# General Key Points

Main points are

1. Development toolkits: ASP.NET MVC 4.0, .NET 4.5, MS VS2010, MS SQL Server 2008.
2. Any project of the solution must have enabled build option: **Treat warnings as errors!** Exceptions are projects which contain wrappers/adapters for 3rd party components (if these components generate warnings)
3. [Design code standards](http://idesign.net/idesign/download/IDesign%20CSharp%20Coding%20Standard.zip) (**TBD**). Currently none code standard is used.
4. Project naming convention (**TBD**).
5. Use [SOLID](http://en.wikipedia.org/wiki/Solid_%28object-oriented_design%29) principles during development. In particular we use [IoC](http://en.wikipedia.org/wiki/Inversion_of_control) container. It is decided to use [Unity](http://unity.codeplex.com/) due to some investigations and it is Microsoft product and it has support. See [IoC](http://en.wikipedia.org/wiki/Inversion_of_control) container comparison [here](http://www.codeproject.com/Articles/43296/Introduction-to-Munq-IOC-Container-for-ASP-NET). For the time being [IoC Unity container](http://unity.codeplex.com/) is used explicitly. In the future if it is necessary we will include [service locator](http://commonservicelocator.codeplex.com/). It has not being done due to lack of time.
6. Code comments (**TBD**)
7. All local specific files have ‘.local’ postfix. For example configuration files are local specific, some batch files and so on. All local specific files are included into the project but they are not added into the source control (they also are added into the .gitignore file). For that reason you can get not working project especially if you get sources at first time. In this case you MUST create duplicates, delete ‘.local’ extension and point your local settings. All configuration files must be added as local specific!
8. All 3rd party components must be added via NuGet! It means that all projects in the solution will use the same version of the component and will be updated at the same time all together. The reason – project consistency. All custom NuGet package projects are stored in the NuGet solution folder. I have greated example of the NuGet package and added description file how and what to do.
9. Not to be confused and avoid namespace conflict problems all data context (LINQ or EF) classes must be with postfix ‘Record’!

Pay special attention to [singleton](http://en.wikipedia.org/wiki/Singleton_pattern), [plugin](http://www.martinfowler.com/eaaCatalog/plugin.html) and [strategy](http://en.wikipedia.org/wiki/Strategy_pattern) design patters. It is mandatory to understand how library including and its usage works.

# System Description

System consists of 3 parts: [Public](#_Public) site and [CMS](#_CMS) (admin part).

# System Databases

The Corbis system has two databases: [Corbis.Main](#_Corbis.Main_database) and [Corbis.Log](#_Corbis.Log_database). It is done due to performance issue. The main reason is not to load database with main data [Corbis.Main](#_Corbis.Main_database).

## Corbis.Main database

This database contains main and actual application data.

## Corbis.Log database

This database contains application logs: errors, warnings, debug info and so on. Database is used to easy analyze system work.

## Database projects

The Corbis solution has corresponding DB projects for the described databases: **Corbis.DB.Main** and **Corbis.DB.Log**. They are used to simplify process of database deployment and database management. All database projects are located in the **@SolutionRoot\DB** solution folder.

## One click database deployment

Logic of one click database deployment is implemented via batch file. The batch file path is **@SolutionRoot\DB\Deployment\DbDeployment.bat.local**. Deployment details are described in the [text file](../DB/Deployment/_README.txt) in the batch file owned folder.

## Session database

Session must be stored in database. It is horizontal scalability requirement (WebFarm). How to install session database is described in the already mentioned [text file](../DB/Deployment/_README.txt).

# Shared components

All system shared components are located in the **@SolutionRoot\Share** solution folder. These components have common logic for all parts on the system: CMS and Public.

## Corbis.Common project

This library contains only common logic and must not have any application specific logic. It means that this library can and must be reused in any other application. Common logic means some usefull utils, .NET framework extensions, common classes and interfaces declarations. **This library must not reference to the other projects of the solutions!**

Corbis system consists of several parts as it was mentioned before. For application identifying application descriptor configuration section is used. The application details (application name, version) automatically are used by the [logging component](#_Corbis.Logging_project). Application descriptor section is defined in **@SolutionRoot\Share\Corbis.Common\Configuration\ApplicationDescriptorSection** project folder.

