МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОПОЛОГІЙ ПРОЕКТУВАННЯ

Обов’язкове домашне завдання

з дисципліни

«Проектування веб-орієнтованих систем»

Виконав студент групи ІТ.мз-62с

Тиченко Я.М.

Варіант 1

Перевірила Парфененко Ю.В.

Суми 2017

# Зміст

[Зміст 2](#_Toc502228592)

[Вступ 3](#_Toc502228593)

[1 Теоретична частина 4](#_Toc502228594)

[1.1 Назва та коротка характеристика шаблону factory. 4](#_Toc502228595)

[1.2 Використання у вебі. 4](#_Toc502228596)

[1.3 UML діаграма класів, що ілюструє суть даного шаблону 4](#_Toc502228597)

[1.4 Опис реалізації шаблону 5](#_Toc502228598)

[1.5 Переваги та недоліки 6](#_Toc502228599)

[1.6 Назва та коротка характеристика шаблону front-controller. 6](#_Toc502228600)

[1.7 Використання у веб. 6](#_Toc502228601)

[1.8 UML діаграма класів, що ілюструє суть даного шаблону 7](#_Toc502228602)

[1.9 Опис реалізації шаблону 8](#_Toc502228603)

[1.10 Переваги та недоліки 10](#_Toc502228604)

[2 Практичне завдання 11](#_Toc502228605)

[2.1 Постановка задачі 11](#_Toc502228606)

[2.2 UML діаграми патернів 11](#_Toc502228607)

[2.3 Тексти файлу проекту 12](#_Toc502228608)

[2.4 Контрольні приклади 17](#_Toc502228609)

[Висновок 18](#_Toc502228610)

[Література 19](#_Toc502228611)

# Вступ

З кожним роком зростає складність програмних комплексів, а також підвищуються вимоги до їх надійності та швидкої. Ці чинники ускладнюють процес написання програмного забезпечення й тому розробники намагаються використовувати певні шаблони для вирішення типових задач.

Такі шаблони називаються шаблонами проектування (англ. design pattern). Вони призначені для вирішення певної типової задачі та мають свої переваги й недоліки. Їх використання може значно полегшити процес написання програмних продуктів.

Метою даної роботи є ознайомлення з деякими шаблонами проектування та їх використання у web-додатках. Дана робота містить дві частини: пошук теоретичних даних про шаблон проектування; практична реалізація задачі, яка вирішується за допомогою деяких шаблонів проектування.

# 1 Теоретична частина

## 1.1 Назва та коротка характеристика шаблону factory.

Фабричний метод (англ. Factory Method також відомий як Virtual Constructor) - породжує шаблон проектування, що надає інтерфейс підкласс для створення екземплярів деякого класу. В момент створення нащадки можуть визначити, який клас створювати. Іншими словами, даний шаблон делегує створення об'єктів нащадків батьківського класу. Це дозволяє використовувати в коді програм не специфічні класи, а маніпулювати абстрактними об'єктами на більш високому рівні.

## 1.2 Використання у вебі.

Фабричний метод у web широко і часто застосовується, а його використовують, коли створення об'єкта передбачає якусь логіку, а не просто кілька присвоєнь, тоді є сенс делегувати завдання відокремленої фабрики, а не повторювати всюди один й той самий код.[2]

## 1.3 UML діаграма класів, що ілюструє суть даного шаблону

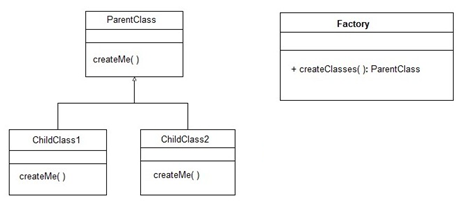


Рисунок 1.1 – UML діаграма прикладу фабричного методу

## 1.4 Опис реалізації шаблону

abstract class Animal

{

// фабричний метод, який на основі типу повертає об'єкт

public static function initial($animal)

