IMG_256IMG_256

**软件工程**

**设计说明书**

张良·王鑫晨·蒋睿

# **说明：本文档分为两个部分**

1. 需求分析
2. .开发技术和架构

**文档包含了我们在开发阶段的思考运用到的一些知识和我们对于博客的一些思考。是我们整个项目的一个初步展示，希望老师多多指教。**

# 第一部分 需求分析

## 1.1我们想要做的

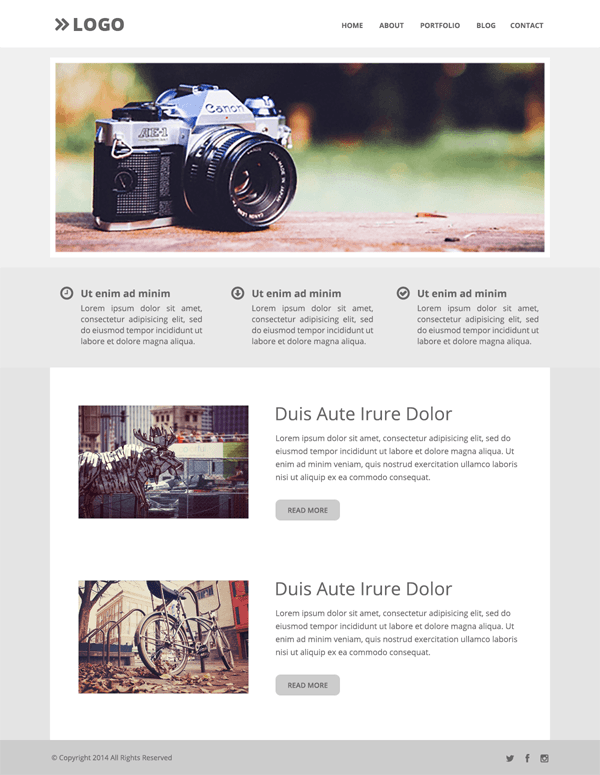
整个系统的角色有三个：访客、博主和管理员。

* 访客具有的权限最少，可以浏览阅读博文并给博文评论。此外，对于访客和博主间的互动，我们还设置了专门的留言板；提供了订阅关注博主、私信交流的功能；此外，点赞与打赏也是我们针对访客博主间互动而设计的功能。
* 对于博主，发表、编辑、删除和浏览博文是我们提供的基本功能。此外，为了便于博主管理自己的博客，我们的博客还满足了博文的分类归档需求。在管理博客方面，对于评论和留言，我们都提供了足够的支持。作为博主，亦是访客，我们提供了对自己信息的维护以及作为访客时的所有功能。
* 管理员是博客系统能正常合法运行的关键一环，我们提供了对注册用户的管理和敏感信息过滤的功能。此外，对于优质的博文，管理员可以进行推荐。
* 系统的响应速度是用户体验关键的一环，我们将加载时间控制在五秒之内，且我们的页面具有良好的兼容性，在PC，手机或是平板上都有着优异的表现。同时，为了不断进步，我们留下了相应的接口供之后的维护或功能的拓展。

**1.1.1我们在界面设计上参考了国内外许多网页作品优秀案例**



Pic 1.1 Lindadong的获奖作品 薄和灰色擦出别样火花



Pic 1.2 灰色的网页给人简洁大气之感

对于灰色的运用也是颇具考量的：灰色虽然在大家的心目中会有忧郁压抑的感觉，但是在设计领域中灰色是非常高级的颜色，因为它跟任何颜色的搭配都是最合适的衬托，是经久不衰、最经看的颜色。灰色常常被我们在设计中用来做辅助色，配合其他色调进行调亮，阴影等效果，利用它来吸收其他色彩的活力，削弱色彩的对立面，而制造出融合的作用。在苹果和微软的官网上，灰色的运用也是颇为惊艳的。容易忽略的，往往蕴含着无穷的价值。



Pic 1.3 Eeharbor在铬黄和灰白的搭配上令人惊艳

在界面图形的选择设计上我们也有自己的想法，从模块状的功能分部到圆形或者扁长的按钮；从网页功能的放置位置到字体大小对用户的浏览体验的影响，我们都颇为留心，力求完美。

**1.1.2 我们在技术上将课内课外知识结合让开发过程更加直观**

在NIIT的之前课程中，HTML 5和ASP.NET MVC的课程给了我们很大的帮助，同时也给了我们很多启发。



MVC 模式可以帮助我们创建使应用程序的不同方面（输入逻辑、业务逻辑和 UI 逻辑）分离的应用程序，同时可在这些元素之间提供松散耦合。 该模式指定每种逻辑在应用程序中应处的位置。 UI 逻辑位于视图中。 输入逻辑位于控制器中。 业务逻辑位于模型中。 在生成应用程序时，通过使用这种分离方式，可以化繁为简，因为它可以使我们侧重于一次实现应用程序的一个方面。 例如，侧重于独立于业务逻辑的视图。

