, payment orders).
- Support for numerous payment systems (modules) (2CheckOut, PayPal, Authorize.Net, iPayment, RuPay, Webmoney).
- Set up payment methods for different areas.
- Calculation of delivery based on weight and price of goods, delivery area.
- Calculation of taxes.

Unfortunately, osCommerce doesn't have a template system (up to Alpha 5) to customize design, but here is some modules that solve the problem. Example: STS, BTS. Maximum osCommerce clones also have a templater system.

«Magento»

«Magento» is a professional online store management system. By According to Alexa, «Magento» is the most popular Internet management system stores in the world in February 2019. Back in 2012. »Magento» Inc. was acquired by eBay Inc.

The appearance of the window in «Magento» is determined by the so-called themes:

- A set of templates for displaying blocks visible on the screen;

- A set of rules that determine which blocks are in which place on a particular page display and merge into layouts;

- Set of resource files (skins): CSS, images, scripts in JavaScript.

Unlike maximum other site-management-systems (e.g. “Joomla”), in «Magento» themes can be related to one relationship inheritance: if one of the specific template is not defined in the current topic, rule display or resource file, the system takes them from the base theme.

For example, all three standard themes for »Magento»: Default, Modern and iPhones are inherited from a single basic theme. A third-party programmer can create your own base theme.

Unlike Joomla, where directly in the administrative part is possible assign the display location and parameters of a specific module on the page, «Magento» is mainly focused on editing display units not showing through administrative part, and through logical markup files and template files.

This approach, on the one hand, gives great flexibility to store developers, but, with on the other hand this is one of the reasons why ordinary «Magento» administrators is considered a difficult system to study.

Opportunities

Basic:

We can manage the retail network from a single administrative part of □ online stores in other domains, languages, with different product range.

Flexible adjustment of appearance ("themes" and "templates") of a show-window.

Display prices in several currencies (including at option of buyer).

Multilingualism.

Payment plus delivery methods.

Admin panel:

Marketing tools.

Multilevel pricing.

Coupons.

Ability to organize sales, set wholesale discounts

Sales reports, customer baskets and the list of marked goods.

Recall report.

Report by tags and Search.

Determination of tax rates.

Define rules for a region, country, or zip code.

Creating tax classes, such as "Normal" or "Wholesale".

Defining tax rules, such as the Clothing Tax, is also possible link different groups of goods with tax classes.

Products and catalog.

Ability to specify different attributes (properties) for the product.

Grouping of goods into sets.

Configured products - products with selective properties (for example color, size, etc.).

Sorting goods by predefined attributes. Attributes are available for sortings are defined in the administration panel.

Content management system.

Buyer groups.

Import / Export (Currently, imports of goods are limited option, it is possible to make goods, but not updates).

Access control system.

Currencies

We can keep records (warehousing and management) in one currency, and prices for show in another.

Different currencies of price tags can be assigned to different showcases.

We can give visitors the power to switch between currencies.

We can set up exchange rate updates on a schedule.

Setting the display rates are determined by the live location of visitor (Zend Locale library can be use).

Search engine optimization capabilities

Standard «Magento» build:

Gives full control over HTML. It's possible to install an individual HTML template for a specific product or product section.

Allows us to specify for each page (product, section or text) individual Meta tags description and keywords.

Allows you to specify individual and for each product and product section configure the end of the address (URL Key).

Allows you to manage the TITLE header for each page (available as automatic mode and manual).

Automatically creates an XML map for search bots for the site.

Search engine optimization of product images. «Magento» when creating

Showcase pages describe product images with the ALT attribute of the IMG tag. The administrator for each product image can either manually specify a description, or allow «Magento» to describe the product image automatically.

2 DEVELOPMENT OF A WEBSITE DESIGN BASED ON THE CONSTITUENT ELEMENTS «MAGENTO» ARCHITECTURE

2.1 General overview

«Magento» is an e-commerce platform for creating e-commerce open source stores, which is distributed according to OSL 3.0.

«Magento» is one of the most popular open systems for organizations e-commerce online: more than 150,000 have been created on the basis of this platform online stores, third-party developers have created more than 2,500 extensions, the project community has about 400,000 members, the platform code was downloaded more than 3.5 million times, through «Magento»-based systems each year goods sold for more than \$ 28 billion. In a few years of existence on the platform, it has repeatedly won award "Best of Open Source Software Awards" and "SourceForge Community Choice Awards".

«Magento» software is created using Zend Framework. Raw «Magento» source code is distributed like approved Open Source Initiative (OSI) Open Software License (OSL) v3.0, same in nature to the AGPL license, but not suitable with the GPL. Feature of OSL license is, if someone modifies program, it will not disseminate the result of its work and simply use it on its website, then it open code of changes, after that distribute it under same license. However, OSL does not prohibit commercial use of software products.

Diagram of online store usage options, which is developed in thesis, shown in Figure 2.1, and the corresponding class diagram on Figure 2.2. The «Magento» content management system greatly simplifies the process creating an online store of any profile.

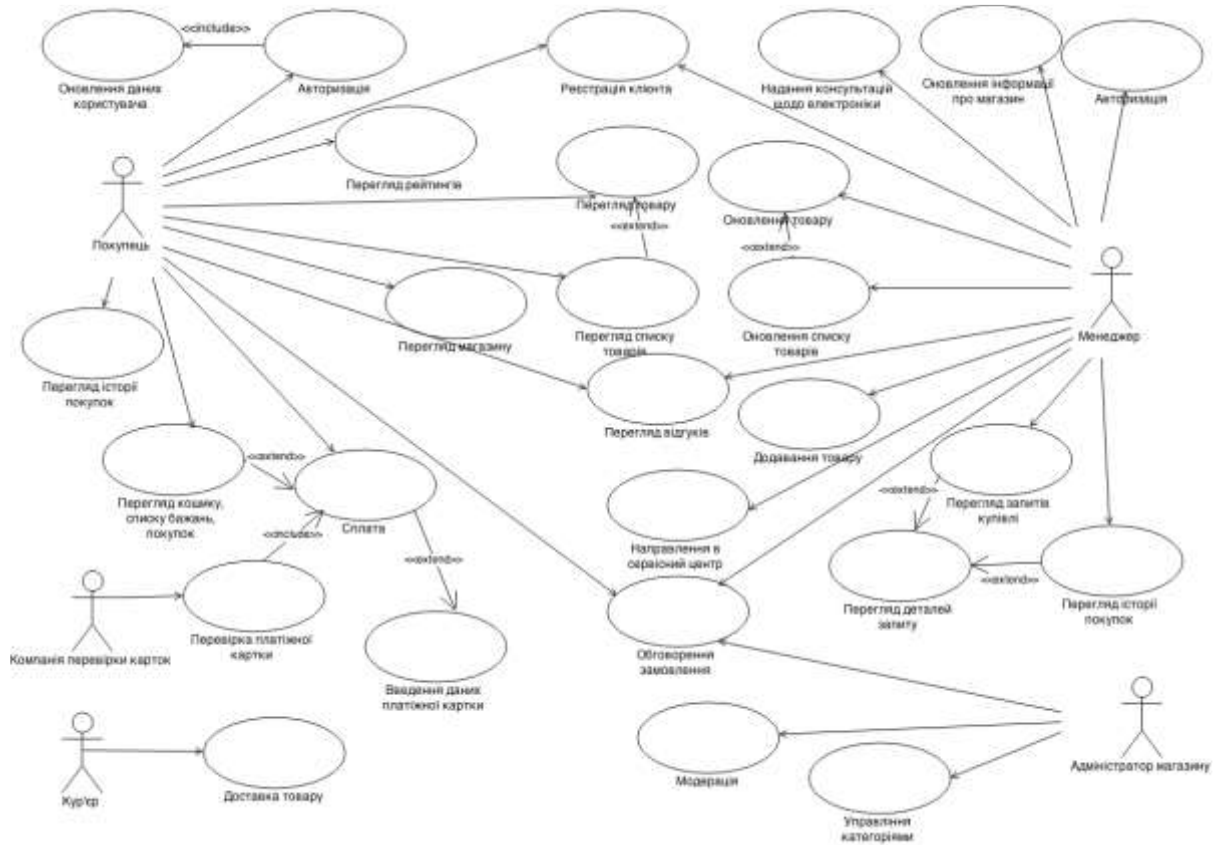


Figure 2.1 – Diagram of options for using the online store

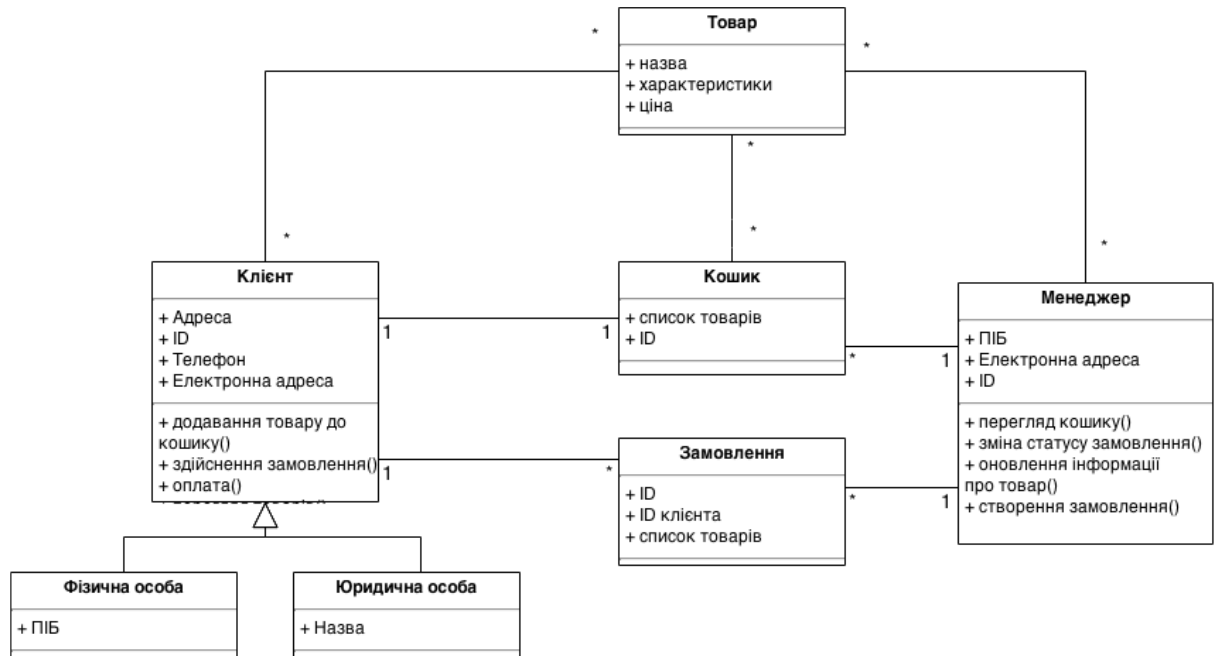


Figure 2.2 – Online store class diagram

It is known that several large Ukrainian online stores also use «Magento», among them:

Comfy (comfy.ua, revenue of UAH 4.2 billion for the first half of 2019.

Allo (turnover in 2018 - \$ 29.5 million).

Mobilochka and others.

2.2. Technological base

Like any other web application, «Magento» has a client-server architecture. Backend, the server portion written in PHP + SQL.

PHP (Hypertext Preprocessor) for now moment is most common language for creating web applications and it supported the vast majority of hosting providers, which is a plus point. MySQL is also supported by most hosting providers so problems with implementation of the system should not occur.

Due to the fact that PHP is a high-level programming language with a large library of implemented functions, development can focus on programming business logic instead of spending time on various low-level things.

2.3. System requirements

Linux or other UNIX-compatible operating system (Windows also is supported, but first you should read all known problems).

Apache 2.x or Nginx 1.7.x web server.

PHP 5.4 and higher.

MySQL 5.6 and higher is also supported by Oracle or Percona.

Send mail- compatible mail server. »Magento» can connect directly with the SMTP server.

Most modern hosting services meet these requirements.

2.4. Overview administrative panel main features

The convenience and feature, as rule, set of the admin panel is important part of the «Magento»-based online store system [8-11]. On the main page The admin panel displays some summary data of the store's activities, such as as the top search queries and best-selling products (Fig. 2.3).

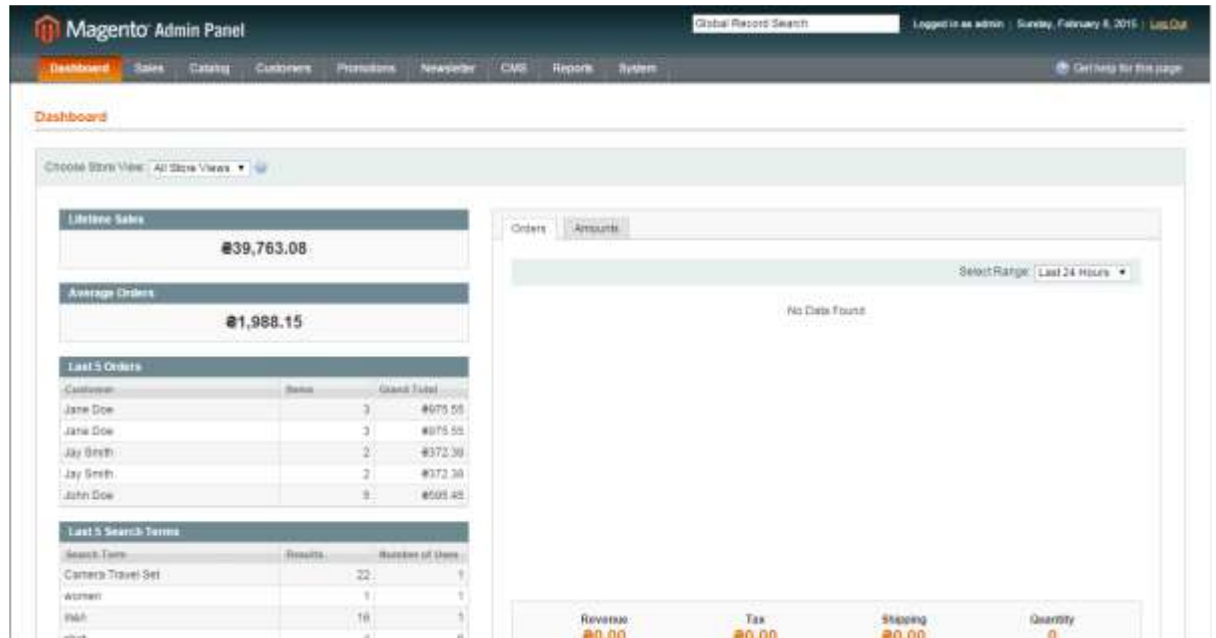


Figure 2.3 – Administrative panel. Main page

The main menu contains other items that are grouped by destination.