{

return new $animal();

}

abstract public function voice();

}

class Lion extends Animal

{

public function voice()

{

echo 'Rrrrrrrr i\'m the lion <br />' . PHP\_EOL;

}

}

class Cat extends Animal

{

public function voice()

{

echo 'Meow, meow i\'m the kitty <br />' . PHP\_EOL;

}

}

$animal1 = Animal::initial('Lion');

$animal2 = Animal::initial('Cat');

$animal1->voice();

$animal2->voice(); [6]

## 1.5 Переваги та недоліки

Перевагами є те, що використання паттерну позбавляє клас від прив'язки до конкретних класів проектів, виділяє код виробництва продуктів в одне місце, спрощуючи підтримку коду, спрощує додавання нових продуктів в програму, реалізує принцип відкритості / закритості.[3]

Недоліком є те, що шаблон може призвести до створення великих паралельних ієрархій класів, так як для кожного класу продукту треба створити свій підклас творця.[4]

## Назва та коротка характеристика шаблону front-controller.

Front-controller (фронт-контролер, єдина точка входу) — шаблон проектування, який є спеціалізованою варіацією шаблону проектування Посередник. Задачею фронт-контролера є надання єдиної точки входу для обробки усіх запитів та виклик відповідної поведінки в залежності від запиту.[5]

## 1.7 Використання у веб.

Фронт-контролер найчастіше використовується у веб-застосуваннях, де є багато подібних речей, які потрібно виконати при обробці запиту. Це може бути безпека, інтернаціоналізація, забезпечення певного вигляду для певних користувачів. Якщо обробка вхідних запитів розподілена між кількома контролерами це може призвести до дублювання поведінки. Крім того виникають складності зі зміною поведінки під час виконання. Фронт-контролер об'єднує обробку запитів шляхом їх направлення через єдиний об'єкт-обробник. Цей об'єкт реалізовує загальну поведінку, яка може бути змінена під час виконання за допомогою декораторів. Після цього Front controller створює потрібні об'єкти відповідно до запиту та викликає методи для реалізації конкретної задачі.

Фронт-контролер може бути реалізований у вигляді Java-об'єкта, або, як скрипт PHP, ASP, JSP або CFML, що викликається на кожен запит веб-сесії. Цей скрипт, наприклад index.php, буде обробляти всі запити, що є спільними для веб-застосування або фреймворку, наприклад обробка сесій, кешування і фільтрація вхідних даних.

Альтернативою фронт-контролеру можуть бути окремі скрипти наприклад login.php і order.php для обробки певного типу запиту. Кожному скрипту доведеться дублювати код або об'єкти, які є спільними для всіх запитів, але кожен скрипт має порівняно більшу гнучкість для обробки конкретного запиту.[6]

## 1.8 UML діаграма класів, що ілюструє суть даного шаблону

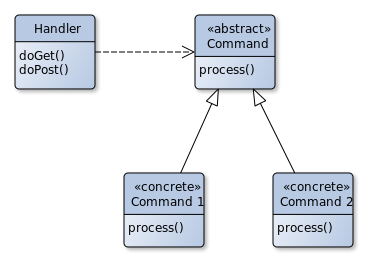


Рисунок 1.2 – UML діаграма прикладу фронт-контролеру

## 1.9 Опис реалізації шаблону

class FrontController {

private $controller;

private $view;

public function \_\_construct(Router $router, $routeName, $action = null) {

//Fetch a route based on a name, e.g. "search" or "list" or "edit"

$route = $router->getRoute($routeName);

//Fetch the names of each component from the router

$modelName = $route->model;

$controllerName = $route->controller;

$viewName = $route->view;

//Instantiate each component

$model = new $modelName;

$this->controller = new $controllerName($model);

$this->view = new $viewName($routeName, $model);

//Run the controller action

if (!empty($action)) $this->controller->{$action}();

