# Application Architecture Guidelines

**Trainer: venkatshiva.reddy@gmail.com**

## 1. Introduction

This document is an overview of software qualities, principles, patterns, practices, tools and libraries.

## 2. Code Qualities

Make software easier to change, debug, extend.

### 2.1 Encapsulation

Encapsulation means any kind of hiding
- Encapsulation of data
- Encapsulation of methods
- Encapsulation of other objects
- Encapsulation of type (behind an abstract class or interface)

Principles:
- Find what varies and encapsulate it (behind an abstract class or interface)

Indicators of missing encapsulation
- Un-maintainable system

### 2.2 Cohesion

Cohesion refers to how much (or how little) the internal parts of something are working on the same issue, and how well they relate to each other. It is the quality of singleness of purpose, and it makes entities (classes, methods) easier to name and understand.

- Method Cohesion:

  The methods should do a single thing.
  This will make them easy to name and helps also with debugging. You know exactly where to look to find the bug, because the responsibility for doing this is clearly assigned to the properly named method.
- Class Cohesion :

  Breaking the problem into classes with responsibilities makes the system easier to test, debug, and therefore maintain.

Related Principles :
- Single Responsibility Principle
- Separation of Concerns

Indicators of Weak Cohesion:
- Difficulty naming

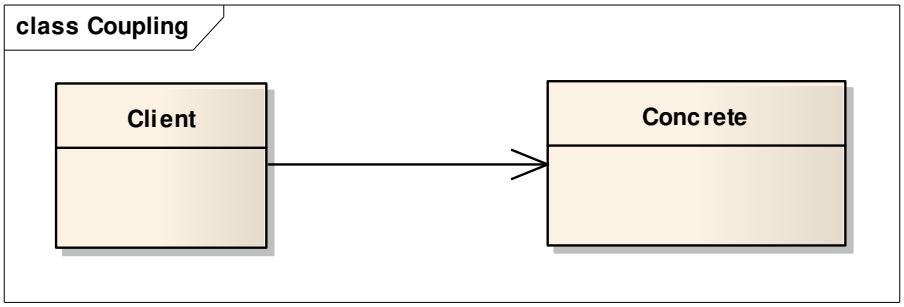
  Difficult-to-name methods are a good sign that you have weak method cohesion.
  I would like the names of classes, methods, and other entities (delegates, packages, and so on) to reveal their intentions in their names. When a class or method has a long name, or a vague name, the reason often is that a really informative name is very difficult to create, due to the fact that the class or method does a number of different things. This, of course, is weak cohesion, and causes lots of other problems anyway. When I cannot name things the way I want to, I suspect that my entities do too much.
- Large classes and methods

  When a class or a method gets big and requires lots of scrolling in your IDE, or when it is hard to see well, I usually suspect weak cohesion. This is not an absolute; algorithms themselves can get large and yet still be about one responsibility.

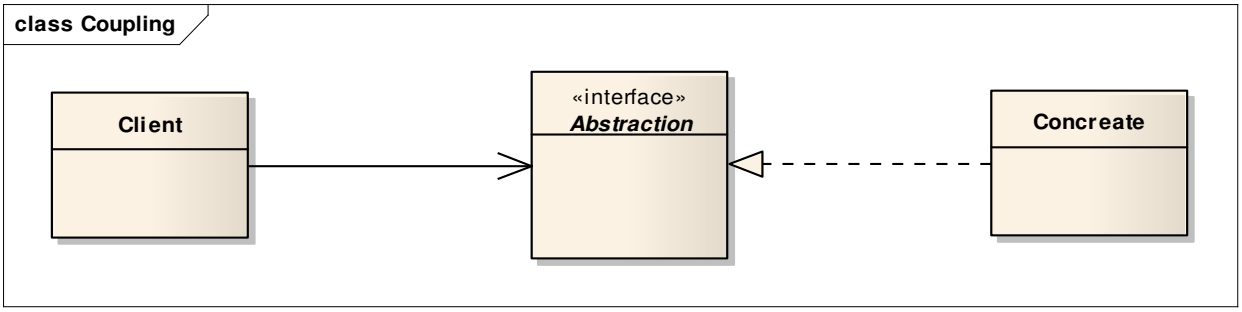
## 2.3 Coupling

Coupling refers to the degree of direct knowledge that one class has of another.
**Strong coupling** occurs when a dependent class contains a pointer directly to a concrete class
**Loose coupling** occurs when the dependent class contains a pointer only to an interface that the concrete class implements