So in the group "Sales", you can view information about orders, invoices, delivery and payment.

In the Catalog, as rule, group, you can usually manage products, product, as we know, categories, attributes, search terms, page redirection, reviews, and ratings.

In the Customers group, you can manage customer information, groups of buyers, to export and import this data.

In the group of promotions (Promotions) - set different promotions and their rules application.

In the Newsletter group, manage newsletters and subscribers to these newsletters.

In CMS group - we can configure static pages, static content blocks, polls, widgets.

In Reports group- we can view various reports, statistics online store activities.

In System group- there are various settings of work systems.

2.5. Store website structure

«Magento» site, as rule, is a group of stores which have their own store views.

He consists of one, as rule, or more stores, usually share information about, as rule, order information, customers, between them shopping cart.

Stores, as rule, are a group of views and it can be usually customized in many ways. Their primary function is giving a logical container for group related store views on the website.

Presentations are actually copies of store in «Magento». Maximum stores have a single representation attached with them. But store, as rule, can have multiple views too.

Example, if we need to show the store in various languages, we can make one store with several views (Fig. 2.4).

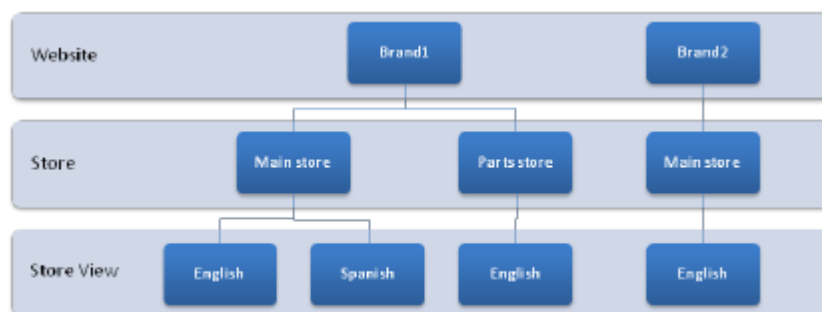


Figure 1. Hierarchy of websites, stores and store views in Magento.

Figure 2.4 – Store hierarchy

So one «Magento» installation we can use with several stores that may look like similar, but not necessarily.

2.6. Design themes

For complete control over the outlook of every «Magento» store allows us to make themes. Related topics are united into design packages. All store can use its personal theme/they can share/make a combination.

We can use many installed design packages, but not less than one. Installing «Magento» has <“special base package”> and other packages, depending on edition. Theme should be in one package.

Traditionally, each package should have a "default theme".

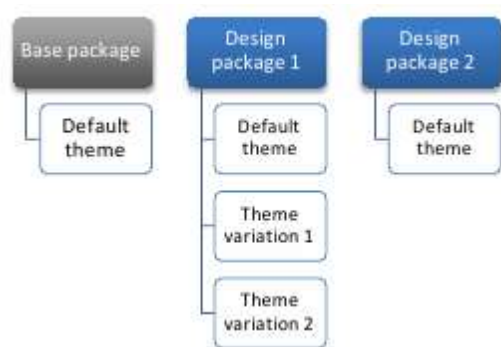


Figure 2.5 – Design packages hierarchy

Design package, as rule, we can be attached theme to both site and a separate store from «Magento» admin panel. If we assign a package to levels of site, then every stores will follow this package. It may be overridden store level by purpose of another package.

Topics include information for templating <markup, templates, translation>, style information <CSS-files, images, JavaScript scripts>.

Every files are located in two main directories:

"App/design" - files which define how many page of templates are executed.

"Skin" - files, as rule, which control visual parts of theme - CSS, images, etc.

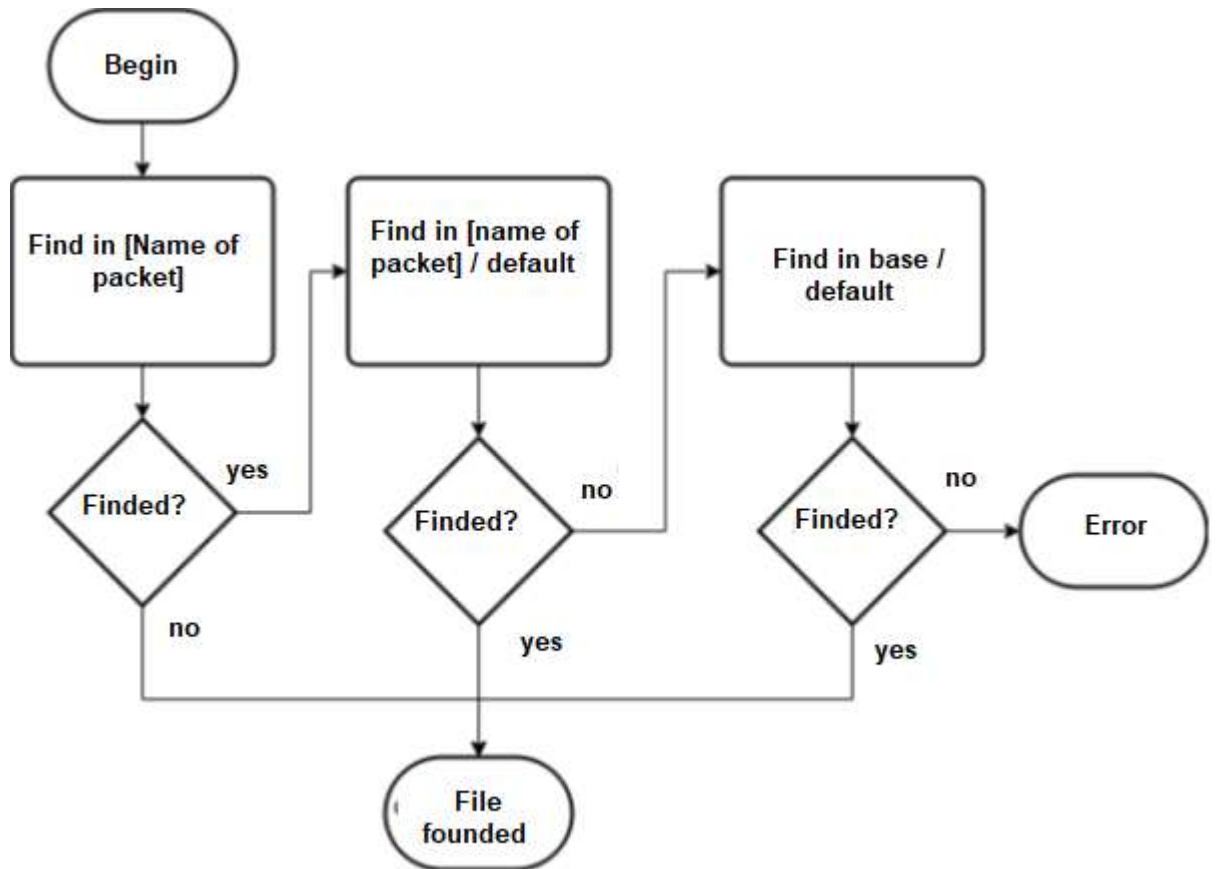


Figure 2.6 – Fallback logic

2.7. Packages and themes in «Magento» directory structure

Themes are logically united into design, as rule, packages, but theme files are divided into 2 directories. Directory names for the design, as rule, packages + themes usually need to be same in app / design and skin directories. Also note that in every from of directories there is an essential group of directories where files actually are stored.

“Base package”

The base-package has been added to CE v1.4, EE v1.8. The motive of base package is providing representation for all basic functionality. Custom themes can only include changes which are inherent design/business they should support. »Magento« use "fallback" model, which searches for files needed for the page display, performed 1st in chosen custom package design, and only then in basic package.

Base packet is endpoint for the rest of entire packets and need to be unchanged when making or editing custom themes. It provides cleaner code and a good upgrade way for custom themes as base package holds all files that control «Magento»'s behaviour by default, and custom themes are the only files which have been modified. Base, as rule, package have only default theme and we should not add other themes to base package.

In directory structure, base package hold template and skin files, which are in app / design, skin directories, respectively. Basic package has only default theme, although it isn't a perfect theme because it lacks many peel files.

Base package structure

App / design / frontend / base / default /

It contains all layout and template files essential for support nuclei Magento-functionality.

<MAGENTO_BASE_DIR>

- + app
 - + code
 - + design
 - + adminhtml
 - + front
 - + base
 - + default
 - + default
 - + install
- + etc.
- + local

Mage.php

Skin / frontend / base / default /

It contains some CSS plus JavaScript files that support kernel-functionality. But all of the CSS files plus images are essential to style the website because they belong to the design.

<MAGENTO_BASE_DIR>

- + skin
 - + adminhtml
 - + front
 - + base
 - + default
 - + default
 - + install

Rules for working with basic package:

Don't modify files in base package.

Don't build custom themes in the base package.

Other defaults

Each «Magento» distribution contains not only the basic package, it contains minimum one extra. In Community Edition, the name of package is "default". In Professional Edition - "pro". In Enterprise Edition - "enterprise".

Normally basic theme package have a full set of CSS files, images.

In Community Edition, default package have several themes:

App/design/frontend/default/

<MAGENTO_BASE_DIR>

- + App
 - + Design
 - + Frontend
 - + Base
 - + Default
 - + Default
 - + Blank
 - + Default
 - + iPhone
 - + Modern

Skin/frontend/default/

<MAGENTO_BASE_DIR>

- + Skin
 - + Frontend
 - + Base
 - + Default
 - + Default
 - + Blank
 - + Blue
 - + Default
 - + iPhone
 - + Modern

Professional Edition or Enterprise Edition have 1 default theme in Pro/Enterprise packages respectively.

Custom design themes

For build a custom theme we need to make folders, as rule, in the custom package in app/design, skin directories.

Example, if we want to build a package called "Dubloo", we need to build a folder with appropriate name and folder in every frontend directory "Default" for that theme.

```

App / design / frontend / dubloo / default /
<MAGENTO_BASE_DIR>
+ App
+ Code
+ Design
+ Front
+ Base
+ Default
+ Double
+ Default

```

```

Skin / frontend / dubloo / default /
<MAGENTO_BASE_DIR>
+ Skin
+ Front
+ Base
+ Default
+ Double
+ Default

```

Further contents of every folders rely on level of change relative to baseline-package.

Every design package have to contain minimum one "default", as rule, theme and it can have any variations.

```

App / design / frontend / dubloo / default /
<MAGENTO_BASE_DIR>
+ App
+ Design
+ Front
+ Base
+ Default
+ Double
+ Default
+ Layout
+ Template

```

```

Skin / frontend / dubloo / default /
<MAGENTO_BASE_DIR>
+ Skin
+ Front
+ Base
+ Default
+ Double
+ Default
+ CSS
+ images
+ js

```

We have small difference working with the PE and EE from CE. Additional operation in PE and EE aren't the part of base package, but it hold in pro and enterprise packages gradually. To build a custom package for PE and EE, we have to copy default theme from pro and enterprise packages gradually to new custom package. After that every changes will be in the variation of the theme.

To build a custom theme in pro or enterprise packages is an alternative, but it's recommended to build a new package to reduce risk of changing enterprise / pro / default code.

App / design / frontend / double / your_theme /
 <MAGENTO_BASE_DIR>

- + app
- + design
 - + front
 - + base
 - + enterprise
 - + double
 - + default (copy from EE / def)
 - + your_theme
 - + layout
 - + template

Skin / frontend / dubloo / your_theme /
 <MAGENTO_BASE_DIR>

- + skin
 - + front
 - + base
 - + enterprise
 - + double
 - + default (copy from EE / def)
 - + your_theme
 - + CSS
 - + images
 - + js

Application of themes

Consider how to build and apply a newly designed, as rule, theme and apply this to «Magento»-store. Example, we have one new package with a few minor changes, downloaded from «Magento» Connect or from another colleague.

To apply theme, we have to go to administrative panel (example, “www.mydomain.com/admin”), after that on design-settings-tab (System -> Configuration -> Design tab). If we have more than one site, store/presentation, here

will be a block in the upper left corner of left column called "Current-Configuration-Scope" where, as rule, we will store the changes which, as rule, are applied.

Step 1.

From Design tab, in Current package called field, enter package name, c which is our new topic. If we leave this field blank, it will use "default" package.

Step 2.

In default field, put the name of new topic we want to put to website of store/presentation (Fig. 2.7). If we leave this field blank, it will apply "default" theme from package. According to indentation scheme in «Magento», if the name of the theme is specified, here will be a search file in that specified topic in "default" package theme, and at end in the "default" - theme "base" package. We can also specify separate theme for certain file-types like layouts, skins, translations, templates.



Figure 2.7 – Installation process in admin panel for theme

Step 3.

Click on "Save-config" to save configuration and reboot shop.

«Magento» approves us to use multiple themes in one layout/template. Example, we need the same form of template and layout for entire views, but different graphics plus colors-scheme. We can solve it without difficulty with «Magento»:

Example, default theme is set for our store as “Default.”

The representation will be like this (Fig. 2.8):



Figure 2.8 – Frontend presentation with the theme default

Example, we need Spanish visitors to see difference not only in language when switch. To get this we can set a "blue" theme for Spanish presentation (Fig.2.9 and Fig.2.10).