}

public function output() {

// спосіб виведення даних з вигляду

// Це дає змогу отримати певний послідовний код генерації макета, наприклад заголовок / нижній колонтитул сторінки

$header = '<h1>Hello world example</h1>';

return $header . '<div>' . $this->view->output() . '</div>';

}

}

## 1.10 Переваги та недоліки

Перевагою фронт контролера є те , що контролер запитів об'єднує всі дії по обробці запитів в одному місці, що дає йому додаткові можливості, завдяки яким можна реалізувати більш важкі завдання, ніж звичайно вирішуються контролером сторінок..

Недоліком є випадки, коли обробка вхідних запитів розподілена між кількома контролерами це може призвести до дублювання поведінки. Крім того виникають складності зі зміною поведінки під час виконання.[7]

# 2 Практичне завдання

## 2.1 Постановка задачі

Практичним завданням було вирішено реалізувати компонент, котрий керує і створює показ повідомлень для користувача при зчитувань даних з деякої форми збору даних. Так як форма повинна відображати дані на 2 мовах, то було вирішено використати фабричний метод. Також при показі повідомлень, в деяких випадках повідомлення показуються «серіями» по 2 штуки. Для реалізацію показу декількох повідомлень за раз було обрано паттерн фасад. Як правило, фасад використовують для спрощення інтерфейсу у важких системах, але тут використано його простіший приклад.

## 2.2 UML діаграми патернів

UML для Фабричного методу було розібрано у теоретичній частині. Для фасаду UML зображено на рисунку 2.1

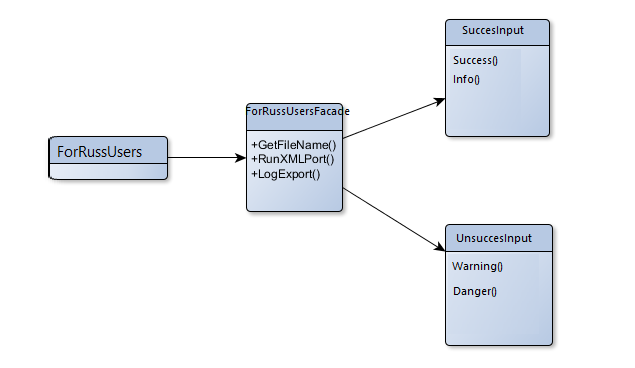


Рисунок 2.1 – UML діаграма прикладу фасаду

## 2.3 Тексти файлу проекту

Текст основного файлу index.php :

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>AlertBuilder</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/css/bootstrap.min.css" integrity="sha384-rwoIResjU2yc3z8GV/NPeZWAv56rSmLldC3R/AZzGRnGxQQKnKkoFVhFQhNUwEyJ" crossorigin="anonymous">

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/js/bootstrap.min.js" integrity="sha384-vBWWzlZJ8ea9aCX4pEW3rVHjgjt7zpkNpZk+02D9phzyeVkE+jo0ieGizqPLForn" crossorigin="anonymous"></script>

</head>

<body>

<?php

abstract class Builder

{

// Шаблонный метод

final public function buildAlert()

{

$this->success();

$this->warning();

$this->danger();

$this->info();

}

abstract public function success();

abstract public function warning();

abstract public function danger();

abstract public function info();

}

class ForRussUsers extends Builder

{

public function success()

{

echo '<div class="alert alert-success" role="alert">

<strong>Все отлично!</strong> Ваши данные успешно сохранены.

</div>';

}

public function warning()

{

echo '<div class="alert alert-warning" role="alert">

<strong>Внимание!</strong> Повторите попытку, небольшие неполадки.

</div>';

}

public function danger()

{

echo '<div class="alert alert-danger" role="alert">

<strong>Беда!</strong> Что-то пошло не так, перешли данные ещё разок.

</div>';

}

public function info()

{

echo '<div class="alert alert-info" role="alert">

<strong>Данные приняты!</strong> Ваши данные успешно приняты для обработки.