- Strong/tight coupling




- Loose coupling




Principles :

- Dependency Inversion Principle
- The Open Close Principle

Indicators of Accidental or Illogical Coupling:

- Unexpected side effects
  When a change in one part of the system changes something in another part of the system, and this is surprising, unexpected, and illogical to you, then most likely there is coupling in the system that was not intended or does not make sense.
- Hesitancy
  When you find yourself hesitant or resistant to making a change to the system, sometimes this is simply your subconscious telling you that you know the system has coupling in it that is going to "get you" when you try to change it.
- Hard to test in isolation
- Bidirectional coupling

## 2.4 (No) Redundancy - DRY (Do Not Repeat Yourself)

Anything that could change should be in a single place in the system.

This typically requires using several small methods. The extra cost is minimal, but it eliminates duplication and often saves many future problems. Duplication is bad not only because of the extra work in typing things in multiple times, but because of the likelihood of

something changing in the future and then you perhaps forgetting to change it in all the required places.
A powerful relationship exists between redundancy and coupling. If redundant code is present and it becomes necessary to change one of the sections of code, it is very likely the other section will need to be changed as well. Hence, these two sections of redundant code are also coupled to each other.
Principles:

- One rule, one place
- DRY

Design Patterns:

- Template Method
- Strategy

Indicators of Redundancy:

- Redundant changes

When you find yourself making the same change in multiple places, clearly the thing you are changing is redundant.

### 2.5 Readability

A measure of how easy it is to read and understand code.

First of all, you may not be the only person who has to read your code. It is quite likely that another developer will have to maintain it in the Future. Secondly, you may have to read the code in a few months, or even in a few years, when other projects and other clever code have taken over your active memory, and at that point it can seem like reading something written by someone other than yourself.

When you are writing code and you need to implement some piece of function, just pretend it already exists, give it an "intention-revealing name," write the method call to it, and move on (implementing this function later). In other words, coding becomes a series of calls to functions that are named in a way that clearly describes their use.

An "intention-revealing name" is a name that clearly communicates what the function is responsible for doing. This results in very readable code because at the larger module level, the reader is seeing the intent of the code, not each little implementation.

Whenever we feel the need to comment something, we write a method instead. This results in shorter, more cohesive methods.

Standards:

- Naming Conventions

### 2.6 Testability

A measure of how easy it is to create test criteria.

Testable code is code that can be tested in isolation and without having to worry about how it is coupled to other modules or entities.
Testability is strongly correlated to the other practices:

- Cohesive code is easier to test because the code is only about one thing.
- Loosely coupled code is easier to test than tightly coupled code because there are minimal interactions to worry about.
- Redundant code is not harder to test in itself, but requires more tests to cover the redundancy. Therefore, the testability of the entire system degrades when you have more redundancy.
- Readable code is easier to test because method names and parameters describe precisely what each is supposed to do.

- Encapsulated code is easier to test because it will have little, if any, coupling to other code.

Principles :

- Dependency Inversion Principle

Addison Wesley - Emergent Design @2008, Chapter 7
Addison Wesley - Essential Skills for the Agile Developer: A Guide to Better Programming and Design @2011
http://www.netobjectives.com/resources/articles/code-qualities

3. **System Qualities**

| | |
|---|---|
| Conceptual Integrity | Conceptual integrity defines the consistency and coherence of the overall design. This includes the way that components or modules are designed, as well as factors such as coding style and variable naming. |
| Maintainability | Maintainability is the ability of the system to undergo repair and evolution. The ease with which a software system or component can be modified to correct faults, improve performance or other attributes. |
| Reusability | Reusability defines the capability for components and subsystems to be suitable for use in other applications and in other scenarios. Reusability minimizes the duplication of components and also the implementation time. |
| Availability | Availability defines the proportion of time that the system is functional and working. It can be measured as a percentage of the total system downtime over a predefined period. |
| Interoperability | Interoperability is the ability of a system or different systems to operate successfully by communicating and exchanging information with other external systems written and run by external parties. |
| Manageability | Manageability defines how easy it is for system administrators to manage the application, usually through sufficient and useful instrumentation exposed for use in monitoring systems and for debugging and performance tuning. |
| Performance | An application performance is measured in terms of its :response time, and resource utilization under a given workload |
| Scalability | An application's ability to handle additional workload, without affecting performance by adding additional resources such as : processor, memory and storage capacity |
| Reliability | The ability of the system to keep operating over time. Reliability is usually measured by mean time to failure |