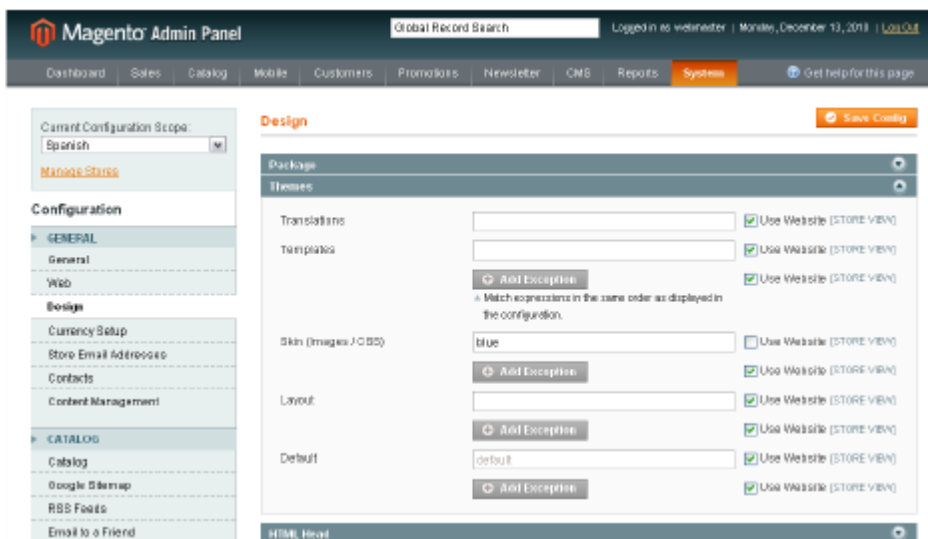


Figure 2.9 – Set default / blue theme for Spanish presentation



Figure 2.10 – Frontend of the Spanish-presentation with default / blue theme.

Design exclusion

Design exceptions allow us to set alternative themes depend on user-agent. Instead of making separate views for various devices, like smartphones or mobile phones (Fig.2.11).

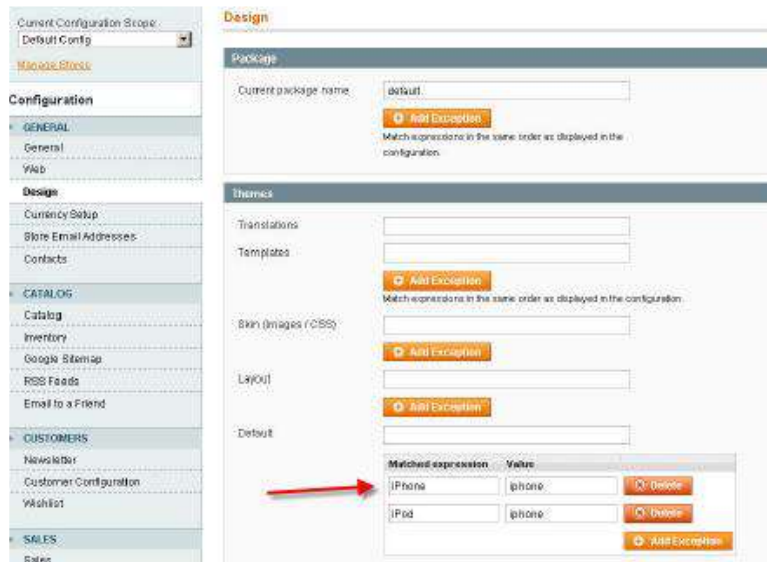


Figure 2.11 – Use of design exceptions

“Widgets”

In the “«Magento»”, widgets, as rule, are frontend blocks with a predefined set of configuration options

Even, they provide a great opportunity for business users except special technical knowledge is easy to add dynamic-content (example: product data) in the places provided in advance by designer or developer. This allow us to manage the creation information better plus more flexibly and marketing content through administrator tools allow us to intuitively and efficiently manage-content as:

- **Dynamic-data of marketing products of homepage campaigns.**
- **Dynamic-information, like recently viewed content pages.**
- **Advertising images in various blocks.**
- **Interactive elements and blocks (external review systems, subscription forms, voting and video chats).**
- **Alternative navigation elements, (tag-cloud, image-slider-catalogue).**
- **Interactive and dynamic Flash elements are easy to configure, embedded in content pages for better user perception.**

Terminology

One frontend unit is an element that makes a visual output or linking the visual structure, or produce the actual content.

«Magento» widget - a frontend unit that implements a special interface widget that allows us to have various configuration options for every instance block and have multiple independent instances of the blocks on pages.

Examples of “widgets”

«Magento» includes following “widgets” by default customize. New widgets can also be made by developers.

CMS-Page-Link: shows a link to the chosen system page content management, permit us to specify every text/title. There are two templates available for this widget - a string link and a block.

CMS-Static-Block: shows the content of the selected static block.

Catalog-Category-Link: shows a link to the selected category directory, gives us to specify every text/title. Two available templates: term, block.

Catalog-Product-Link: shows a link to selected product catalog and gives us to specify every text/title. Two available templates: block, term.

Recently-Compared-Products: shows the recent comparable goods. This widget gives us to specify the number of product display. Two available templates: list, grid.

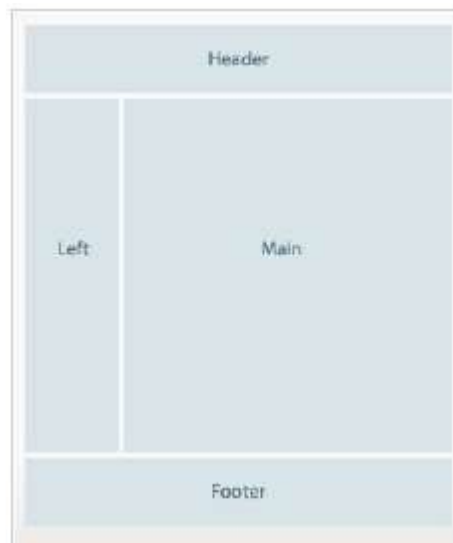


Figure 2.12 – Structural of blocks.

The diagram of classes of blocks for online store is given in fig. 2.13.

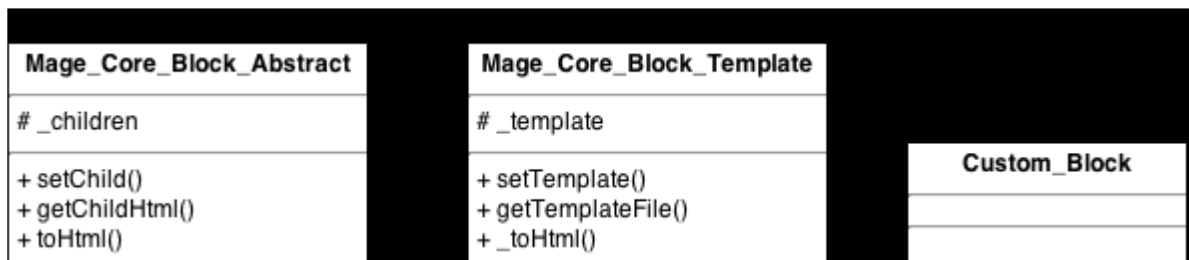


Figure 2.13 – Block class diagram

“Layouts”

Layouts: These usually files basically attach structural-blocks to content-blocks to. »Magento« layouts have 2 functions, structural, as rule, and content blocks, after that inform «Magento»: how and where we need to connect them (Fig.2.14).



Figure 2.14 – Example of binding blocks

“Layout” is a tool which we use to assign content blocks to structural blocks. »Magento« layouts are actually XML files. By changing layout we can move blocks around the page and set templates content blocks to make the layout of structural blocks.

**Base layout files are `app / design / frontend / base / default / layout`.
Layout files for custom theme should be in corresponding directory: `app / design / frontend / <package> / <theme> / layout`.**

Layout, as rule, builds of default-layout + usually layout-updates. With these updates, we can change the links between content and structural blocks without

difficulty and manage the functions of frontend, like how to load/unload a block-specific JavaScript on the page.

Every «Magento» module has its personal layout file, example

"Catalog.xml" is a layout file for directory module. Every file is divided on handles. Handles are determined by “«Magento» core” and activate additions. Most often, every handle corresponds to the type of page in store, example `<catalog_category_default>` , `<catalog_product_view>`, but there are handles apply to all pages, such as `<default>`, there are also corresponding ones status, like `<customer_logged_in>` / `<customer_logged_out>`.

Handle can be built in many layout file and blocks can be set to this reassigned to every layout file. Example, maximum layout files may hold a `<default>` handle. When parsing layout files, «Magento» 1st takes a layout update appoint to the `<default>` handle in every layout files by reading them in order fix in `app / modules / Mage_All.xml`. After that they understand page specific layout handles, also view every layout files for this handle.

The «Magento» page making system is made for ensuring free addition and removal of modules without affecting other modules. This means we can add, remove, move maximum of the functionality by simply adding, deleting, or moving a block of declarations to a file layouts.

Layouts have a small group of XML tags that work as detailed commands for application what can tell us how we can build a page that should be on its behaviour of every building block.

Handle is the identifier where the application determines what to do with updates that are in it.

If the file called `<default>`, then the application knows that updates must be downloaded on almost every store page before downloading page - specific layouts

(almost because some are exceptional pages, like a pop - up image of product, do not load the layout from <default> handle).

«Magento» describe the behaviour, visual representation of every building block page tag- <block>. We already mentioned 2 varieties of blocks – structural, content. The best way to see which one assigned to these blocks by tag attributes. Structural blocks normally contain the attribute "as" through the application can communicate with the corresponding area (by using getChildHtml method) in the template. The default layout consists usually structural blocks like: "left", "Right", "content" and "footer".

The following attributes are available for “blocks”:

Type is the module class identifier that describe the functionality of the block.

Name is the name of the block; other blocks can refer to the block by this name, “c” defines this attribute.

Before and after are two ways of positioning content blocks in structural blocks. Before = “-” and after = “-” are commands to position the block from the top or below the structural block.

Template - it describe the functionality of the block to be represented in this attribute. Example, if this attribute specifies 'catalog / category / view.phtml', the application will download template file 'app / design / frontend / template / catalog / category / view.phtml'.

Action - used to control functionality of the frontend, such as loading and unloading JavaScript.

As – by using this attribute the template calls the block. Example, the method <? = \$ This-> getChildHtml ('header')?> C template refers to "block as =" header ">.

<Reference> (link) is use to link to another block.

Linking to another update block inside the <reference> will be applied to the corresponding block.

We must use "name" attribute for link. This attribute is aimed at the "name" attribute of the <block> tag. Like reference <reference name = "right"> direction of block <block name = "right"> (Fig.2.15).

```

catalog.xml
</reference>
<reference name="right">
</reference>
</catalog_product_view>

page.xml
<block-type="core/text_list" name="right" as="right">
</block-type="page/html_footer" name="footer" as="1">
<block-type="core/store_switcher" name="store">
<block-type="cms/block" name="footer_links">
<action-method="setBlockId"><block_id>foot
</block>

```

Figure 2.15 – Example of using links.

You can enable hints to find the right layout faster templates in System -> Configuration -> Advanced -> Developer (Fig.2.16).

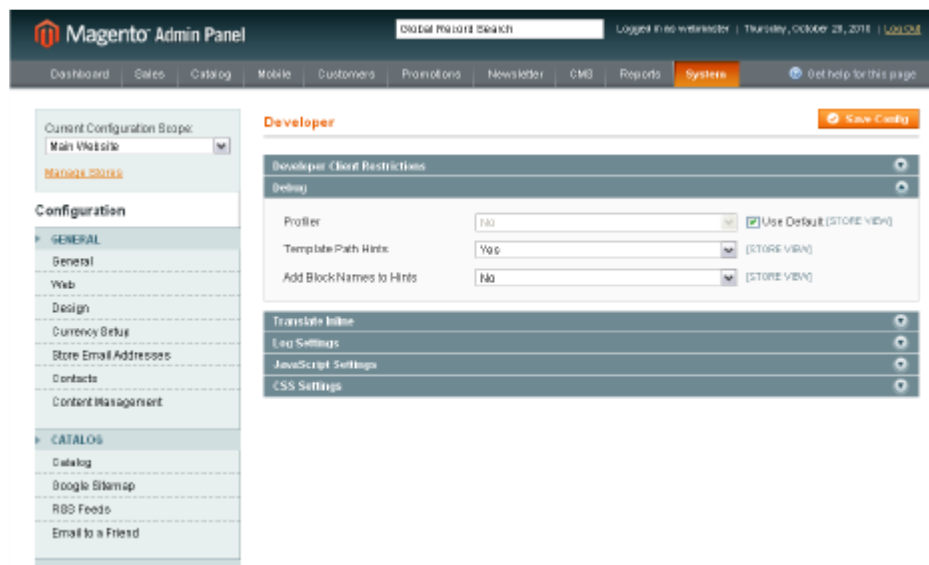


Figure 2.16 – Template hints in the admin panel

Previously, we looked at how you can change themes, including just a few modified files in a custom theme. This is possible due to the retreat mechanism. This method of changing topics is perfectly acceptable. But there is a more elegant one an

easier-to-support block-on / off approach that allows do this using only one file in the custom theme directory.

Instead of copying files from the base theme and deleting lines of code, you can just unlink the links to unwanted blocks. Additionally, save changes to one file makes it easier to track them. By the logic of «Magento», after connection and processing of all XML layouts and their updates, connects local layout file called "local.xml", it is connected to last, by this we can override all previous updates layouts.

File «Local.xml» has to be in

«App / design / frontend / <package> / <theme> / layout».

```
<? xml version = "1.0"?>
<layout>
  <default>
    <remove name = "cart_sidebar" />
  </default>
</layout>
```

The element of the file will built on the base theme, to delete the blocks names we can see the corresponding layout file. <Default> indicates global area, instead we can specify specific structural blocks like "left" or "Right".

"Local.xml" can be used to update the desired sites.