</div>';

}

}

class ForUkrUsers extends Builder

{

public function success()

{

echo '<div class="alert alert-success" role="alert">

<strong>Усе добре!</strong> Ваші данні успішно передані.

</div>';

}

public function warning()

{

echo '<div class="alert alert-warning" role="alert">

<strong>Увага!</strong> Виникли неполадки, треба передати дані ще раз.

</div>';

}

public function danger()

{

echo '<div class="alert alert-danger" role="alert">

<strong>Ой біда!</strong> Сталась помилка і ваші дані не буде передані.

</div>';

}

public function info()

{

echo '<div class="alert alert-info" role="alert">

<strong>Дані прийняті!</strong> Відправлені вами дані прийнято.

</div>';

}

}

$forRussUsers = new ForRussUsers();

$forRussUsers->buildAlert();

$forUkrUsers = new ForUkrUsers();

$forUkrUsers->buildAlert();

class ForRussUsersFacade

{

protected $ForRussUsers;

public function \_\_construct(ForRussUsers $ForRussUsers)

{

$this->ForRussUsers = $ForRussUsers;

}

public function unsuccessInput()

{

$this->ForRussUsers->warning();

$this->ForRussUsers->danger();

}

public function successInput()

{

$this->ForRussUsers->success();

$this->ForRussUsers->info();

}

}

$ForRussUsers = new ForRussUsersFacade(new ForRussUsers());

$ForRussUsers->unsuccessInput();

$ForRussUsers->successInput();

