**技术文档**

# 一、系统说明

本次设计的目标系统为：图书馆座位预订系统。解决的主要问题为：图书馆座位资源的均衡管理。

本系统主要基于 Asp.Net Core MVC 框架以及 B/S结构进行设计。

# 二、文档面向对象

开发人员及测试人员

# 三、文档用词说明

LSS：Library Seating System

# 四、系统架构图

图 1 系统架构图

# 五、系统技术选型

* 系统框架：ASP.Net Core MVC；
* SDK：.Net Core 3.1；
* 对象关系映射框架（ORM）： Entity Framework Core；
* 后端开发语言：C#
* 前端开发语言：HTML、CSS、JS、JQuery等
* 前端框架：Bootstrap
* 数据库选型：MySQL 5.7
* 开发平台：Windows 10
* 开发环境：Visual Studio 2019

# 六、设计开发工具

* 文档制作：Microsoft Word
* 图表绘制：Microsoft Visio
* 数据库设计：PowerDesigner
* 数据库管理工具：Navicat

# 七、系统目录结构：

1. LSS.Data：实体层。存放与数据库表对应的C# 实体类，通过ORM对象关系映射自动生成该实体层中的C#实体类。
2. LSS.Infratructure：基础结构层。
3. Untility：主要为全局工具方法或工具类，用于提供全局的辅助方法或辅助类。命名规范详见本文档第八部分第1条。
4. DB.EFCore：用于实现LSS.Service中提供的接口中的各种方法（即数据访问层的增、删、改、查等操作）。

命名规则为详见本文档第八部分第2条。

1. LSS.Marpper：对于业务逻辑的处理，主要是完成对具体业务流程的实现。

同时提供系统具体业务流程处理的公共方法。

命名规则为详见本文档第八部分第3条。

1. LSS.Service：实体类的接口 IRepository，提供对实体操作的相关接口。并在DB.EFCore中具体实现。

命名规则为详见本文档第八部分第4条。

1. LSS.UI：
2. Controllers：存放控制器文件，每个控制器应实现对应业务的业务逻辑控制。

命名规则为详见本文档第八部分第五条。

1. ViewModels：主要是与前端进行交互的,用于模型绑定和模型验证的实体类。

命名规则为详见本文档第八部分第六条。

1. Views:为Razor页面，文件夹命名为对应控制器的名称，即【控制器名】。在对应文件夹下的页面命名中，使用对应控制器的Action名。

# 八、命名规范

**总体命名规范如下：**

变 量：全名加注释

函数名：每个单词首字母大写

注释函数功能

注释函数参数

注释函数返回值

属 性：首字母大写

每个单词首字母大写

类 名：首字母大写

1. LSS.Infratructure 中Untility部分中，在对工具方法进行命名的过程中，必须使用【方法含义】+【\_Until】

在对工具类进行命名的过程中，必须使用【类名】+【.Until】

1. LSS.Infratructure中EFCore部分中，在对类进行命名的过程中，需使用【类名】+【.DB】；
2. LSS.Marpper中对于公共类的命名，格式为【Public】+【类名】+【.Marper】；对于公共方法的命名，格式为【Public】+【方法名】+【\_Marper】；

对于其他相关类的命名格式为：【类名】+【.Marper】；

对于其他相关方法的命名格式为：【方法名】+【\_Marper】

1. LSS.Service中接口方法命名格式为：【方法名】+【\_DB】
2. LSS.UI中Controllers部分控制器命名格式为：【业务名】+【Controller】对应控制的Action命名为：【业务功能】
3. ViewModels的命名格式为：【类名】+【ViewModel】