Example, to set a template with 2 columns on the left by default, we can use nodes which control the display of directory pages. Example, by adding the following lines to the local.xml file:

```

<catalog_category_default>
<reference name="root">
    <action method="setTemplate"><template>page/
    2columnsleft.phtml</template></action>
</reference>
</catalog_category_default>
<catalog_category_layered>
<reference name="root">
    <action method="setTemplate"><template>page/
    2columnsleft.phtml</template></action>
</reference>
</catalog_category_layered>

```

Using this approach, we can also add blocks and modify their position on the page, as well as their behavior, changing only one file, and without need to change layout source files.

Templates

Whereas layout files control the presence or absence of content blocks in the topic, the actual content of these blocks depends on the respective files templates. Most templates do not contain logic that determines whether they will be reflected. When the template is called, it is expected to be parsed and displayed.

Template files in «Magento» are PHTML files that contain HTML and PHP, which will be parsed and displayed by the appropriate browser.

It is generally recommended that the template files that need to be modified are copied to a custom theme with a slightly changed name. It saves the original template in case it is used in several places.

3 CREATING AN ONLINE STORE WEBSITE AND YOUR OWN MODULE BASED ON CMS «MAGENTO»

3.1 Installing the system on a local server

To create a website based on «Magento», you must first get distribution. The latest version of «Magento» Community Edition is available for download from the official site after free registration.

A local server is usually used for development. We will use WAMP, which includes Apache, MySQL and PHP, in advance configured for comfortable design. Download the latest from the official website

«Magento» version, unpacked. Also download, from there, demonstration products in the form of SQL dump file and image set. Place the unpacked distribution in the appropriate folder, which is located depends on the server settings. Create a new database called for example, "«Magento»". You can now import demo items into the database. The image must be placed in the "media" folder. You can configure the database to carry out by means of phpMyAdmin which is delivered complete with WAMP.

After setting up and starting the local server, we can continue system installation by opening the specified address in the browser. On the first page you can read the license. Next, set the localization settings, database connection, base site address, and administrative panel path and administrator credentials. This completes the initial setup and you can go to the admin panel, or to the main page of the site.

At this stage, we have actually received a working store. But for at the beginning of sales we need to fill it with your own products, configure methods of payment and delivery, and all of this can be done from administrative panel.

While administration and content filling, thanks to the understandable interface are not too difficult tasks, change the look or mechanisms of operation require a deeper understanding of the system.

3.2. Overview of implementation of MVC scheme

In typical MVC (ModelViewController) framework such as Codeigniter, all models are in 1 directory, all controllers are in another and all presentation in its own. But in «Magento» this isn't the case, they are grouped in a module, and every module plays role of a container for various functionality. Modules may composition of a model, controller, configuration unit (etc), auxiliary classes (Helper) and sql (description of database tables, migrations between versions).

In «Magento», for using modules correctly, 1st we need to connect and configure by following the appropriate descriptions in configuration XML files. Everyone's configuration files are read at beginning of workflow active modules, then combined into a single configuration tree. Next, analysis the URL of the corresponding module instantiated controller and required method is called. Correspondence of modules to a certain address is determined by referring to configuration files.

Unlike other MVC frameworks where controller is responsible for obtaining certain data from model and transmitting them to appropriate representation (view), the «Magento» controllers don't necessarily communicate with the models they have, the main role is to download and build (render) the layout without transmission data. Thus views (view - interfaces, representations) are divided into blocks, layouts and templates.

Blocks are PHP classes, the directory of the module, templates are

PHTML files and they are in the design directory. Every block is tied to layout template.

Layouts are XML files, where a design directory directories with layouts. As a result, system instantiates certain layouts blocks, associates them with templates. Thanks to these links, templates can be obtained required data by calling block methods, finally inserting the result in HTML. Once every templates are built, the page will transferred browser. Thus, «Magento» has an unusual implementation of MVC in it, use a thin controller to connect the layout (Fig. 3.1).

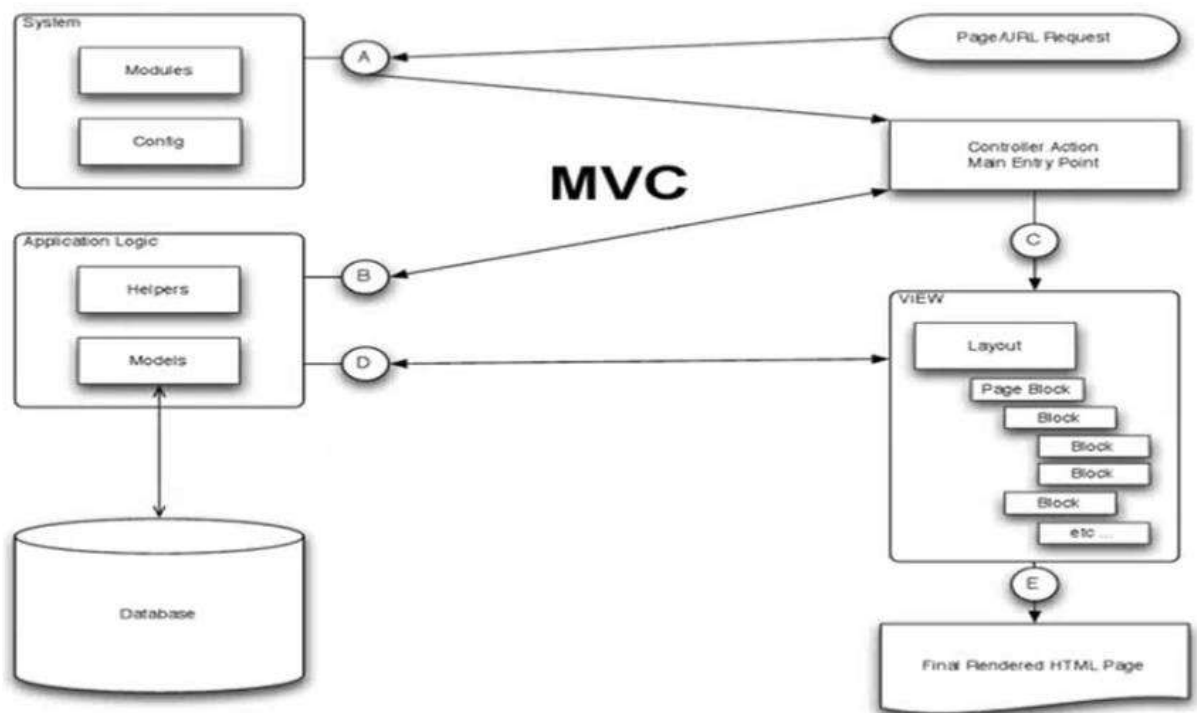


Figure 3.1 – Implementation of MVC in «Magento»

Legend:

- A. Parsing URLs. Controller definition and action.
- B. The actions of the controller manipulate the model.
- C. The action loads the layout and starts drawing the view.
- D. Templates through blocks receive data from models.
- E. Blocks and child blocks are drawn in the HTML page.

3.3. Rules for creating modules

Structure placing modules next: / app / code / [area code] / [namespace] / [module name].

In the "core" area of the code are the kernel files of the system, which are not worth it modify, because changes may be lost during the system upgrade. Third-party developers need to keep their specific files systems in the "local" area of the code, and the files to be shared with community - in the field of "community". Namespace can be arbitrary and frequent corresponds to the name of the developer company.

For example, let's create a module. Let's place it in the directory / app / code / local / Svetlana / Demo. To activate the module we need to create

XML configuration file in modules directory. Configuration name file can be arbitrary but traditionally it is called [namespace] _ [name module] .xml, so let's call the configuration file Svetlana_Demo.xml. To activate module, place the following text in the newly created file:

```
<config>
  <modules>
    <Svetlana_Demo>
      <active>true</active>
      <codePool>local</codePool>
    </Svetlana_Demo>
  </modules>
</config>
```

Then in the administrative panel on Configuration / Advanced page our module will appear.

Let's create file Product.php in directories / app / code / local / Svetlana / Demo / Model. It will house a classroom appropriate model. The class name must be [namespace] _ [module name] _ [type class] _ [file name], ie Svetlana_Demo_Model_Product. Such a name is necessary so that the autoloader can find the right class.

Let's create a new method, such as sayHello, which will just output some text for print. To test how our module works, create a test.php file with the following content:

```
require_once 'app/Mage.php';
Mage::app();
$product = new Svetlana_Demo_Model_Product;
$product->sayHello();
```

When we open corresponding page in browser, make sure that it is called by right method, it happened to the autoloader which is located in /lib/Varien/Autoload.php. But in «Magento» the necessary modules it's better to use other design.

3.4. Pattern of factory

Abstract, as rule, Factory is a design template that provides encapsulation of separate factories in a single scheme, "hiding" them detailing. It belongs to class of building design patterns.

In typical applications, client code creates specific implementation of an abstract factory, after that uses a general universal factory interface, to build instances of objects which are the part of schema.

The client code "doesn't know", which specific objects it will get from factories because it uses a universal interface to build them. Template differentiate the details of implementation of many objects from normal use in code, because the building of object is carried out by following the methods provided by factory interface.

We need to use Abstract Factory template when:

system should not depend on how they are formed, composed the objects included it presented;

Those in family must have interconnected objects□to use together and it's necessary to ensure depiction of this limitation;

system have to configured with one of the families of objects that□build it up;

It's necessary to submit (present) a library of objects, disclosing only their interfaces, but not the implementation.

Class diagram of abstract factory is shown in Fig. 3.2.

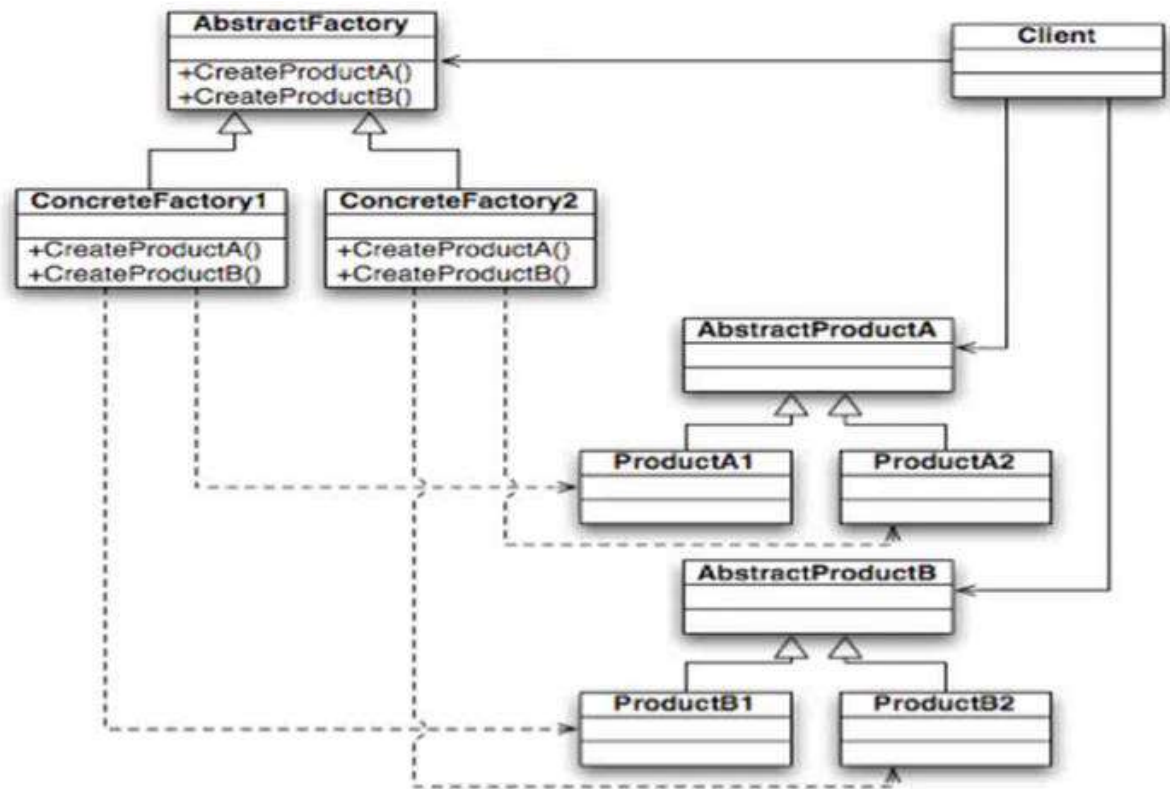


Figure 3.2 –Abstract factory

- **Abstract Factory:**
 - announces interface for operations which make abstract objects-products;
- **Specific product:**
 - defines object-product made by corresponding concrete factory;
 - implements Abstract Product interface;
- **Client:**
 - Uses only interfaces which are declared in classes *Abstract Factory* and *Abstract Product*.

Usually, a single instance is made (instantiated) during execution

Concrete Factory class.

This particular factory makes facilities that have enough defined implementation. To build other types of objects, client must use another specific factory.

AbstractFactory entrusts the building of product objects to its subclass

Concrete Factory.

The "Factory" pattern allows us to flexibly change function of default with configuration files. For example, instead `$ customer = new Mage_Customer_Model_Session` uses this design `$ customer = Mage::getModel("customer / session")`.

The system finds the desired file due to the configuration the tree in which the directory name with model classes is written. So, by changing the configuration tree, you can force the system to connect custom classes without having to make changes to the system kernel.

In order for `Mage::getModel ()` to be able to connect our model, let's create `/app/code/local/Svetlana/Demo/etc/config.xml` file with the following content:

```
<config>
  <global>
    <models>
      <demo>
        <class>Svetlana_Demo_Model</class>
      </demo>
    </models>
  </global>
</config>
```

Now challenge `Mage::getModel ('demo / product')` equivalent new `Svetlana_Demo_Model_Product`.

Blocks and helpers are described in the same way in the configuration files (helpers) in the corresponding nodes of the configuration tree (blocks and helpers).

Let's create a helper `Svetlana_Demo_Helper_Customer`, which will be in `/app/code/local/Svetlana/Demo/Helper/Customer.php`. Often custom helpers inherited from `Mage_Core_Helper_Abstract`.