?>

</body>

</html>

## 2.4 Контрольні приклади

## 

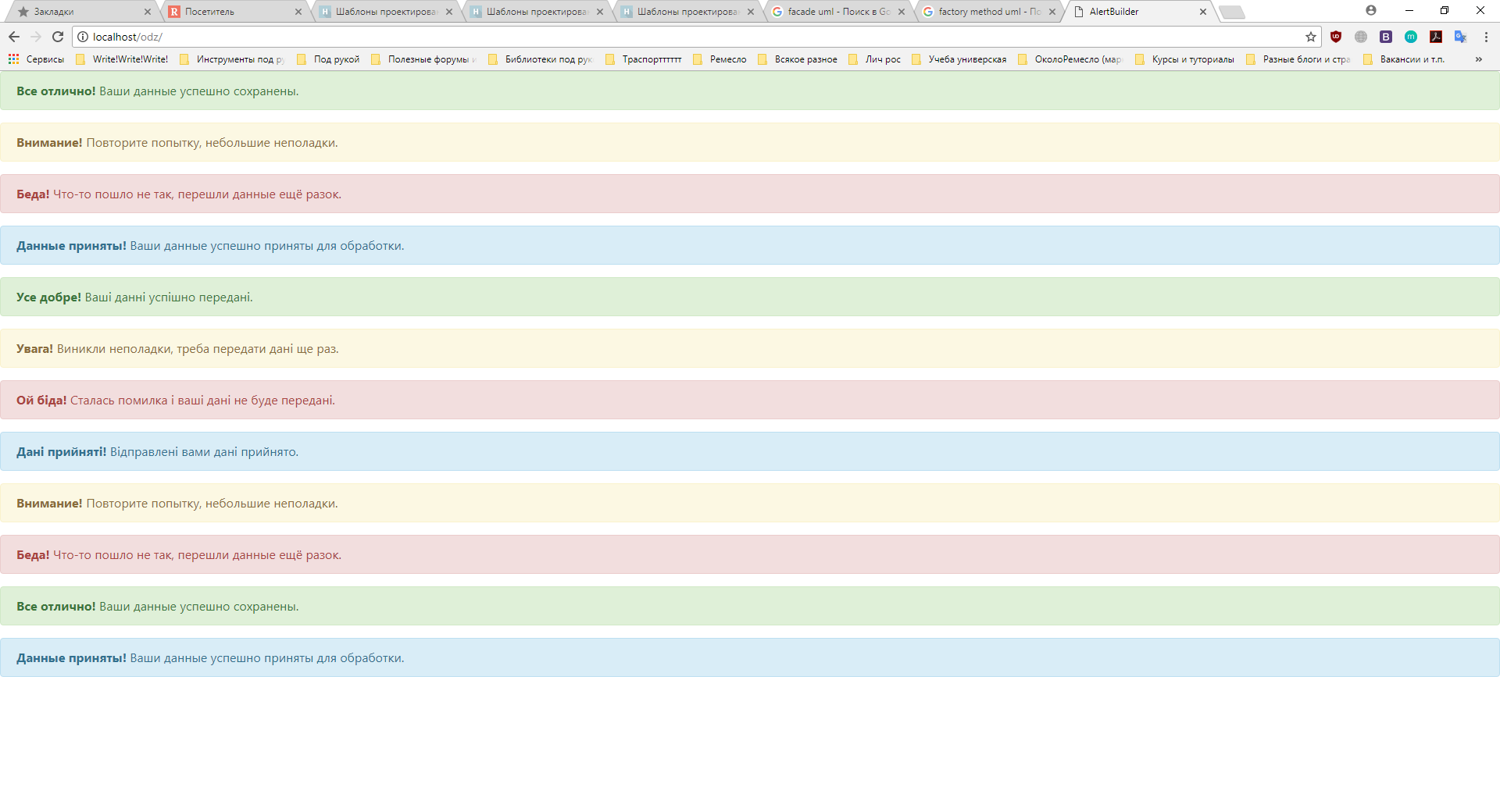


Рисунок 2.2 – Повідомлення для російськомовних користувачів

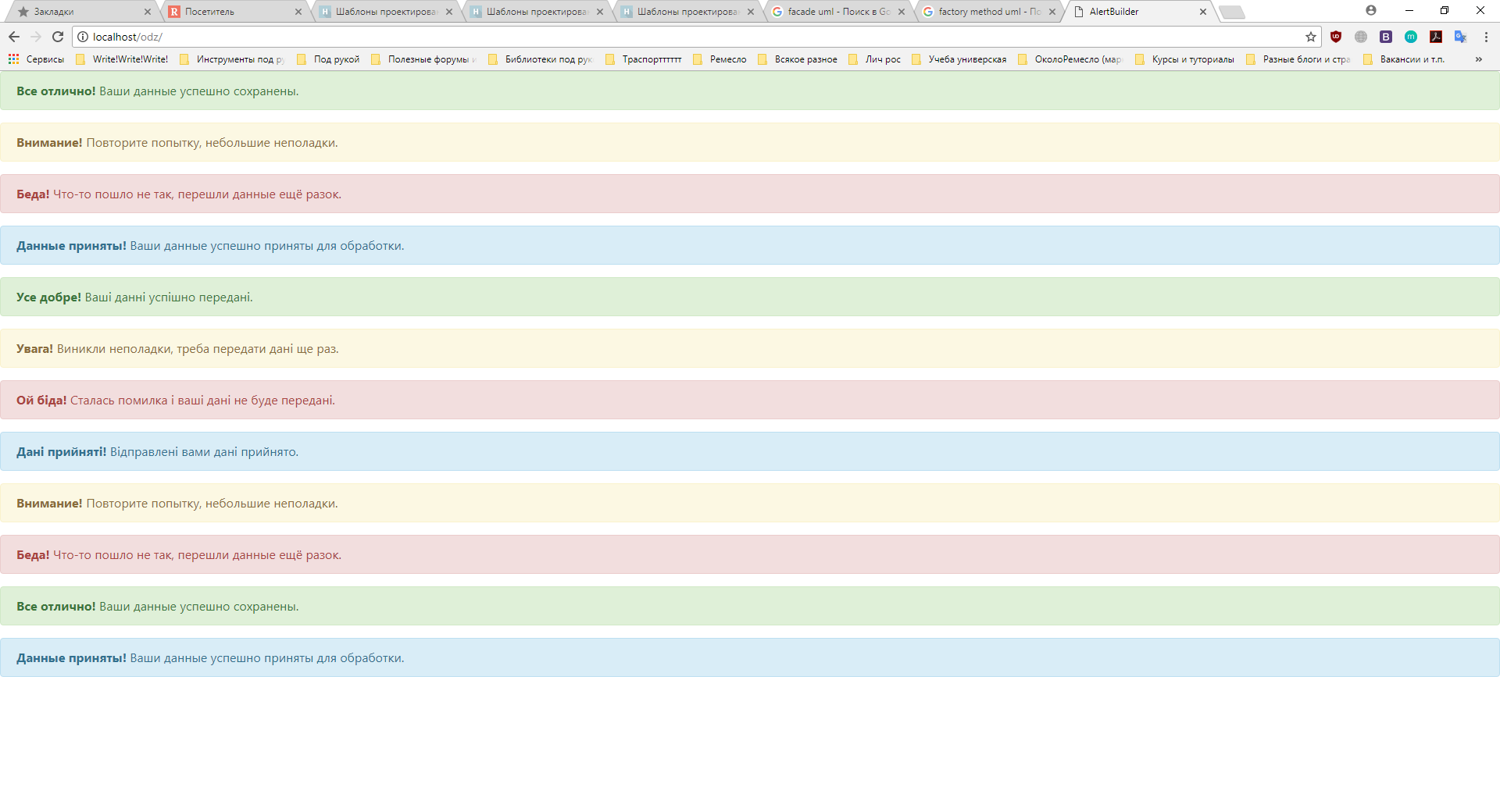


Рисунок 2.3 – Повідомлення для україномовних користувачів

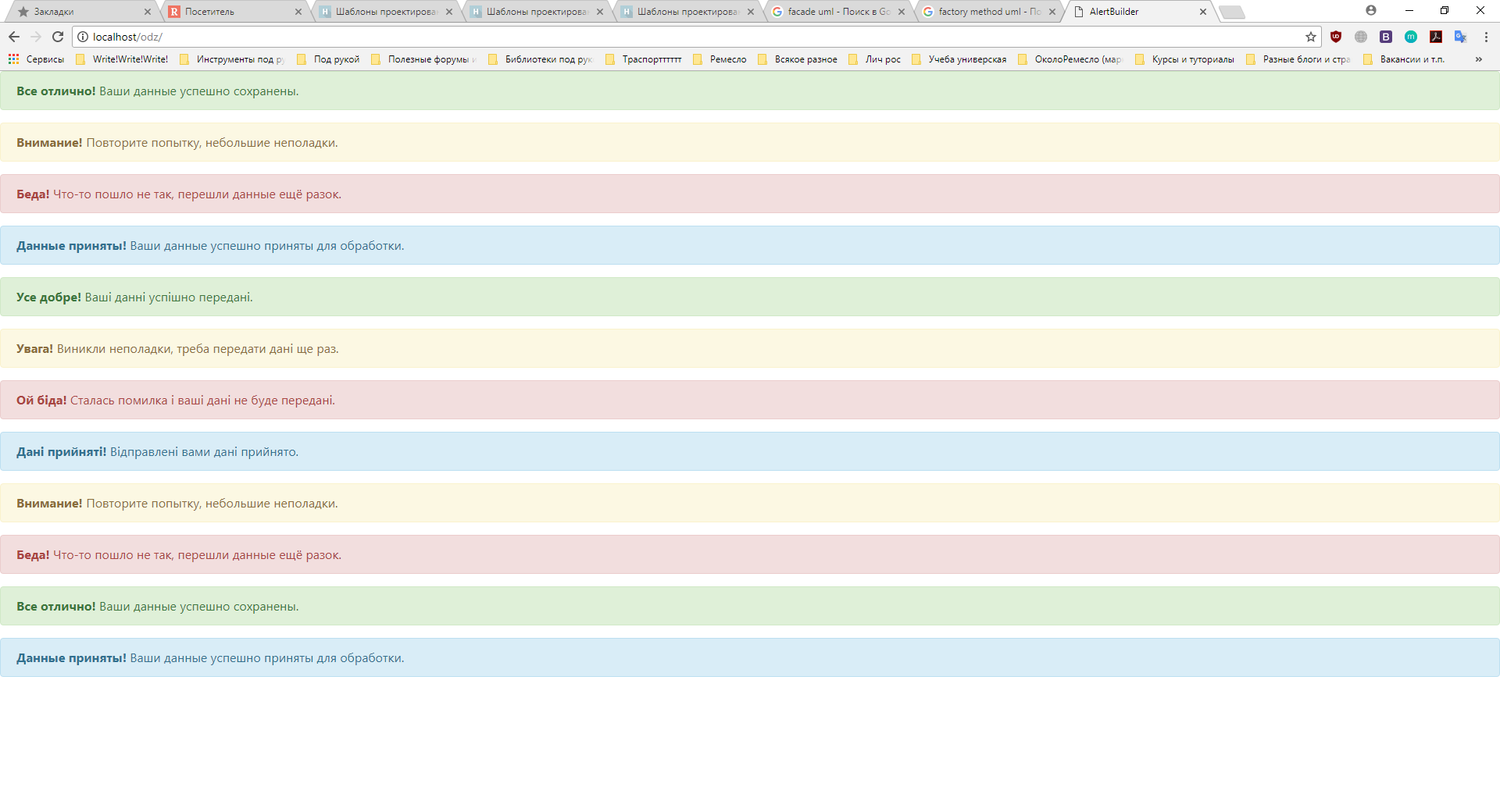


Рисунок 2.4 – Повідомлення при невдалій передачі

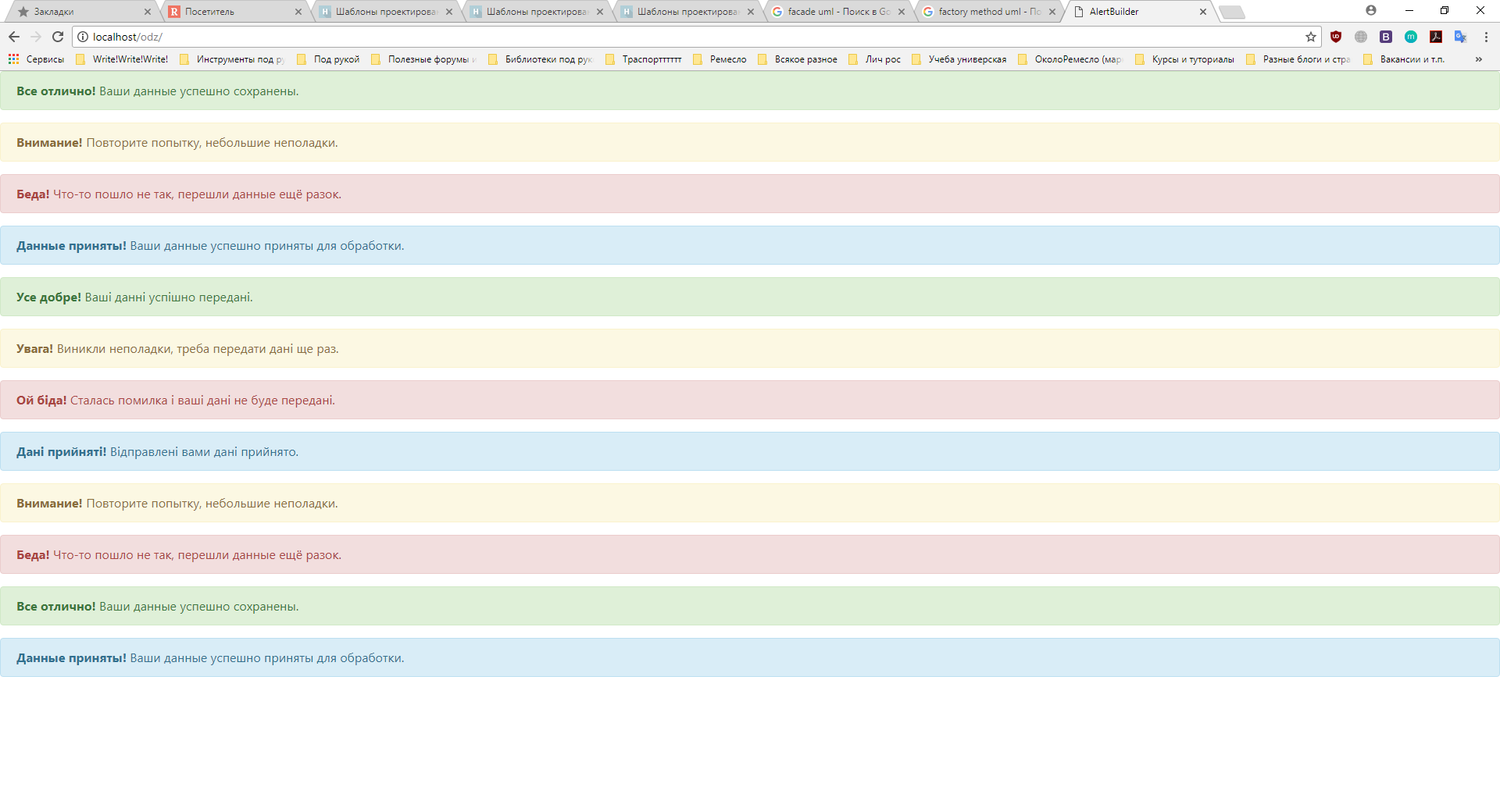


Рисунок 2.5 – Повідомлення при вдалій передачі

