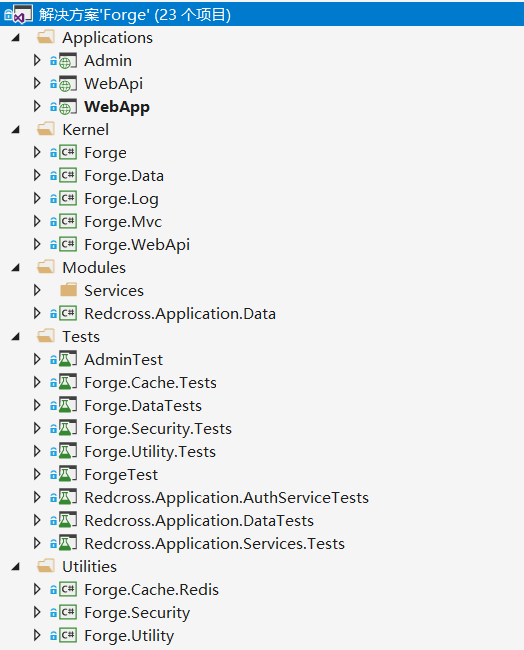
# 红十字会项目开发说明文档

### 一、项目结构



Applications 文件夹中项目为表示层，包含一个后台管理网站项目（Admin），一个学员中心客户端网站项目（WebApp）以及一个学员中心客户端Web接口项目（WebApi）。

Kernel 文件中项目为框架核心组件，Forge.Mvc项目中包含Mvc项目的扩展，包括Controller扩展以及Filter的扩展

Modules 文件夹包含数据访问层实现以及业务逻辑层实现。

数据访问层实现（Redcross.Application.Data）包含数据库模型，数据交换模型，枚举以及一个管理数据库模型与数据交换模型的映射管理类。

Services 文件夹中包含业务逻辑层全部实现，其中包含多个服务。如： Redcross.Application.Services 项目中包括除用户权限相关以及学员认证相关的所有业务逻辑。

Tests 文件夹包含多个单元测试项目，用于测试项目框架以及相关业务的正确性。单元测试框架采用 NUnit 3 ，vs本地运行单元测试需要使用插件Nunit 3 Test Adapter。

同时作为git仓库服务器端搭建有 CI 服务。每次提交带服务器端前应当自行运行所有测试再提交推送至服务器进行测试，服务器端全部测试通过后可申请将代码合并至主仓库中。以此避免由于不可预知的问题造成的代码混乱或崩溃问题。

Utilities 文件夹包含多个工具类项目。Forge.Utility项目为工具类项目，包含文字处理的扩展以及图像处理的扩展。

该项目使用 Autofac 作为IoC框架，表示层 业务逻辑层 数据访问层之间的依赖由采用IoC控制。实际代码中不应在各层之间相互 new 对象实例，所有依赖的对象实例均由依赖注入容器完成创建。

### 二、详细说明

#### 1.数据库访问框架

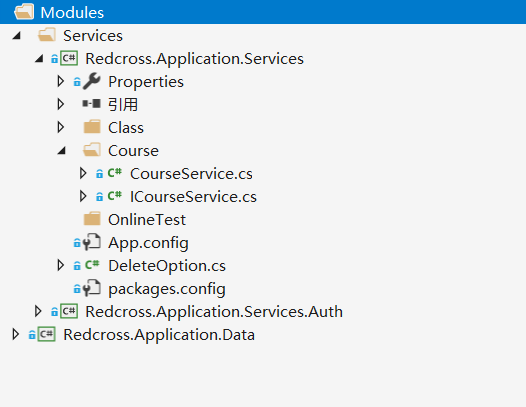
数据库访问采用EntityFramework 6.2框架，Code First开发模式。同时使用Migration迁移，使用vs 包管理控制台可进行数据库迁移配置。

命令 Add-Migration versionname 用于更新本地数据迁移脚本。

命令 Update-Database 用于推送更新至数据库连接所标识的服务器数据库。

详细文档请查阅：<https://msdn.microsoft.com/en-us/library/jj591621(v=vs.113).aspx>

#### 2.服务编写规范



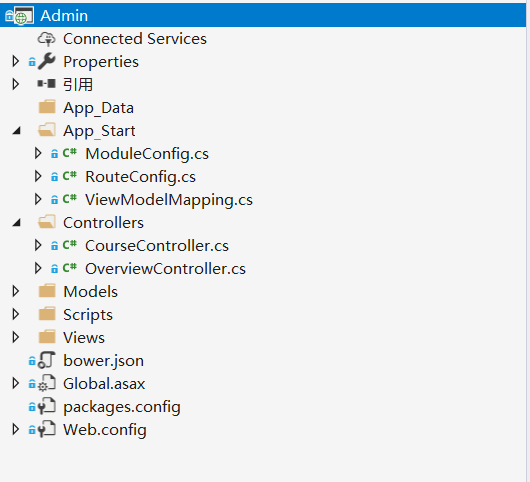
包含主要业务逻辑的服务接口应归放在 Redcross.Application.Services 项目下，使用模块命名相应的文件夹。如 课程管理 模块所有服务接口以及相应实现放置于 Course 文件夹下，包括一个服务接口 ICourseService 以及对应实现 CourseService。



服务实现应继承自Forge.Service.ServiceBase 服务接口实现中应包含一个或多个数据库访问所需的 DbContext 类型的字段，如 AppContext \_context 字段。同时应当包含一个或多个带有对应DbContext 类型参数的构造函数 ，如

public CourseService(Data.Entity.AppContext context)。

注：服务编写完成后应到表示层所包含的项目中进行注册，否则实际运行的项目将无法通过IoC容器获取到相应的服务组件。



表示层 项目应包含一个模块注册类 ModuleConfig ，该类中包含一个组件注册方法

public static void Register(ContainerBuilder builder)，

所有组件注册应包含在该方法中。



服务注册相关的代码应包含在 注册服务组件 区域内。如：

builder.RegisterType<CourseService>().As<ICourseService>();