Unlike models and blocks, helpers are not called through `getModel` or `getBlock`, and through `Mage::helper`, for example for our helper the call will be `Mage::helper('demo / customer')`.

There is another feature when calling helpers: if you do not specify a name file, such as `Mage::helper('demo')`, the class will be called

`Svetlana_Demo_Helper_Data`.

For example, change the `getChildren ()` method to `Mage_Catalog_Model_Category`, this method returns direct descendants of a category as a string separated by commas, and we, for example, need to get an array.

Do not modify the kernel file, because changes can be lost when system updates. We assume that «Magento» connects files in such order: local, community and core.

Therefore, you can copy `/app/code/core/Mage/Catalog/Model/Category.php` `/app/code/local/Mage/Catalog/Model/Category.php` and make changes already there. But when system updates can add new features and fix bugs in existing.

There is another way of changing functionality of the kernel, which gives us to get these renewal. To do this, we need to make our own model class. Example, `Svetlana_Catalog_Model_Category`, which will follow kernel class.

In order to connect the system class instead of the kernel class, we need to file `etc / config.xml`, which is in our module to prescribe:

```

<config>
  <global>
    <models>
      <catalog>
        <rewrite>
          <category>Svetlana_Catalog_Model_Categor</category>
        </rewrite>
      </catalog>
    </models>
  </global>
</config>

```

Blocks/helpers can be modified in the same way.

3.5. Events

Another important tool, allows us to change behavior of system default - events. Events are referenced in many places in system.

Event-driven programming (POP) is a programming paradigm where program execution is determined by user actions (keyboard, mouse), messages from another programs and threads, operating system events (example, network packet receipt).

POP can be defined as way to build a computer program, where the code (usually in main function of the program) explicitly main cycle of the program, body consists of two parts is allocated: receiving event notification, event processing.

As a rule, it is unacceptable for a long time execution of the event handler in real task, because the program can't respond to other events. For this, when writing event-oriented programs often use automatic programming.

Event-oriented programming is normally used in three cases:

- when building user interfaces (including graphic);
- when building server applications, if any or other reasons, it's undesirable to generate service processes;

When programming games that are controlled significantly number of objects.

Event-oriented programming used in server programs to solve the problem of scaling 10,000 simultaneous connections and more.

In servers built on model of "one thread per connection", problems with scalability arise for following reasons:

Excessive overhead for operating data structures □ systems needed to describe a single task (task status segment, □ stack, etc.);

Context switching overheads are too high.

Philosophical prerequisite for abandoning the server streaming model

Alan Cox's statement may be: "A computer is a finite automaton.

Stream programming is required for those who don't know how to program finishes automata ».

Server application for event-oriented programming is implemented based on system call that receives the event message simultaneously from many descriptors (multiplexing). When processing events only non-blocking I / O operations are used, if not one descriptor did not interfere with event processing from other descriptors.

It is better to use events than to imitate classes, if that gives the ability to achieve the desired result, because it is preserved update support.

To register an observer who will listen to the event, you need to make an entry in the configuration tree. In module configuration file a node is created, example for the event "customer_login":

```

<config>
  <global>
    <events>
      <customer_login>
        <observers>
          <demo>
            <type>model</type>
            <class>demo/observer</class>
            <method>logCustomer</method>
          </demo>
        </observers>
      </customer_login>
    </events>
  </global>
</config>

```

The type can be model or singleton.

Singleton in «Magento» is somewhat different from others. When calling a method `getSingleton ()` returns the same object, but can still be retrieved other objects by calling `getModel ()` method.

3.6. Database model

Here is a difference between «Magento» and other CMSs in how to store data.

The design of database is quite unique. Database contains about 300 tables.

Normally, models communicate with database, presenting data in object-oriented form, perform business logic.

For greater abstraction, «Magento» uses a three-layer approach to models, ie we have: models of entities, models of resources and collections.

The entity model executes a certain business logic, but also provides access to CRUD (create, read, update, delete) operations, which is actually achieved through resource model. Thus entity models do not contain things related to database, like connecting or querying database, instead each model uses the appropriate resource

model which communicates with the database through adapter and performs all CRUD tasks.

Separating logical model (essence) and code that communicates with database resources and adapters), it's possible to write new adapters for different databases at immutable models, and this is the biggest advantage of this approach.

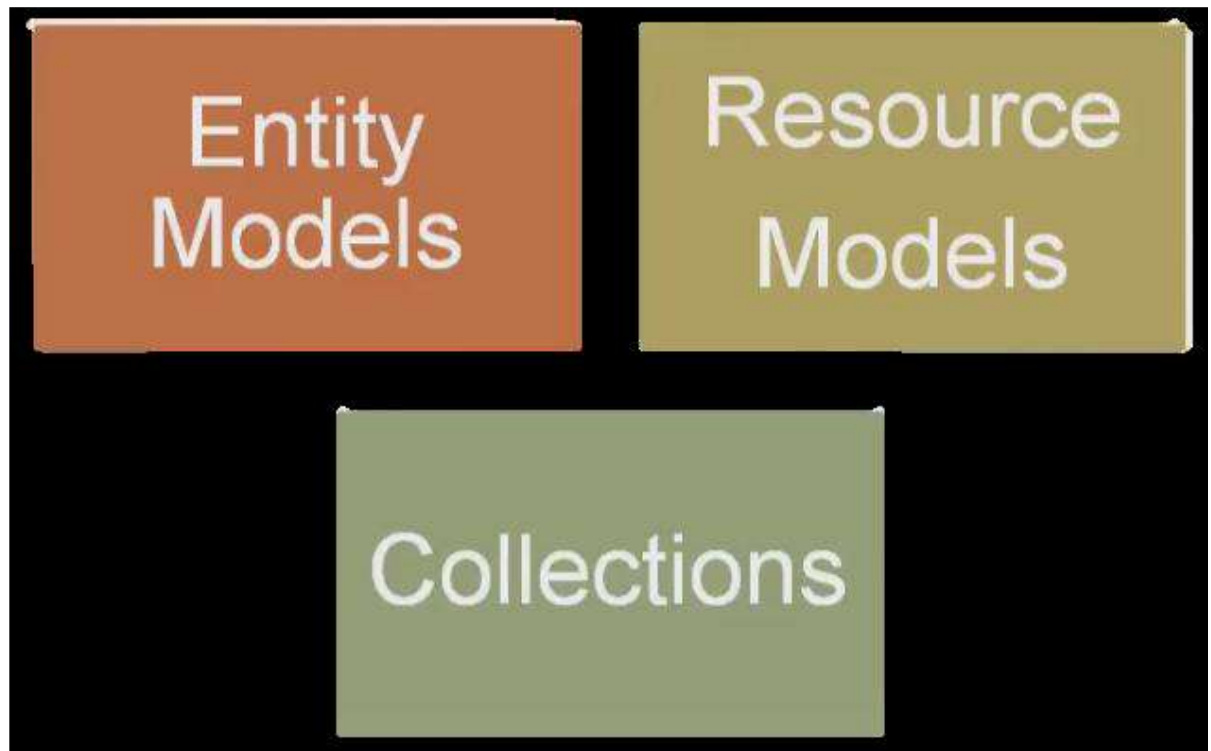


Figure 3.3 –Three-layer model

There are different types of models.

In simple models, each entry in database is a separate entity. Example, in configuration tables or URL Rewrite, where each row represents a complete information about the essence. This approach is often used in other systems.

There are also more complex models such as CMS pages, or even more complex ones such as sales quotes and orders. In them for a complete record entities need to view multiple tables.

Here is an EAV (entity-attribute-value) model also. Class diagram is given in Appendix A. Such models are used when representation table is a sparse matrix. A sparse matrix is a matrix, maximum elements of which are zeros. Examples of this kind of models are products, categories, buyers. They are the most flexible, but with some overhead costs. Products and categories are stored in almost 50 tables. That's why EAV is very complex parts of the system. But you don't have to know the details, thanks to «Magento» ORM (object-relational mapping, object-relational display), we can work with database using objects.



Figure 3.4 –Types of models

EAV model.

Entity - that is modeled, like a product or category.

Attribute - property of the entity.

Value - value of attribute.

Consider the usual table of users (Table. 3.1).

Table. 3.1.

Normal user table

customer_id	username	Password	first_name	last_name
1	milanche	121hjfd14588...	Milan	Stojanov
2	t_anja	5484dsd8464...	Tanja	Simic
3	vstoiljkovic	Sde454adf45...	Vladimir	Stoiljkovic

Suppose we need to have a large number of columns for users but not all of them will be filled for each of them. As a result, we get table with lots of empty values. EAV solves this problem. The above table can be replaced by EAV with three tables:

Tables. 3.2.

EAV user tables

customer_id	username	password
1	milanche	121hjfd14588...
2	t_anja	5484dsd8464...
3	vstoiljkovic	Sde454adf45...

customer_entity

attribute_id	Name	display	type
1	first_name	First	varchar
2	last_name	Last	varchar

eav_attribute

entity_id	attribute_id	value
1	1	Milan
1	2	Stojanov
2	1	Tanja
2	2	Simic
3	1	Vladimir
3	2	Stoiljkovic

customer_varchar

In first table we have basic data which are required. In table attributes stored a kind of meta-information about entity. Its stored attributes that every entity can have. Adding a new attribute comes down to simply adding a new row to the attribute

table in the table values. The values of certain attributes for certain entities are stored. For each type of value has its own table.

In some modules you can find the Mysql4 directory, it is there for backward compatibility, because support for multiple databases was added only in version 1.6. Prior to that, MySQL-specific resources were present in the resource models code. Now they contain only code that does not depend on database, and all specific requests are formed in adapters.

Resource models are configured in same node of configuration tree, as corresponding resource model, next to its class name, node with another name node in which the class of resource model is specified.

```
<cms>
  <class>Mage_Cms_Model</class>
  <resourceModel>cms_resource</resourceModel>
</cms>
<cms_resource>
  <class>Mage_Cms_Model_Resource</class>
  <deprecatedNode>cms_mysql4</deprecatedNode>
  <entities>
    <page>
      <table>cms_page</table>
    </page>
    <page_store>
      <table>cms_page_store</table>
    </page_store>
    <block>
      <table>cms_block</table>
    </block>
    <block_store>
      <table>cms_block_store</table>
    </block_store>
  </entities>
</cms_resource>
```

While `Mage::getModel ('cms / page')` refers to `Mage_Cms_Model_Page`, `Mage::getResourceModel ('cms / page')` refers to `Mage_Cms_Model_Resource_Page`.

But at the same time to link model with its resource model is required call the `_init ('cms / page')` method, which is usually called in constructor object and defined in `Mage_Core_Model_Abstract`, from which most classes models are inherited. The

method of same name is in resource model at the time startup reads the configuration tree, which spells out the names of tables in database.

Thanks to «Magento» ORM it is possible to receive data from a DB in a convenient way, for example we get some product data:

```
$product = Mage::getModel('catalog/product')->load(42);
$metaTitle = $product->getData('meta_title');
$product->setData('meta_title', 'updated');
$product->save();
```

Collections are used to obtain data on several products at once.

Collections do not need to be described in the configuration, but we need to create a file `Resource / [resource name] /Collection.php`, which contains collection model accordingly, example for appropriate goods `/app/code/core/Mage/Catalog/Model/Resource/Product/Collection.php`.

To obtain product names, for example:

```
$products = Mage::getModel('catalog/product')->getCollection()
    ->addAttributeToSelect(array('name', 'price'))
    ->addFieldToFilter('price', array('lt'=>50));
echo count($products) . '<br>';
foreach($products as $product)
{
    echo $product->getData('name');
    echo '<br>';
}
```

Diagram of collection classes is shown in Figure 3.5 [15-18].

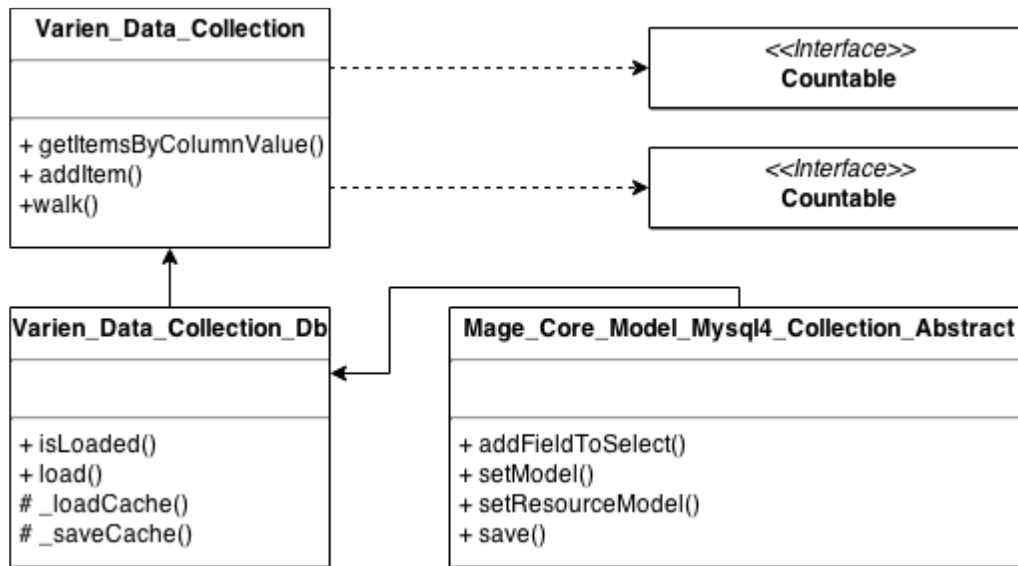


Figure 3.5 – Chart diagram for collections

It needs to be noted that without calling over the collection object of method "AddAttributeToSelect" returns only sample from base table EAV. For its recommended to increase the speed increase only the required samples attributes.

3.7. Addressing and controllers

Normally all queries are redirected to `index.php` which is in the root directory of «Magento» installation.

Sequence diagrams, which show process of request processing browser, shown in Fig.3.6 and Fig.3.7.