MVC 应用程序的这三个主要组件之间的松散耦合也可促进并行开发。 例如，一个开发人员可以从事视图方面的工作，第二个开发人员可以从事控制器逻辑方面的工作，第三个开发人员可以侧重于模型中的业务逻辑。

**1.1.3 我们在对于博客这类网络应用的理解上更加深刻**

Pic 1.3 Instagram以图片为主独一无二的社交新体验

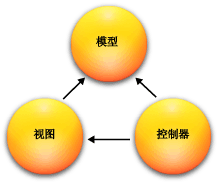
组内成员都有着长时间的社交网络经历，我们是伴随着互联网飞速发展成长的一代，见证了网络交际的一个个里程碑式的变迁。如今，学生之间的线上互动更多的会通过博客，微博这样的形式进行。从国外基于用户订阅关系的社交产品Instagram到中国本土的微博、空间到以兴趣为切入点的豆瓣和知乎，先是用户再是开发者的我们更懂自己。我们明白设计和开发一定从用户的角度出发，“用户需求”才是开发设计的第一准则。

# 第二部分 开发技术和架构

## 2.1 ASP.NET MVC

模型-视图-控制器 (MVC) 体系结构模式将应用程序分成三个主要组件：模型、视图和控制器。 ASP.NET MVC 框架提供用于创建 Web 应用程序的 ASP.NET Web 窗体模式的替代模式。 ASP.NET MVC 框架是一个可测试性非常高的轻型演示框架，（与基于 Web 窗体的应用程序一样）它集成了现有的 ASP.NET 功能，如母版页和基于成员资格的身份验证。 MVC 框架在 System.Web.Mvc 程序集中定义。

MVC 是许多开发人员熟悉的标准设计模式。 一些类型的 Web 应用程序将得益于 MVC 框架。 一些类型将继续使用基于 Web 窗体和回发的传统 ASP.NET 应用程序模式。 其他类型的 Web 应用程序将结合这两种方法；这两种方法彼此互不包含。



3.1 MVC 设计模式

2.2.1 MVC 框架包括以下组件：

* 模型。 模型对象是实现应用程序数据域逻辑的应用程序部件。 通常，模型对象会检索模型状态并将其存储在数据库中。 例如，Product 对象可能会从数据库中检索信息，操作该信息，然后将更新的信息写回到 SQL Server 数据库内的 Products 表中。
* 在小型应用程序中，模型通常是概念上的分离，而不是实际分离。 例如，如果应用程序仅读取数据集并将其发送到视图，则该应用程序没有物理模型层和关联的类。 在这种情况下，数据集担当模型对象的作用。
* 视图。 视图是显示应用程序用户界面 (UI) 的组件。 通常，此 UI 是用模型数据创建的。 Products 表的编辑视图便是一个视图示例，该视图基于 Product 对象的当前状态显示文本框、下拉列表和复选框。
* 控制器。 控制器是处理用户交互、使用模型并最终选择要呈现的视图来显示 UI 的组件。 在 MVC 应用程序中，视图仅显示信息；控制器则用于处理和响应用户输入和交互。 例如，控制器处理查询字符串值，并将这些值传递给模型，而模型可能会使用这些值来查询数据库。