## Corbis.Logging project

It is logging component. Any solution project (except for [Corbis.Common](#_Corbis.Common_project)) can reference to this project. It means that you have possibility to use logging on any application level (DAL, BL, Presentation layers).

To use logging you must do 2 steps:

1. Register logging in the application entry point (in our case it is Global.asax file MvcApplication.ApplicationStart method). Details are described [here](#_Logging_registration).
2. Configure logging. Details are described [here](#_Logging_configuration).

Likewise [Corbis.Common](#_Corbis.Logging_project) the **Corbis.Logging** must not have any application specific logic and this project can and must be reused in other applications. It must not have reference to any solution project except for [Corbis.Common](#_Corbis.Common_project)!

### Loggers

Logging component can have one or more registered loggers. Loggers are registered in logging configuration section. [Strategy](http://en.wikipedia.org/wiki/Strategy_pattern) is used to load loggers. So for test purposes we can use one set of loggers, for release mode another and so on. Any logger must be inherited from ILogger interface or LoggerBase abstract class. Any existing log system (log4net, nlog, …) can be adapted for the usage in the system ([adapter design pattern](http://en.wikipedia.org/wiki/Adapter_pattern)). It simplifies logging and we must not add reference to multiple 3rd party components and this approach avoid refactoring if we decide to change log system for some reasons.

### Logging registration and usage

Logging component usage is simplest. It is initialized via 1 line:

Corbis.Logging.LogManagerProvider.Initialize("LoggingSection");

Take into account configuration is implemented via configuration file only. Custom configuration API was not implemented. If it is necessary then it will be done in the future.

Usage:

Logging.LogManagerProvider.Instance.WriteError(…);

Logging.LogManagerProvider.Instance.WriteDebug(…);

Logging.LogManagerProvider.Instance.WriteWarning(…);

### Logging configuration

The Corbis.Logging project has several classes which describe configuration section structure and example of section. It is located in **@SolutionRoot\Share\Corbis.Logging\Core\Configuration** project folder. In order to configure logging component you must do 2 steps:

1. Register configuration section
2. Add configuration section

# Public

It contains web interface for public browsers. [CMS](#_CMS) functionality is based on Public functionality (has or the same or extended or a bit changed => Public is subset in [CMS](#_CMS) in some sence). It means that we can use OOP principles for implementation and implement Public and [CMS](#_CMS) independent as much as it possible. It is key point for application structure.

## Corbis.Public.Entity project

This project contains only domain object type definitions. Any entity type is data container only. It means that it must not contain application business logic. All entities must have portion only necessary data for Public web site. These entities can be extended with any data in [Corbis.CMS.Entity](#_Corbis.Public.Entity_project) project (via inheritance for example). This project must not reference to any project except for [Corbis.Common](#_Corbis.Common_project) or [Corbis.Logging](#_Corbis.Logging_project).

## Corbis.Public.Repository.Interface project

This project contains contract declarations for manipulating with [domain entities](#_Corbis.Public.Entity_project). It is separated from [Corbis.Public.Entity](#_Corbis.Public.Entity_project) to separate description logic from manipulation logic (with the same objects we can manipulate with different ways). Likewise [Corbis.Public.Entity](#_Corbis.Public.Entity_project) project is has part of all possible logic. It means [Public](#_Public) contains GET logic, [CMS](#_CMS) extends it with SET logic in [Corbis.CMS.Repository.Interface](#_Corbis.CMS.Repository.Interface_pro) project via inheritance.

[Repository contract implementer](#_Corbis.Public.Repository) encapsulates inside business logic and data access logic.

Public site uses this repository contract only for manipulation. Specific repository implementation (in our case it is [Corbis.Public.Repository](#_Corbis.Public.Repository)) is loaded via strategy design pattern. It means that Public Web site has reference to this library and does not have reference to the specific implementation. It is good point for application extensibility. We can replace repository implementation with repository implementation only without recompilation or any refactoring of other application part. For example we can replace data storage from MSSQL to Oracle or we can implement nosql case for claude.

## Corbis.Public.Repository

ANY REPOSITORY MUST BE STATELESS AND THREAD SAFE !!!!

It is necessary for horizontal scalability (WebFarm). Another reason is performance. We can use repositories as singletons!

The repository library contains EntryPoint class for repository implementer providing (plugin design pattern)

# CMS

## Corbis.CMS.Entity project

## Corbis.CMS.Repository.Interface project