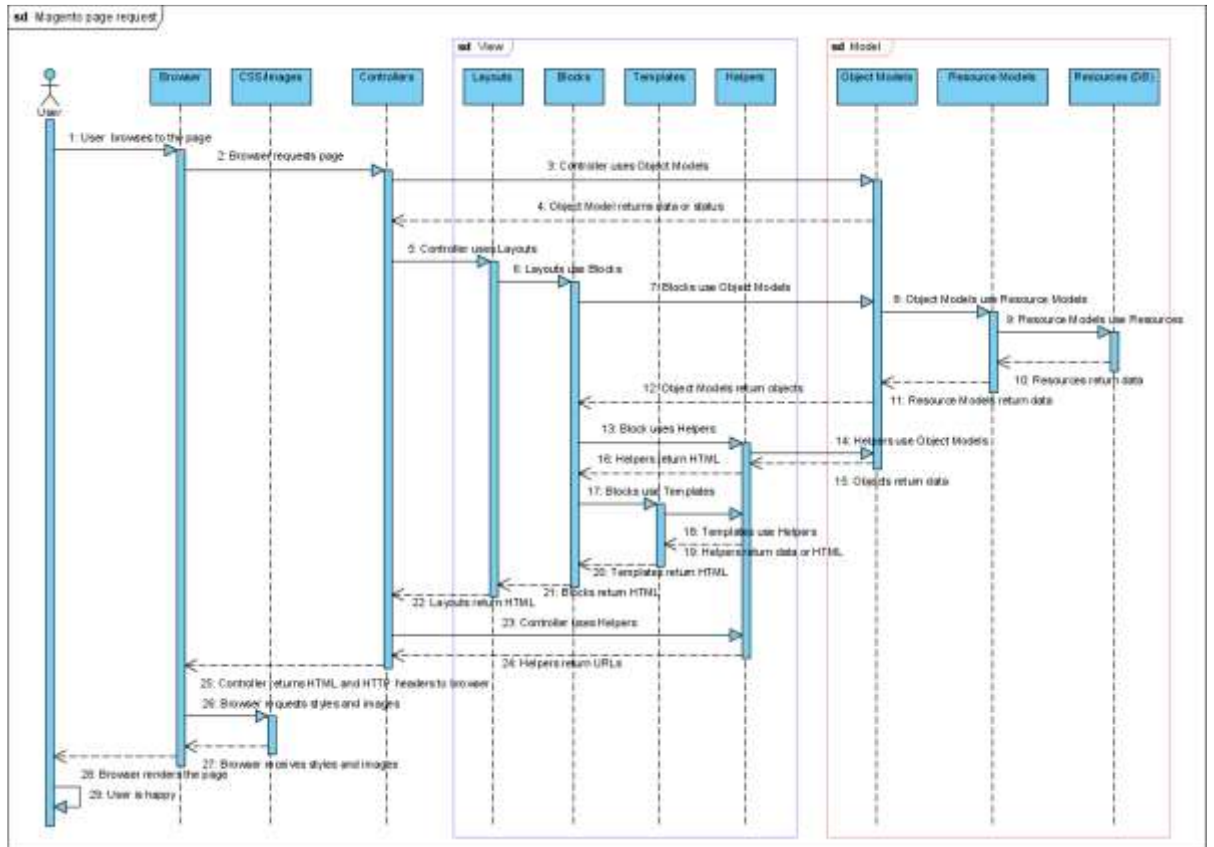


Figure 3.6 – Browser request processing sequence diagram

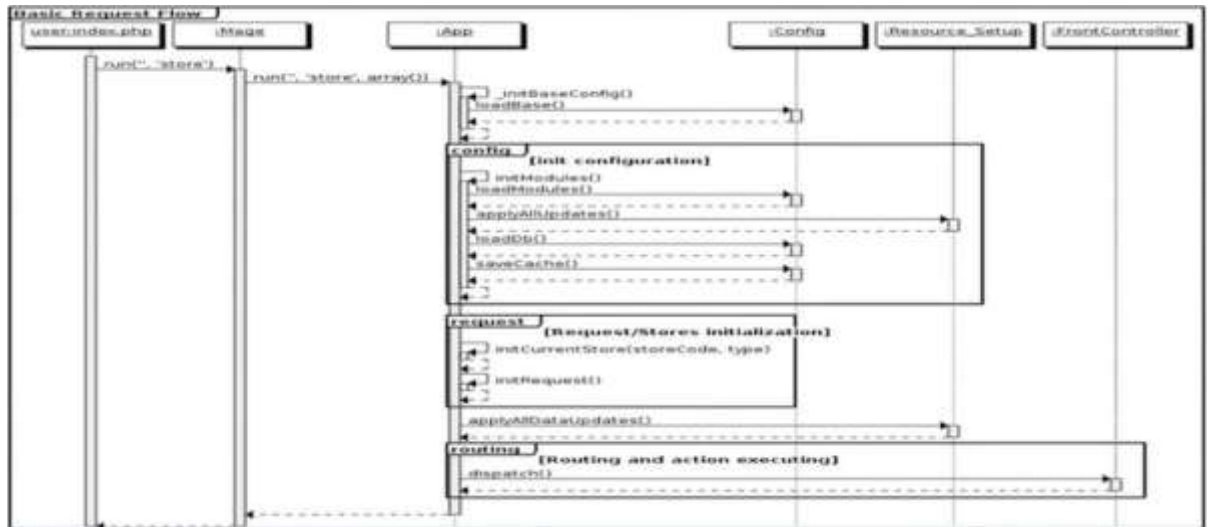


Figure 3.7 – Browser request processing sequence diagram

If no specific address is specified, in other words a request has been made home page of site, request is processed by the controller which is defined in settings. We can change these settings in admin panel at

System / Configuration / Web page on the Default Pages tab.

To call a specific controller using standard router, we need to make a request to address [address site] /index.php/ [frontName] / [controller name] / [action name]. If name of the controller or action name isn't specified, index is substituted instead. Every module can be set to frontName in configuration node <router> < [unique name]> <args> <frontName>, example:

```
<router>
  <Svetlana_demo>
    <use>standard</use>
    <args>
      <module>Svetlana_Demo</module>
      <frontName>demo</frontName>
```

Default home page controller is "cms".

In addition to standard router, others can be specified. So it is possible to have a convenient address, example, for product page - [store address] / [name goods].

Controller can retrieve requested data using method `getRequest ()`.

Most often, everything that happens in controller is a call to methods `loadLayout ()` and `renderLayout ()` to load and create a layout.

Depending on conditions, like the page address, the title of the presentation store, user type and others, layouts are generated during download layout handles, which are then used to select nodes layouts to build and display.

When building a layout (`render Layout`), the necessary blocks are connected and templates required for display.

Layouts consist mainly of blocks. Each block in layout has two required attributes, type and name. Type indicates name of class block, and name is use to refer the block. Exist a special block called "root" block, it usually has an attribute "Output" with the value "toHtml", which indicates to system that block should show the output to the browser.

The template can be connected to a block in block itself, or using special attribute "template", indicating its value path to template relative to theme directory.

Each template is HTML with some calls PHP methods such as "\$ this->getChildHtml ('head')". "\$ This" indicates current one block, "head" corresponds to value of attribute "as" of one of its child blocks in layout. This way blocks call other blocks until page is gone built.

There are other blocks that don't use templates - text lists (text_list). All child blocks are automatically built in these blocks. They are kind of containers for other units. Such blocks include "left", "right" and "Content". Automatic construction of child blocks allows us to refer to these blocks from your layout update files, and unnecessarily modify parent block template file. Thus, inclusion of turning off the module with layout connected in this way won't affect site performance. Another child element of block can be "action" (action) with the "method" attribute, that's used to call block methods.

3.8. Creating a module

To show how everything works together, let's build a module that will be display all configurable products. Our module will have a name "Configurable".

Activate module by placing file Svetlana_Configurable.xml in /app/etc/modules.


```

<config>
  <modules>
    <Svetlana_Configurable>
      <active>true</active>
      <codePool>local</codePool>
    </Svetlana_Configurable>
  </modules>
</config>

```

In local configuration file `/app/code/local/Svetlana/Configurable/etc/config.xml` set:

"frontName" is "configurable".

Class directories:

For blocks - "Svetlana_Configurable_Block"

For helpers - "Svetlana_Configurable_Helper"

Layout update file - Svetlana_configurable.xml

```

<config>
  <global>
    <blocks>
      <configurable>
        <class>Svetlana_Configurable_Block</class>
      </configurable>
    </blocks>
    <helpers>
      <configurable>
        <class>Svetlana_Configurable_Helper</class>
      </configurable>
    </helpers>
  </global>
  <frontend>
    <routees>
      <Svetlana_configurable>
        <use>standard</use>
        <args>
          <module>Svetlana_Configurable</module>
          <frontName>configurable</frontName>
        </args>
      </Svetlana_configurable>
    </routees>
    <layout>
      <updates>
        <configurable>
          <file>Svetlana_configurable.xml</file>
        </configurable>
      </updates>
    </layout>
  </frontend>
</config>

```

Let's create a class "Svetlana_Configurable_Block_Configurable" with a method "GetConfigurableProducts", which will return a collection of products from a type identifier equal to "configurable".

```
{  
    public function getConfigurableProducts()  
    {  
        $products = Mage::getModel('catalog/product')  
            ->getCollection()  
            ->addAttributeToSelect(array('name', 'price', 'url_key'))  
            ->addAttributeToFilter('type_id', array('eq'=>'configurable'));  
        return $products;  
    }  
}
```

Let's create an index controller with index action, which will be simple download and build layout.

```
<?php  
class Svetlana_Configurable_IndexController extends  
    Mage_Core_Controller_Front_Action  
{  
    public function indexAction()  
    {  
        $this->loadLayout();  
        $this->renderLayout();  
    }  
}
```

Let's create a helper "Svetlana_Configurable_Helper_Data" with a method "BeautifyPrice" to format the output of prices.

```

<?php
class Svetlana_Configurable_Helper_Data extends
    Mage_Core_Helper_Abstract
{
    public function beautifyPrice($price)
    {
        if(!isset($price))
        {
            return 'unknown price';
        }
        return number_format($price, 2, '.', ',');
    }
}

```

Let's create a variation of theme and enable it in administrative panel.

In the directory of newly built topic:

Build a layout file "layout / Svetlana_configurable.xml", in which for anchor "Svetlana_configurable_index_index", which corresponds to index page of our controller, add a child block to block "content", assigning him template "Svetlana / configurable.phtml".

```

<layout>
    <Svetlana_configurable_index_index>
        <reference name="content">
            <block type="configurable/configurable" name="
                configurable" template="Svetlana/configurable.phtml"/>
        </reference>
    </Svetlana_configurable_index_index>
</layout>

```

Build a template file "template / Svetlana / configurable.phtml", in which display an unordered list of links to received goods together with their prices.

```

<?php
$products = $this->getConfigurableProducts();
$helper = Mage::helper('demo');
?>
<h3>Products</h3>
<?php if($products->count() > 0): ?>
    <ul>
        <?php foreach($products as $product): ?>
            <li><a href="<?php echo $product->getProductUrl(); ?>"><?php
                echo $product->getData('name'); ?></a>
            <span><?php echo $helper->beautifyPrice($product->getPrice()); ?></span></li>
        <?php endforeach; ?>
    </ul>
<?php else: ?>
    <p>No products are available!</p>
<?php endif; ?>

```

Now, going to the page [site address] / configurable, we get list of all configurable products (Fig. 3.8).

The screenshot shows the Madison Island website interface. At the top, there is a navigation bar with 'YOUR LANGUAGE: English' and 'WELCOME'. Below the navigation bar, the 'MADISON ISLAND' logo is prominently displayed. To the right of the logo, there are links for 'ACCOUNT' and 'CART', and a search bar with the placeholder text 'Search entire store here'. Below the navigation bar, there are several menu items: 'WOMEN', 'MEN', 'ACCESSORIES', 'HOME & DECOR', 'SALE', and 'VIP'. The main content area is divided into three columns. The left column is titled 'POPULAR TAGS' and lists 'grey, blue, red, white' and 'View All Tags'. The middle column is titled 'PRODUCTS' and lists various items with their prices: 'French Cuff Cotton Twill Oxford 190.00', 'Slim Fit Dobby Oxford Shirt 175.00', 'Raid Cotton Shirt 160.00', 'Sullivan Sport Coat 510.00', 'Linen Blazer 455.00', 'Stretch Cotton Blazer 490.00', 'Chelsea Tee 75.00', 'Chelsea Tee 75.00', 'Chelsea Tee 75.00', 'Merino V-neck Pullover Sweater 210.00', 'Lexington Cardigan Sweater 240.00', 'Core Striped Sport Shirt 125.00', 'Bowery Chino Pants 140.00', 'The Essential Boot Cut Jean 140.00', 'Flat Front Trouser 195.00', 'NoLita Cami 150.00', and 'The Tank an on'. The right column is titled 'RECENTLY VIEWED PRODUCTS' and shows a 'LINEN BLAZER' with a small image. Below this, there is a 'COMPARE PRODUCTS' section with the text 'You have no items to compare.' and a 'COMMUNITY POLL' section titled 'What is your favorite color?' with radio buttons for 'Green', 'Red', 'Black', and 'Magenta', and a 'VOTE' button.

Figure 3.8 –List of configurable products

3.9. Creating themes

Many «Magento» themes, even commercial ones, are created by modification files or redefining blocks that are defined in base package. Also it's possible to build a new theme from scratch without using structural and content units that come with «Magento». It is not necessary use header, footer, left, right and content as structural containers. Of course, creating a theme from scratch is more difficult and time consuming, but depending on the project or other needs, this may be the right approach.

Before we start building a mark-up for a store, we need to decide on structural type of pages, example:

Home page will have a three-column structure.

Category list page - two columns with the right column.

User pages - two columns with a left column.

You can then create a markup for each structural type and save her skeletal templates in `gmail app / design / frontend / design_package / theme_variation / template / page /`.