意为将 CourseService 类型注册为 ICourseService ，在通过IoC容器调用类型为 ICourseService 的对象时将由IoC容器创建或提供一个 CourseService 类型的实例。

详细文档请查阅：<http://autofac.readthedocs.io/en/latest/getting-started/index.html>



Controller 对象应全部继承自 Forge.Mvc.Controller.ControllerBase 。同时包含一个或多个所需服务接口的字段 如：

readonly ICourseService \_service。

同时包括一个带有对应服务接口为参数的构造函数 如：

public CourseController(ICourseService service)。

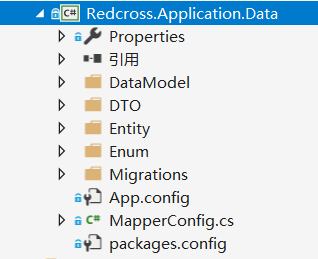
#### 3.模型映射管理

模型映射使用 AutoMapper 框架，通过注册映射关系可由AutoMapper自动映射源模型到目标模型。采用该方式可大量减少模型转换所需要的代码量，节省开发时间。

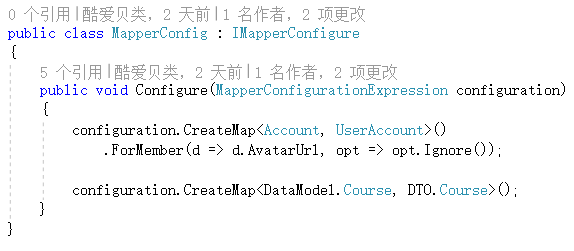
本项目中模型映射的注册均由接口 Forge.Data.Configure.IMapperConfigure 完成。

主要使用到模型映射的部分分为两个模块：

数据库模型（DataModel） → 数据交换模型（DTO）:



此处映射管理实现在 MapperConfig 中



实现 Forge.Data.Configure.IMapperConfigure 接口中 Configure 方法。

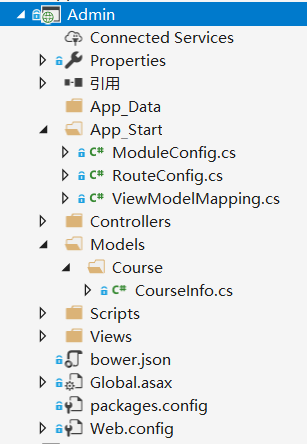
configuration.CreateMap<DataModel.Course, DTO.Course>();

意为创建一个由 源模型（TSource） DataModel.Course 与 目标模型（TDestination） DTO.Course 的映射关系。该接口默认根据模型中名字相同的属性进行转换赋值。如有特殊映射 如名称不同或不映射则需要调用ForMember(d => d.AvatarUrl, opt => opt.Ignore()) 接口进行单独配置。

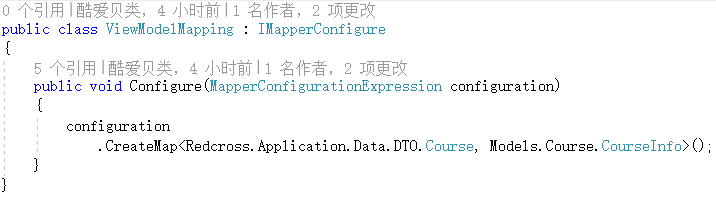
以上接口表示 目标对象 中 AvatarUrl 属性的配置选项设定为 opt.Ignore() 忽略，意即在对象映射过程中忽略 AvatarUrl ，不会从 **源对象** 中匹配任何属性对其进行赋值。

详细文档请查阅：<http://automapper.readthedocs.io/en/latest/index.html>

数据交换模型（DTO） → 视图模型（ViewModel）：



此处映射管理实现在 ViewModelMapping 中

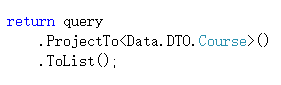


以上为对象的映射管理配置，在项目代码中使用模型映射时需要调用

命名空间 AutoMapper.QueryableExtensions 下的EF查询扩展方法ProjectTo<Data.DTO.Course>()

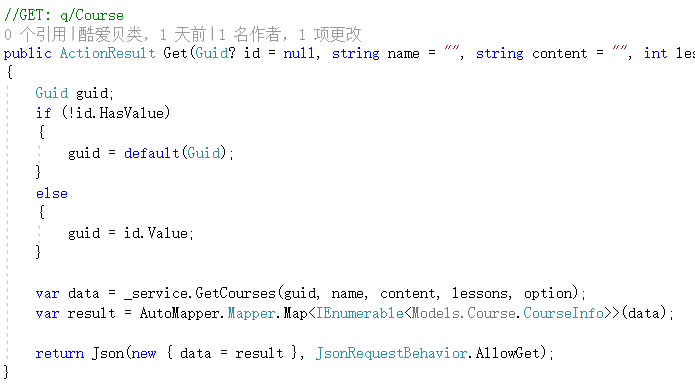
如：

CourseService 实现 GetCourses 方法 第 183 行



query 为 System.Linq.IQueryable<out T> 对象。

Admin项目中的Controller CourseController 中 Get 方法 第 35 行



AutoMapper.Mapper.Map<IEnumerable<Models.Course.CourseInfo>>(data)

意为将源对象转换为目标类型 IEnumerable<Models.Course.CourseInfo>