2.2.2 ASP.NET MVC 框架具有以下功能：

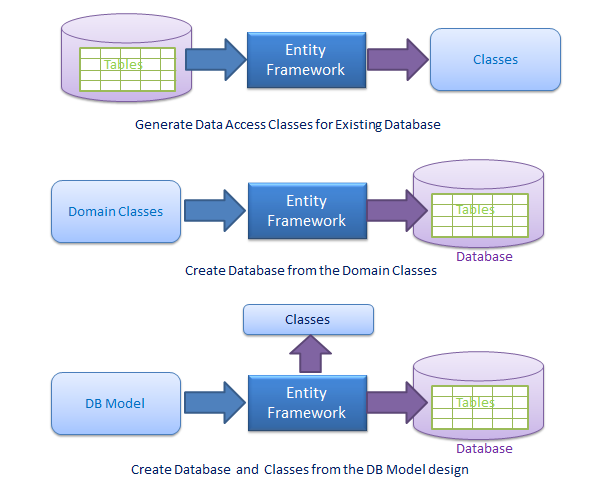
* 应用程序任务（输入逻辑、业务逻辑和 UI 逻辑）的分离、可测试性和测试驱动的开发 (TDD)。 MVC 框架中的所有核心协定都基于接口并且可使用 mock 对象进行测试，mock 对象是模仿应用程序中实际对象的行为的模拟对象。 您可以对应用程序进行单元测试，而不必在 ASP.NET 进程中运行控制器，这使得单元测试既快速又灵活。 您可以使用任何与 .NET Framework 兼容的单元测试框架。
* 可扩展且可插入的框架。 设计 ASP.NET MVC 框架组件的目的是为了可以轻松地替换或自定义它们。 您可以插入自己的视图引擎、URL 路由策略、操作方法参数序列化以及其他组件。 ASP.NET MVC 框架还支持使用依赖项注入 (DI) 和控制反转 (IOC) 容器模型。 DI 使您能够将对象注入到类中，而不是依靠类来创建对象本身。 IOC 指定某个对象是否需要其他对象，第一个对象应该从配置文件之类的外部源中获取第二个对象。 这样，测试会更加轻松。
* 广泛支持 ASP.NET 路由，ASP.NET 路由是一个功能强大的 URL 映射组件，它允许您生成具有易于理解的可搜索 URL 的应用程序。 URL 未必包含文件扩展名，并且旨在支持非常适合搜索引擎优化 (SEO) 和具象状态传输 (REST) 寻址的 URL 命名模式。
* 支持将现有 ASP.NET 页面（.aspx 文件）、用户控件（.ascx 文件）和母版页（.master 文件）标记文件中的标记用作视图模板。 您可以将诸如嵌套母版页、内联表达式 (<%= %>)、声明性服务器控件、模板、数据绑定、本地化等现有 ASP.NET 功能与 ASP.NET MVC 框架结合使用。
* 支持现有 ASP.NET 功能。 ASP.NET MVC 允许您使用一些功能，如 Forms 身份验证和 Windows 身份验证、URL 授权、成员资格和角色、输出和数据缓存、会话和配置文件状态管理、运行状况监视、配置系统以及提供程序体系结构。

## 2.2 Entity Framework

2.2.1 实体框架是支持开发面向数据的软件应用程序的一系列技术。实体框架使开发人员能够以特定于域的对象和属性（如客户和客户地址）形式使用数据，而不必自己考虑存储这些数据的基础数据库表和列。借助实体框架，开发人员在处理数据时能够以更高的抽象级别工作，并且能够以相比传统应用程序更少的代码创建和维护面向数据的应用程序。有关更多信息，请参阅实体框架简介。

概述

实体框架是一种对象关系映射机制 (ORM)，它能够减少 .NET Framework 开发人员的面向对象编程环境与关系数据库环境之间的不匹配情形。它使开发人员基本上可以通过熟悉的面向对象的技术来与应用程序的概念模型进行交互。在实体框架中，您能够以特定于域的对象和属性（如 客户 和 客户地址 ）形式使用数据，而不必考虑存储此数据的基础数据库表和列。开发人员可以对概念模型发出数据访问操作，然后实体框架会将该操作转换为关系数据库操作。



Pic 2.1 Entity framework的模式

2.2.2 实体框架应用程序有两个主要的层：

* 建模层
* 对象层

2.2.3 建模层包含三个组件：

* 一个概念模型，其中包含特定于域的实体类型和关系（基于实体数据模型 (EDM)）
* 一个数据库架构，用于定义表和关系
* 概念模型与数据库架构之间的一个映射

2.2.4实体框架使用映射组件将针对实体对象的操作（例如创建、读取、更新和删除）转换为数据库中的等效操作。

实体框架的对象层包含类型化的公共语言运行时 (CLR) 对象，这些对象反映了概念模型中定义的实体和关系。这些对象可以由编程语言使用。类型的确切格式由您提供给实体框架的选项控制。

映射和建模

* 创建映射层和对象层有几种不同的方法：
* 可以使用实体框架工具从现有数据库生成模型。这会生成默认的概念模型和映射，然后可以使用 Entity Data Model Designer 进行自定义。也可以使用工具（例如 Entity Data Model Designer）以图形方式创建概念模型，然后根据该模型中的工具所生成的元数据来生成数据库。
* 可以使用 Code First 开发方法以代码方式定义概念模型。实体框架会根据所定义的对象类型和其他配置来推断概念模型。基于您定义域类型的方式和在代码中提供的其他配置信息的组合，在运行时生成映射元数据。模型可以映射到现有的数据库，也可以从模型生成新的数据库。

2.2.5 使用对象

通过实体框架的对象层，您可以执行以下任务：

* 针对概念模型运行查询。
* 具体化从数据源作为对象返回的数据。
* 跟踪针对对象做所的更改。
* 将对象更改传播回数据源。
* 将对象绑定到控件。