The skeletal pattern is named because of its purpose, because all it contains (except for `<head>` elements) is a presentation markup which is used for positioning of structural blocks in the relevant areas.

```
<html>
<head></head>
<body>
<div class="header"><?=$this->getChildHtml('header') ?></div>
<div class="middle">
  <div class="col-left"><?=$this->getChildHtml('left') ?></div>
  <div class="col-main"><?=$this->getChildHtml('content') ?></div>
</div>
<div class="footer"><?=$this->getChildHtml('footer') ?></div>
</body>
</html>
```

Figure 3.9–Example of a skeleton theme file

Inside the presentation markup you can see a call to the PHP method «<? = \$ This-> getChildHtml ()?>» (See Fig.3.9). So «Magento» loads structural blocks in skeletal patterns. Such calls are made by name relevant structural blocks. Skeleton templates are assigned to the store by using the layout.

After creating the skeletal templates, you need to create a template for of each content block.

«Magento» loves meaningful templates. You need to break page layout according to functionality of the page. Example, if the design provides mini-basket, markup of this area should be placed in a separate template file.

Now we proceed to create the layout.

There are two types of layouts - default and update. Layout for the default (page.xml) is commonly used on most pages shop. Rest of the layout files are updates that are simply updated the default layout for certain pages.

Returning to the created module, we have as a layout for the default is three columns that will be used on most pages. But for the product page we need a different structure, namely two columns with right column. To achieve this, create a "catalog.xml" in which we will place some markup commands to load the structure from the two columns instead of three. This process is called updating the layout.

```
<reference name="root">
<action method="setTemplate"><template>page/2columns-right.phtml</template>
</action >
</reference>
```

We need to have a block in the right column by default subscriptions to newsletters, but not on the pages of the user's personal account. In this case, update file layout of the relevant pages (customer.xml) we need to place commands to remove the content block mailings.

4 OCCUPATIONAL HEALTH AND EMERGENCY SAFETY

4.1 Effects of electromagnetic radiation on the human body

A large body of literature exists on the response of tissues to electromagnetic fields, primarily in the extremely-low-frequency (ELF) and microwave-frequency ranges. In general, the reported effects of radiofrequency (RF) radiation on tissue and organ systems have been attributed to thermal interactions, although the existence of nonthermal effects at low field intensities is still a subject of active investigation. This chapter summarizes reported RF effects on major physiological systems and provides estimates of the threshold specific absorption rates (SARs) required to produce such effects. Organ and tissue responses to ELF fields and attempts to characterize field thresholds are also summarized. The relevance of these findings to the possible association of health effects with exposure to RF fields from GWEN antennas is assessed.

Nervous System

The effects of radiation on nervous tissues have been a subject of active investigation since changes in animal behavior and nerve electrical properties were first reported in the Soviet Union during the 1950s and 1960s.¹ RF radiation is reported to affect isolated nerve preparations, the central nervous system, brain chemistry and histology, and the blood-brain barrier.

In studies with in vitro nerve preparations, changes have been observed in the firing rates of *Aplysia* neurons and in the refractory period of isolated frog

sciatic nerves exposed to 2.45-GHz microwaves at SAR values exceeding 5 W/kg.^{2,3,4} Those effects were very likely associated with heating of the nerve

preparations, in that much higher SAR values have not been found to produce changes in the electrical properties of isolated nerves when the temperature was controlled.^{5, 6} Studies on isolated heart preparations have provided evidence of bradycardia as a result of exposure to RF radiation at nonthermal power densities,⁷ although some of the reported effects might have been artifacts caused by currents induced in the recording electrodes or by nonphysiological conditions in the bathing medium.^{8,9,10} Several groups of investigators have reported that nonthermal levels of RF fields can alter Ca²⁺ binding to the surfaces of nerve cells in isolated brain hemispheres and neuroblastoma cells cultured in vitro (reviewed by the World Health Organization¹¹ and in Chapters 3 and 7 of this report). That phenomenon, however, is observed only when the RF field is amplitude-modulated at extremely low frequencies, the maximum effect occurs at a modulation frequency of 16 Hz. A similar effect has recently been reported in isolated frog hearts.¹² The importance of changes in Ca²⁺ binding on the functional properties of nerve cells has not been established, and there is no clear evidence that the reported effect of low-intensity, amplitude-modulated RF fields poses a substantial health risk.

Results of in vivo studies of both pulsed and continuous-wave (CW) RF fields on brain electrical activity have indicated that transient effects can occur at SAR values exceeding 1 W/kg.^{13,14} Evidence has been presented that cholinergic activity of brain tissue is influenced by RF fields at SAR values as low as 0.45 W/kg.¹⁵ Exposure to nonthermal RF radiation has been reported to influence the electroencephalograms (EEGs) of cats when the field was amplitude-modulated at frequencies less than 25 Hz, which is the range of naturally occurring EEG frequencies.¹⁶ The rate of Ca²⁺ exchange from cat brain tissue in vivo was observed to change in response to similar irradiation conditions.¹⁷ Comparable effects on Ca²⁺ binding were not observed in rat cerebral tissue exposed to RF radiation,¹⁸ although the fields used were pulsed at EEG

frequencies, rather than amplitude-modulated. As noted above, the physiological significance of small shifts in Ca^{2+} binding at nerve cell surfaces is unclear.

A wide variety of changes in brain chemistry and structure have been reported after exposure of animals to high-intensity RF fields.¹⁹ The changes include decreased concentrations of epinephrine, norepinephrine, dopamine, and 5-hydroxytryptamine; changes in axonal structure; a decreased number of Purkinje cells; and structural alterations in the hypothalamic region. Those effects have generally been associated with RF intensities that produced substantial local heating in the brain.

Extensive studies have been carried out to detect possible effects of RF radiation on the integrity of the blood-brain barrier.^{20,21} Although several reports have suggested that nonthermal RF radiation can influence the permeability of the blood-brain barrier, most of the experimental findings indicate that such effects result from local heating in the head in response to SAR values in excess of 2 W/kg. Changes in cerebral blood flow rate, rather than direct changes in permeability to tracer molecules, might also be incorrectly interpreted as changes in the properties of the blood-brain barrier.

Effects of pulsed and sinusoidal ELF fields on the electrical activity of the nervous system have also been studied extensively.^{22,23} In general, only high-intensity sinusoidal electric fields or rapidly pulsed magnetic fields induce sufficient current density in tissue (around 0.1-1.0 A/m² or higher) to alter neuronal excitability and synaptic transmission or to produce neuromuscular stimulation. Somewhat lower thresholds have been observed for the induction of visual phosphenes (discussed in the next section) and for influencing the electrical activity of *Aplysia* pacemaker neurons when the frequency of the applied field matched the endogenous neuronal firing rate.²⁴ Those effects, however, have been observed only with ELF frequencies and would not be expected to occur at the higher frequencies associated with GWEN transmitters. Recent studies with human volunteers exposed to 60-Hz electric and magn.

Electromagnetic radiation can be classified into two types: ionizing radiation and non-ionizing radiation, based on the capability of a single photon with more than 10 eV energy to ionize oxygen or break chemical bonds. Ultraviolet and higher frequencies, such as X-rays or gamma rays are ionizing, and these pose their own special hazards: see radiation and radiation poisoning. By far the most common health hazard of radiation is sunburn, which causes over one million new skin cancers annually.

4.2 Types of hazards

Electrical hazards

Very strong radiation can induce current capable of delivering an electric shock to persons or animals.[citation needed] It can also overload and destroy electrical equipment. The induction of currents by oscillating magnetic fields is also the way in which solar storms disrupt the operation of electrical and electronic systems, causing damage to and even the explosion of power distribution transformers, blackouts (as occurred in 1989), and interference with electromagnetic signals (e.g. radio, TV, and telephone signals).

Fire hazards

Extremely high power electromagnetic radiation can cause electric currents strong enough to create sparks (electrical arcs) when an induced voltage exceeds the breakdown voltage of the surrounding medium (e.g. air at 3.0 MV/m). These sparks can then ignite flammable materials or gases, possibly leading to an explosion.

This can be a particular hazard in the vicinity of explosives or pyrotechnics, since an electrical overload might ignite them. This risk is commonly referred to as Hazards

of Electromagnetic Radiation to Ordnance (HERO) by the United States Navy (USN). United States Military Standard 464A (MIL-STD-464A) mandates assessment of HERO in a system, but USN document OD 30393 provides design principles and practices for controlling electromagnetic hazards to ordnance.

On the other hand, the risk related to fueling is known as Hazards of Electromagnetic Radiation to Fuel (HERF). NAVSEA OP 3565 Vol. 1 could be used to evaluate HERF, which states a maximum power density of 0.09 W/m^2 for frequencies under 225 MHz (i.e. 4.2 meters for a 40 W emitter)/

Biological hazards

The best understood biological effect of electromagnetic fields is to cause dielectric heating. For example, touching or standing around an antenna while a high-power transmitter is in operation can cause severe burns. These are exactly the kind of burns that would be caused inside a microwave oven.[citation needed]

This heating effect varies with the power and the frequency of the electromagnetic energy, as well as the distance to the source. A measure of the heating effect is the specific absorption rate or SAR, which has units of watts per kilogram (W/kg). The IEEE and many national governments have established safety limits for exposure to various frequencies of electromagnetic energy based on SAR, mainly based on ICNIRP Guidelines, which guard against thermal damage.

There are publications which support the existence of complex biological and neurological effects of weaker non-thermal electromagnetic fields , including weak ELF magnetic fields and modulated RF and microwave fields. Fundamental mechanisms of the interaction between biological material and electromagnetic fields at non-thermal levels are not fully understood.

Lighting.

Fluorescent lights.

Fluorescent light bulbs and tubes internally produce ultraviolet light. Normally this is converted to visible light by the phosphor film inside a protective coating. When the film is cracked by mishandling or faulty manufacturing then UV may escape at levels that could cause sunburn or even skin cancer.

LED lights.

High CRI LED lighting.

Blue light, emitting at wavelengths of 400–500 nanometers, suppresses the production of melatonin produced by the pineal gland. The effect is disruption of a human being's biological clock resulting in poor sleeping and rest periods.

EMR effects on the human body by frequency

Warning sign next to a transmitter with high field strengths

While the most acute exposures to harmful levels of electromagnetic radiation are immediately realized as burns, the health effects due to chronic or occupational exposure may not manifest effects for months or years.[citation needed]

Extremely-low frequency

High-power extremely-low-frequency RF with electric field levels in the low kV/m range are known to induce perceivable currents within the human body that create an annoying tingling sensation. These currents will typically flow to ground through a body contact surface such as the feet, or arc to ground where the body is well insulated.

Shortwave

Shortwave (1.6 to 30 MHz) diathermy heating of human tissue only heats tissues that are good electrical conductors, such as blood vessels and muscle. Adipose tissue (fat) receives little heating by induction fields because an electrical current is not actually going through the tissues.

4.3 Road Transport Safety

The basic strategy of a Safe System approach is to ensure that in the event of a crash, the impact energies remain below the threshold likely to produce either death or serious injury. This threshold will vary from crash scenario to crash scenario, depending upon the level of protection offered to the road users involved. For example, the chances of survival for an unprotected pedestrian hit by a vehicle diminish rapidly at speeds greater than 30 km/h, whereas for a properly restrained motor vehicle occupant the critical impact speed is 50 km/h (for side impact crashes) and 70 km/h (for head-on crashes).

As sustainable solutions for all classes of road have not been identified, particularly low-traffic rural and remote roads, a hierarchy of control should be applied, similar to classifications used to improve occupational safety and health. At the highest level is sustainable prevention of serious injury and death crashes, with sustainable requiring all key result areas to be considered. At the second level is real time risk reduction, which involves providing users at severe risk with a specific warning to enable them to take mitigating action. The third level is about reducing the crash risk which involves applying the road design standards and guidelines (such as from AASHTO), improving driver behavior and enforcement.

4.4 Conclusions

A serious workplace injury or death changes lives forever for families, friends, communities, and coworkers too. Human loss and suffering is immeasurable. Occupational injuries and illnesses can provoke major crises for the families in which they occur. In addition to major financial burdens, they can impose substantial time demands on uninjured family members. Today, when many families are operating with very little free time, family resources may be stretched to the breaking point. Every person who leaves for work in the morning should expect to return home at night in good health. Can you imagine the knock on the door to tell you your loved one will never be returning home? Or the phone call to say he's in the hospital and may never walk again? Ensuring that husbands return to their wives, wives to their husbands, parents to their children, and friends to their friends that is the most important reason to create a safe and healthy work environment. But it isn't the only reason.

CONCLUSION

«Magento» is a popular open systems for organizations e-commerce online: more than 150,000 have been built on basis of this platform online stores, third-party developers have built more than 2.5 thousand extensions, the project community has about 400,000 members. The platform has been downloaded more than 3.5 million times through «Magento»-based systems more than \$ 28 billion worth of goods are sold every year. In a few years of its existence, this platform has won several awards "Best of Open Source Software Awards" and "SourceForge Community Choice Awards".

If we have a fairly simple store, the number of products in which isn't exceeds 100 - 200, we can use popular platforms for Internet WordPress, Joomla or other websites that contain special applications for Internet store and easy to operate.

However, if we need to place in the store quite a significant amount goods (200-500 and more), provide easy navigation by category and individual goods, as well as product filters, we can't do without specialized platforms for online stores - «Magento». First version of «Magento» was released in 2008 year. Today, «Magento» is the most functional and efficient platform for development of online stores, which has gained popularity in many countries around the world.

One of the disadvantages of «Magento» is complexity of development. Technology designed for large projects designed for large numbers of users. Building by using Zend Framework, «Magento» has a flexible architecture, modular structure and well-thought-out mechanisms for modifying the existing functionality.

