WWDS

Object Design

<1.0>

29.12.2017

Muhammed Ali Altınel
Mercan Gültekin
Serdar Ünlüsoy
Erdal Yurtseven

Prepared for
SE301 Software Engineering

Table of Contents

OBJECT DESIGN DOCUMENT

1. Introduction

This Object Design Document (ODD) defines the object-level design of WWDS to be developed. The goal of this project is to develop for Users who would like to donate to one or more projects and to start a project or more projects.

2. Object Design Trade-offs

- **Reliability**

We tested all the steps of the software, we developed it with different inputs and tried to improve the experience of the user by fixing all the mistakes. We tested the system after every new feature we added. When we encountered any problem, we figured out where we made a mistake.

- **Expandability**

While designing our system, we paid attention to extensibility. Thanks to its modular structure, our system is ready for new technologies that can be added later.

- **Programmability**

In our project, we tried to create a simple, functional product using the ASP.NET Framework. The reason we use the Framework is that simple operations such as logging in and signing up are readily available and accordingly we can spend more time on other important parts of the software.

- **Availability**

Availability is the ratio of time a system or component is functional to the total time it is required or expected to function. The system we developed must be accessible 24/7 and we designed it this way.

- **Compatibility**

Another reason why we use the ASP.NET MVC framework in our project is that it helped us about compatibility and it has provided us with the following advantages:

- It made easier to manage the mix by dividing the application into model, view, and controller
- It was better support for test-based development.

<WWDS>

3. Interface Documentation Guidelines

Naming Guideline:

Classes: Class names should be noun, short and summarize what class is about.

Methods: Method names should start with uppercase first letter and each first letter of internal word should be capitalized. Generally, methods should be written as verb:

Variables: Variables should start with lowercase first letter and each first letter of internal word should be written as uppercase letter. Variables should be short but descriptive.

Using the URL Structure:

There should be a relation between a controller class and a view which is a CSHTML file. Developers should develop controller class able to make sure when user visits a page (or view), Controller class should understand User's intention and direct user to the page user indented to visit.

4. Definitions, Acronyms, and Abbreviations

- **Model:** A Model class of MVC Architectural pattern.
- **View :** A visual representation of a model which might
- **Controller :** The controller translates interactions with the view into actions to be performed by the model.
- **DBMS** : Database Management System
- **HTTP :** A protocol for secure communication over a computer network which is widely used on the Internet.
- **ODD :** Object Design Document
- **PACKAGE:** A UML grouping concept denoting that a set of objects or classes are related.

5. References

https://www.gofundme.com/

https://www.lds.org/help/support/finance/online-donations?lang=eng

<WWDS>

6. Packages

A package is a grouping mechanism for organizing elements into groups to increase their readability. In other words, it is a grouping of model elements, such as use cases, or activities, defining scopes of understanding. Packages are used to deal with complexity in the same way a user organizes files and subdirectories into directories.

In our project we have used Model View Controller approach. Since we have developed our project in Visual Studio environment we have used MVC Web Application structure of the environment. In this structure when user makes an entry using a view, information is sent to relevant controller. Controller processes this information and gathers data from relevant models if neccessary. Our models contain information about domain object. Therefore, Model Package is about managing the database. In our View package, html documents are being held. These HTML provide users the UI. Since they aren't classes but only files, HTML files weren't added into our package diagram. Controller package also used to "direct" users to pages. For example, when User visits the website, HTML file of the WWDS project will be shown by the HomeController class by returning the view of the HTML file.

Thus ; we have three main packages which are obviously Model View and Controller.Besides we also have sub package divisons under these three which are as follows;

(*Please note that some class and package names in our project are choosen from Turkish keywords by our developer team in order to prevent some technical conflicts.This situation does not interfere with the any functionality or purpose of the system.These different named classes and packages are also explained in the following structure.)

<WWDS>

<table>
<tr><th colspan="3">Models</th></tr>
<tr><td colspan="2">aspnet_Applications.cs
aspnet_Membership.cs
aspnet_Paths.cs
aspnet_PersonalizationAllUsers.cs
aspnet_PersonalizationPerUser.cs
aspnet_Profile.cs
aspnet_Roles.cs
aspnet_SchemaVersions.cs
aspnet_Users.cs
aspnet_UsersInRoles.cs
aspnet_WebEvent_Events.cs</td><td>These are auto generated classes by .NET Framework in order to provide some services used by our developer team.These classes are not developed by our developer team hence no explanation provided.</td></tr>
<tr><td>Donate.cs</td><td rowspan="7">These classes are also automatically generated by the database design envioronment that our developer team used but designed by our developer team.</td><td>Holds information about donations.Every instance represents a donation.</td></tr>
<tr><td>DonationDB.cs</td><td>Holds information about the database connection to use in our codes.(Fully auto generated)</td></tr>
<tr><td>Kampanya.cs</td><td>Holds information of projects.Every instance represents a project.</td></tr>
<tr><td>Kategori.cs</td><td>Holds the information that which categories are available to start projects.</td></tr>
<tr><td>Kullanici.cs</td><td>Holds user information.Every instance represents a user.</td></tr>
<tr><td>Resim.cs</td><td>Simply a project has a list of images.</td></tr>
<tr><td>Stuff.cs</td><td>Holds the information of a stuff.A project has a list of stuff and a stuff has some properties.(i.e:Name,quantity etc.)</td></tr>
</table>

Views		
Home Package	Index.cshtml	This is the homepage of the WWDS
Kampanya Package	Index.cshtml	This is the page that a user starts a project.
	KampanyaDetay.cshtml	This is the page that displays a projects details
	Yenile.cshtml	This is the page that a project is edited by it's owner.
Kategori Package	Index.cshtml	This page displays the current project categories in the site.
	KategoriDetay.cshtml	This page displays the list of continuing projects in a specific category.
Kullanıcı Package	Index.cshtml	This is the profile page of a user.
Search Package	Index.cshtml	This is the page that search results displayed.
Shared Package	_AdminLayout.cshtml	In this page site admin sees the functions that s/he can use.
	_Layout.cshtml	This is a page that every other page includes just for styling.
Yönetim Package	Bekleyenler.cshtml	This is the page that the site admin views the projects that waiting for approval.
	Bitenler.cshtml	This is the page that site admin displays the projects that are succeeded and no longer available.
	DevamEdenler.cshtml	This page displays the continuing projects on the site.
	EditBekleyenler.cshtml	This is the page that the site admin views the projects that continuing but edited and waiting for an approval for the edits.

<WWDS>

7. Class Interfaces

Controllers		
Yonetim Controller Package	Resetle()	Provides password changing function for users.
	Kayit()	Provides registering service for users and prevents multiple registrations of same email.
Search Controller Package	Arama()	Provides search service among various projects in the system via their name or category.
Kullanici Controller Package	EditMe()	Provides operations to user in order to let him/her edit his/her profile.
Kategori Controller Package	Index()	Provides service in order to display projects listed according to their categories.
Kampanya Controller Package	KampanyaBaslat()	Provides required storage management services when a user is starting a project.
	BagisYap()	Provides necessary operations for initiating a donation and distributes free cargo code.
	Onayla()	Let's a user to approve an incoming donation to his/her project.
	Red()	Let's a user to reject an incoming donation to his/her project when the donation is wrong.
	Yenile()	Let's user to edit his/her continuing projects information.
	KampanyaSil()	Let's user to delete a project when s/he has started a project and waiting for an approval
	GlnBagsOnay()	Let's user to complete his/her initiated donation.
Home Controller Package	Index()	Loads home page and continuing projects.

<WWDS>