

Tourist Assistant Website

Software Architecture

**Project Code: TAW**

**Document Code: TAW\_Software Architecture – <v0.1>**

**Ha Noi, Octorber 3th 2016**

Record of Change

\*A - Added M - Modified D - Deleted

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Effective Date | Changed Items | A\* M, D | Change Description | New Version |
| 2016/10/03 | Create new | A | Create document | 0.1 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Signature Page

**ORIGINATOR:** Chu Dai Cao Cuong Date: 2016/10/03

*Member*

**REVIEWERS:**

**APPROVAL:** Bùi Đình Chiến Date: 2016/10/08

*Supervisor*

Table of Contents

1 Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms and Abbreviations 4

1.4 References 4

1.5 Overview 4

2 Architectural Representation 4

3 Architectural Goals and Constraints 4

4 Use-Case View 4

4.1 Use-Case Realizations 4

5 Logical View 4

5.1 Overview 4

5.2 Architecturally Significant Design Packages 4

6 Process View 4

7 Deployment View 4

8 Implementation View 4

8.1 Overview 4

8.2 Layers 4

9 Data View (optional) 4

10 Size and Performance 4

11 Quality 4

12 Other Considerations 4

# Introduction

## Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions that have been made on the system.

## Scope

The scope of this document is to depict the architecture of the OHRM website created by OHRM capstone project team.

## Definitions, Acronyms, and Abbreviations

|  |  |  |
| --- | --- | --- |
| Acronym | Definition | Note |
| TAW | Tourist Assistant Website |  |
| MVC | model-view-controller |  |
| DB | Database |  |
| N/A | Not available |  |

## References

1. TAW-Report 1 -SoftwareRequirementSpecification-v0.2, Project Code: TAW, FPT University, Hanoi, Vietnam

## Overview

The Software Architecture Document contains the following subsections:

* **Section 1**: Provide an overview of entire Software Architecture Document.
* **Section 2**: Choice of Architecture Design
* **Section 3**: Architectural Representation
* **Section 4**: Architectural Goals and Constraints
* **Section 5**: Use-Case view
* **Section 6**: Logical View
* **Section 7**: Process View
* **Section 8**: Deployment view
* **Section 9**: Quality

# Choice of architecture design

## MVC Model

The purpose of TAW is developing as tourist assistant website. The system of TAW is structured based on MVC combined with layered architecture.

### MVC model overview

The model-view-controller or MVC is software architecture commonly used for creating web applications or software. In other words, it's a structure for web applications to follow in order to ensure efficiency and consistency. Many of the most popular frameworks use the MVC architecture, including ASP.NET, CodeIgniter, Zend, Django, and Ruby on Rails. At the same time, there are many web developers who don't use a coding framework yet still set up their applications to follow the MVC structure.

The Model-View-Controller (MVC) design pattern assigns objects in an application one of  
three roles: model, view, or controller. The pattern defines not only the roles objects play in  
the application, it defines the way objects communicate with each other. Each of the three  
types of objects is separated from the others by abstract boundaries and communicates with  
objects of the other types across those boundaries. The collection of objects of a certain MVC  
type in an application is sometimes referred to as a layer—for example, model layer.

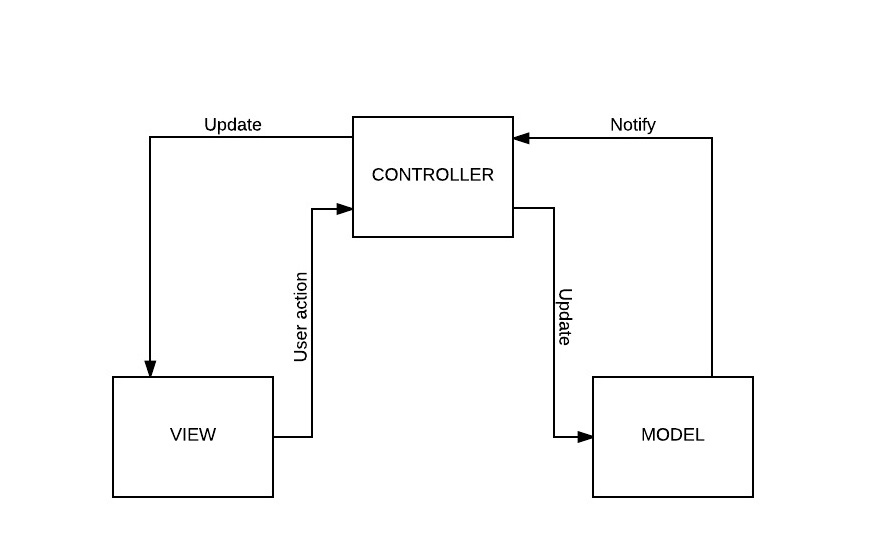


Figure 2‑1: **MVC Model**

In addition to dividing the application into three kinds of components, the MVC design defines the interactions between them \*:

* **Controller:** The controller can be considered the "middle man" of the application. It works with the user, taking in data, and then working with the model to get the appropriate data or calculation, and then working with the view to show the response to the user.
* **Model:** A model is simply a representation of something we need to deal within our application. It is a "model" for something we must represent in code, such as a book, user, bank account, or whatever. The model is responsible for holding the functions and variables that are involved with whatever it's representing. You can think of a model's logic as the core concept to object oriented programming — models are just our "classes". However, don't let this confuse you as controllers are technically structured as classes as well.
* **View:** Finally, after the controller requests information from the model it sends it to a view. A view is just like the application's template system — there might be a view for a certain type of page layout *(profile page)*, a mobile view, or a view for a particular theme/skin. A view will contain all of the markup, CSS, and etc. that you traditionally use with creating a static web page

Source: http://www.onextrapixel.com/2012/03/14/a-detailed-overview-of-the-model-view-controller-mvc-coding-structure/

### Advantages and disadvantages of MVC Model

* Advantages:
* MVC model divides system into components, which can be developed, maintained and upgraded individually without stopping the system.
* Develop tools is useful and easy to use.
* Large of documentary sources.
* Disadvantages:
* For small projects that apply MVC model caused cumbersome, time consuming in development process.
* Time consuming to transits data between components.
* Not suitable for agent-oriented applications such as interactive mobile and robotics applications.
* Multiple pairs of controllers and views based on the same data model make data model change expensive.
* The division between the View and the Controller is not clear in some cases.

### The reason for choosing MVC Model

## Spring framework

### Spring Framework overview

Spring is the most popular application development framework for enterprise Java. Millions of developers around the world use Spring Framework to create high performing, easily testable, reusable code.

Spring framework is an open source Java platform and it was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003.

Spring is lightweight when it comes to size and transparency. The basic version of spring framework is around 2MB.

The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make J2EE development easier to use and promote good programming practice by enabling a POJO-based programming model.

### Advantages and disadvantages of Spring Framework

* Advantages
* Spring WEB framework has a well-designed web MVC framework, which provides a great alternate to web framework.
* Spring can eliminate the creation of the singleton and factory classes.
* Spring framework is both complete and modular, because spring framework has a layered architecture.
* Spring framework has taken the best practice that have been proven over the years in several applications and formalized as design patterns.
* Disadvantages
* One of the major criticisms faced by the spring framework is that it is complex! No so clear focus, more than 2400 classes, 49 other tools and tons of the other things make it complicated for the developers..

### The reason for choosing Spring Framework

* Spring WEB framework has a well-designed web MVC framework, which provides a great alternate to web framework.

## Hibernate SQL

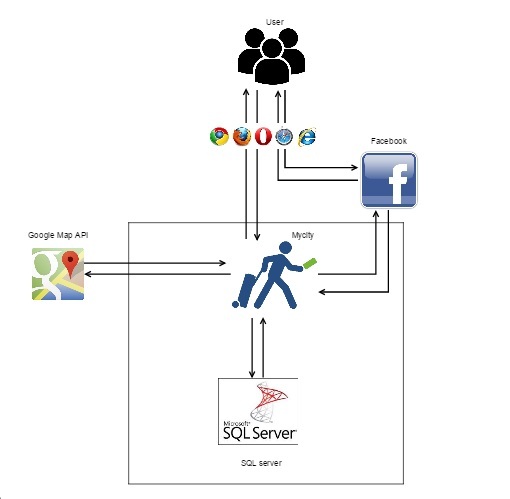
### Hibernate SQL overview

Hibernate is an Object-Relational Mapping(ORM) solution for JAVA and it raised as an open source persistent framework created by Gavin King in 2001. It is a powerful, high performance Object-Relational Persistence and Query service for any Java Application.

Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieve the developer from 95% of common data persistence related programming tasks.

Hibernate sits between traditional Java objects and database server to handle all the work in persisting those objects based on the appropriate O/R mechanisms and patterns.

# Architectural Repersentation



# Architectural goals and constraints

# Use –case view

# Logical view

## Overview

## Architecturally Significant Design Packages

# Process View

# Deployment View

# Quality