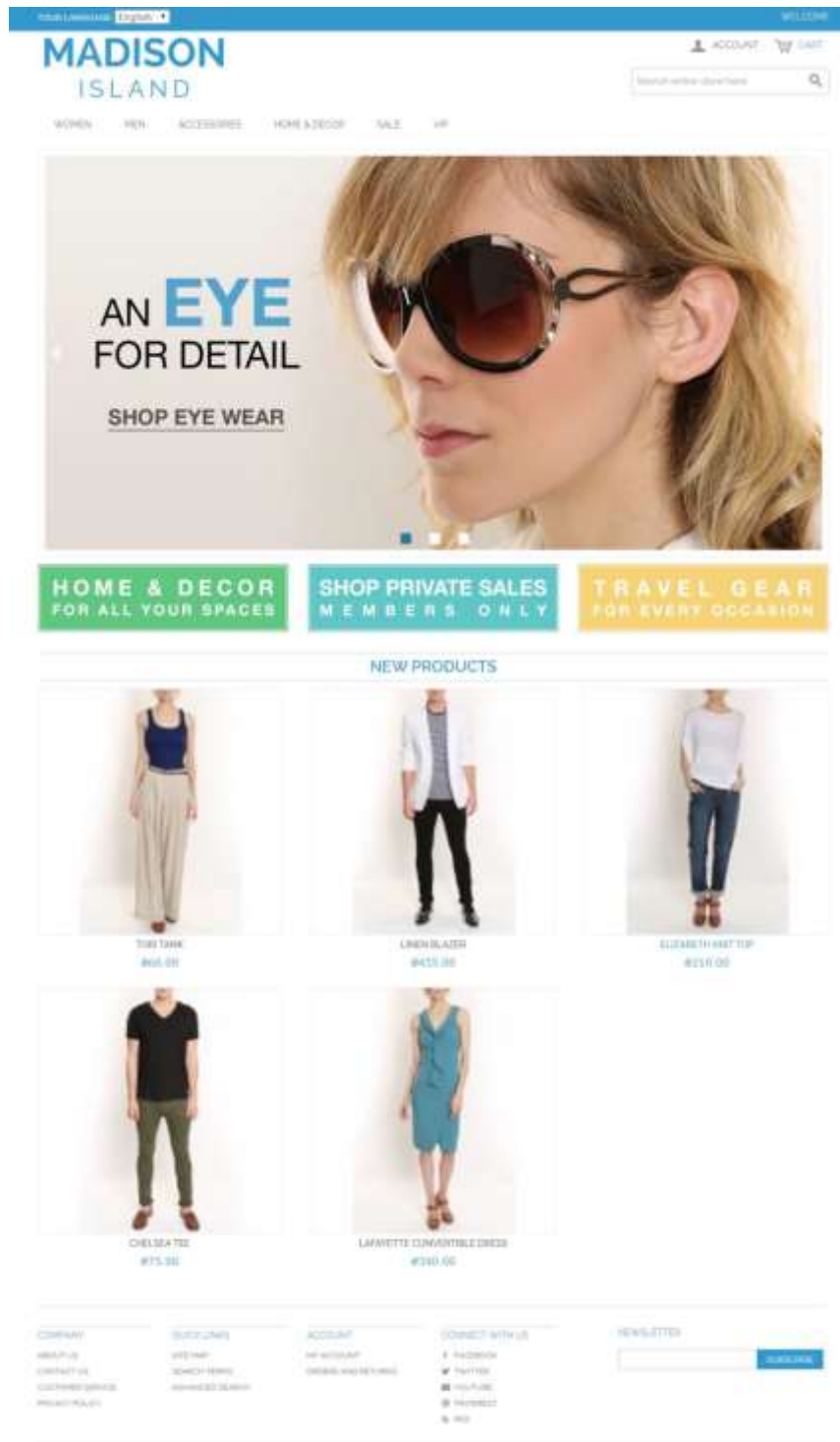
An important result of the thesis is the construction of and setting up the technology of creating an online store with capabilities scaling the range of goods and the number of buyers, and this technology is not too difficult to learn and use by ordinary users.

REFERENCES

1. <http://www.opennet.ru/opennews/art.shtml?num=30814>
2. <http://www.cnet.com/news/and-the-winners-of-the-2018-sourceforgecommunity-choice-awards-are/>
3. <http://dou.ua/lenta/interviews/interview-»Magento»-2019/>
4. <http://forbes.ua/company/1476>
5. <http://builtwith.com/allo.ua>
6. <http://forbes.ua/magazine/forbes/1336514-v-nachale-bolshogo-vzryva-top-15-internet-kompanij>
7. <http://builtwith.com/mo.ua>
8. <http://wikipedia.org>
9. <http://www.»Magento»commerce.com/wiki/>
10. «Magento» Inc. Designer’s Guide to “«Magento»” — 2018.
11. Milan Stojanov. «Magento» Fundamentals. — 2014.
12. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Receptions object-oriented design. Design patterns. SPb: Peter, 2012. –368 p.
13. Linux-Kernel Archive: Re: Alan Cox quote?m (<http://www.uwsg.indiana.edu/hypermail/linux/kernel/0106.2/0405.html>)
14. Deyt KJ. Introduction to database systems - 8th ed. - M. Williams, 2006. 1328 p.
15. Raichev I.E. Principles of design of open distributed systems / I.E. Raichev, O.G. Kharchenko, V.V. Zamkovy // Textbook way. –K: Published by Nat.aviation. NAU-Druk University, 2010 - 240 p.
16. Information systems and data structures / O.B. Ivankevich, H.M. Kremenetsky, VI Mazur // Textbook. –K: NAU, 2006. –232p.

ANNEXES

Screenshots of the online store website



MADISON ISLAND

ACCOUNT CART

Search entire store here.

WOMEN MEN ACCESSORIES HOME & DECOR SALE VIP

HOME / HOME & DECOR

HOME & DECOR

HOME CHIC HOME
...
luxury isn't only for hotel living

BOOKS & MUSIC

BED & BATH

ELECTRONICS

DECORATIVE ACCENTS

COMPANY
[ABOUT US](#)
[CONTACT US](#)
[CUSTOMER SERVICE](#)
[PRIVACY POLICY](#)

QUICK LINKS
[SITE MAP](#)
[SEARCH TERMS](#)
[ADVANCED SEARCH](#)

ACCOUNT
[MY ACCOUNT](#)
[ORDERS AND RETURNS](#)

CONNECT WITH US
[FACEBOOK](#)
[TWITTER](#)
[YOUTUBE](#)
[PINTEREST](#)
[RSS](#)

NEWSLETTER
 [SUBSCRIBE](#)

MADISON ISLAND

ACCOUNT CART

Search entire store here

WOMEN MEN ACCESSORIES HOME & DECOR SALE VIP

HOME / HOME & DECOR / ELECTRONICS

SHOP BY

PRICE

- #0.00 - #99.99 (4)
- #100.00 - #199.99 (5)
- #400.00 - #499.99 (2)
- #700.00 and above (1)

COLOR

- Black (1)
- Charcoal (2)

ELECTRONIC TYPE













- Accessories (4)
- Camera (3)
- Media Players (2)
- Speakers + Earphones (3)

POPULAR TAGS

grey blue red white
View All Tags

ELECTRONICS

SORT BY: Position VIEW AS 12 Item(s) SHOW 12

 <p>MADISON LX2200 #400.00 Click for price ★★★★</p> <p>ADD TO CART Add to Wishlist Add to Compare</p>	 <p>MADISON RX3400 #615.00 Click for price</p> <p>ADD TO CART Add to Wishlist Add to Compare</p>	 <p>MP3 PLAYER WITH AUDIO From #185.00 To #275.00 ★★★★</p> <p>VIEW DETAILS Add to Wishlist Add to Compare</p>	 <p>MADISON 8GB DIGITAL MEDIA PLAYER #150.00</p> <p>ADD TO CART Add to Wishlist Add to Compare</p>
 <p>MADISON OVEREAR HEADPHONES #125.00 ★★★★</p> <p>ADD TO CART Add to Wishlist Add to Compare</p>	 <p>MADISON EARBUDS #35.00</p> <p>ADD TO CART Add to Wishlist Add to Compare</p>	 <p>LARGE CAMERA BAG #120.00</p> <p>ADD TO CART Add to Wishlist Add to Compare</p>	 <p>8GB MEMORY CARD #30.00</p> <p>ADD TO CART Add to Wishlist Add to Compare</p>
 <p>8GB MEMORY CARD #20.00</p> <p>ADD TO CART Add to Wishlist Add to Compare</p>	 <p>CAMERA TRAVEL SET From #445.00 To #965.00</p> <p>VIEW DETAILS Add to Wishlist Add to Compare</p>	 <p>3-YEAR WARRANTY #75.00</p> <p>ADD TO CART Add to Wishlist Add to Compare</p>	 <p>5-YEAR WARRANTY #100.00</p> <p>ADD TO CART Add to Wishlist Add to Compare</p>

SORT BY: Position VIEW AS 12 Item(s) SHOW 12

RECENTLY VIEWED PRODUCTS



COMPARE PRODUCTS
You have no items to compare.

COMMUNITY POLL

What is your favorite color

- Green
- Red
- Black
- Magenta

VOTE

COMPANY

- ABOUT US
- CONTACT US
- CUSTOMER SERVICE
- PRIVACY POLICY

QUICK LINKS

- SITE MAP
- SEARCH TERMS
- ADVANCED SEARCH

ACCOUNT

- MY ACCOUNT
- ORDERS AND RETURNS

CONNECT WITH US

- FACEBOOK
- TWITTER
- YOUTUBE
- PINTEREST
- RSS

NEWSLETTER

SUBSCRIBE



MP3 PLAYER WITH AUDIO

From \$185.00

To \$275.00

Price as configured **\$150.00**

★★★★
3 Reviews (0) | Add Your Review

IN STOCK

Pick up your Media Player and Audio Output together.

Media Player * * Required Fields
 Madison 3GB Digital Media Player - 18 (10) (0)

Qty:

Audio Output *
 (Choose a selection...)

Qty:

Qty: **ADD TO CART**

[Add to Wishlist](#) [Add to Compare](#) [Facebook](#) [Twitter](#)

MORE VIEWS



REVIEWS

Customer Reviews (3) (0)

LEEDY P. WOULD LOVE TO HAVE 3 MORE COPIES

I wasn't able to choose the color but other than that a great experience and very nice

PRICE	★★★★
VALUE	★★★★
QUALITY	★★★★

Review by LEEDY P. WOULD LOVE TO HAVE 3 MORE COPIES

GOOD BATTERY LIFE

Battery life is good

PRICE	★★★★
VALUE	★★★★
QUALITY	★★★★

Review by LEEDY P. WOULD LOVE TO HAVE 3 MORE COPIES

THANK YOU

I just love this little device and it holds plenty of music. I use it with my wife and share with friends. Thank you for such wonderful pricing.

PRICE	★★★★
VALUE	★★★★
QUALITY	★★★★

Review by LEEDY P. WOULD LOVE TO HAVE 3 MORE COPIES

PRODUCT TAGS

Add Your Tag

ADD TAG

Use spaces to separate tags. Use single quotes (') for phrases.

COMPANY
ABOUT US
CONTACT US
CUSTOMER SERVICE
PRIVACY POLICY

QUICK LINKS
SITE MAP
SEARCH TERMS
SEARCHED HISTORY

ACCOUNT
MY ACCOUNT
ORDERS AND RETURNS

CONNECT WITH US
FACEBOOK
TWITTER
YOUTUBE
INSTAGRAM
RSS

NEWSLETTER
 SUBSCRIBE

SHOPPING CART

PROCEED TO CHECKOUT

MP3 Player with Audio was added to your shopping cart.

PRODUCT	PRICE	QTY	SUBTOTAL
 MADISON RX3400 <small>S&C: Island</small>	#715.00	1	#715.00
 MP3 PLAYER WITH AUDIO <small>S&C: Island-Island-Island</small> Media Player: 1 x Madison 8GB Digital Media Player #150.00 Audio Output: 1 x Madison Overear Headphones #125.00	#275.00	1	#275.00

[EMPTY CART](#) [UPDATE SHOPPING CART](#) [CONTINUE SHOPPING](#)

DISCOUNT CODES [APPLY](#)

ESTIMATE SHIPPING AND TAX

COUNTRY * STATE/PROVINCE *

United States Please select region, 1

CITY ZIP *

ADDRESS_HOME_CIT ADDRESS

[ESTIMATE](#)

SUBTOTAL	#900.00
TAX	#91.68
GRAND TOTAL	#1,071.68

[PROCEED TO CHECKOUT](#)

COMPANY

- [ABOUT US](#)
- [CONTACT US](#)
- [CUSTOMER SERVICE](#)
- [PRIVACY POLICY](#)

QUICK LINKS

- [SITE MAP](#)
- [SEARCH TERMS](#)
- [ADVANCED SEARCH](#)

ACCOUNT

- [MY ACCOUNT](#)
- [ORDERS AND RETURNS](#)

CONNECT WITH US

- [FACEBOOK](#)
- [TWITTER](#)
- [YOUTUBE](#)
- [INTEREST](#)
- [RSS](#)

NEWSLETTER

[SUBSCRIBE](#)

CHECKOUT

- 1 CHECKOUT METHOD Edit
- 2 BILLING INFORMATION

YOUR CHECKOUT PROGRESS

- BILLING ADDRESS
- SHIPPING ADDRESS
- SHIPPING METHOD
- PAYMENT METHOD

* Required Fields

First Name *

Last Name *

Company

Email Address *

Address *

Street Address 2

City *

State/Province *

Please select region, state or province

Zip/Postal Code *

Country *

United States

Telephone *

Fax

- Ship to this address
- Ship to different address

CONTINUE

- 3 SHIPPING INFORMATION
- 4 SHIPPING METHOD
- 5 PAYMENT INFORMATION
- 6 ORDER REVIEW

COMPANY

- ABOUT US
- CONTACT US
- CUSTOMER SERVICE
- PRIVACY POLICY

QUICK LINKS

- SITE MAP
- SEARCH TERMS
- ADVANCED SEARCH

ACCOUNT

- MY ACCOUNT
- ORDERS AND RETURNS

CONNECT WITH US

- FACEBOOK
- TWITTER
- YOUTUBE
- INTEREST
- RSS

NEWSLETTER