# Висновок

В результаті виконання роботи розглянуто шаблони проектування фронт контролер та фабричний метод. Були вивчені теоретичні аспекти шаблонів проектування, такі як: класифікація шаблону, призначення, мотивація використання, приклади ситуацій для застосування, його структура, переваги та недоліки.

Також, було виконане практичне завдання з використанням шаблонів фасад та фабричний метод, їх реалізація та опис. В ході практичної роботи були набуті навичка застосування цих шаблонів на практиці.

# Література

1. Шаблони проектування програмного забезпечення [Електронний ресурс] – Режим доступу до ресурсу:<http://www.kytok.org.ua/?page_id=905>.
2. Приклади об'єктно-орієнтованого проектування / Е. Гамма, Р. Хелм, Р. Джонсон, Дж. Вліссідес. – Санкт-Петербург: Питер, 2001. – 344 с.
3. Паттерны проектирования.Структурные паттерны [Електронний ресурс] – Режим доступу до ресурсу: <http://faq.linemedia.ru/patternyi-pro/>
4. Learning JavaScript Design Patterns [Електронний ресурс] – Режим доступу до ресурсу: [https://addyosmani.com/resources/essentialjsdesignpatterns/book/#observerpatternjavascript](https://uk.wikipedia.org/wiki/Front_controller)
5. Адаптер (шаблон проектирования) [Електронний ресурс] – Режим доступу до ресурсу: <http://dic.academic.ru/dic.nsf/ruwiki/116914>
6. Адаптер [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Adapter_pattern>
7. Шаблони проектування [Електронний ресурс] – Режим доступу до ресурсу: <https://refactoring.guru/ru/design-patterns>
8. Front Controller - шаблон проектування [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Front\_controller