| | |
|---|---|
| Security | A measure of the system's ability to resist unauthorized attempts at usage and denial of service, while still providing its services to legitimate users |
| Supportability | Supportability is the ability of the system to provide information helpful for identifying and resolving issues when it fails to work correctly. |
| Testability | Testability is a measure of how easy it is to create test criteria for the system and its components, and to execute these tests in order to determine if the criteria are met. |
| Usability | Usability defines how well the application meets the requirements of the user and consumer by being intuitive, easy to localize and globalize, and resulting in a good overall user experience. |
| Flexibility | The ease with which a system or component can be modified for use in applications or environments, other than those for which it was specifically designed |

http://msdn.microsoft.com/en-us/library/ee658094.aspx
http://msdn.microsoft.com/en-us/library/bb402962.aspx

## 4. **Principles**

### 4.1 GoF Principles

- Objects are things with responsibilities.
- Design to interfaces
- Favor aggregation over inheritance
- Find what varies and encapsulate it (behind an abstract class or interface)
- Separate use from construction

### 4.2 SOLID Principles

- Single Responsibility Principle

The principle states that every object should only have one reason to change and a single focus of responsibility. By adhering to this principle, you avoid the problem of monolithic class. By having concise objects, you again increase the readability and maintenance of a system.

- The Open-Closed Principle

Open for extension, but closed for modification.
Extend software's capabilities without changing it
Design the software so new functionality can be added in separate, distinct modules, with the integration cost being minimal

- Liskov Substitution Principle

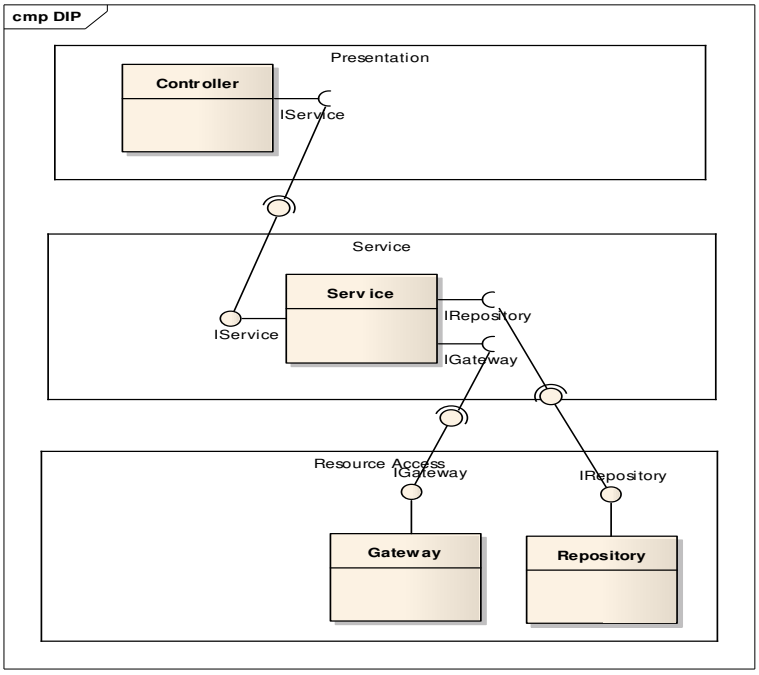
The principle dictates that you should be able to use any derived class in place of a parent class and have it behave in the same manner without modification.
The Principle reminds you that subclasses should act as you would expect a base class to be used, without the need to downcast to check for specific subclass behavior.

- Interface Segregation Principle

States that clients should not be forced to depend on interfaces they don’t use.

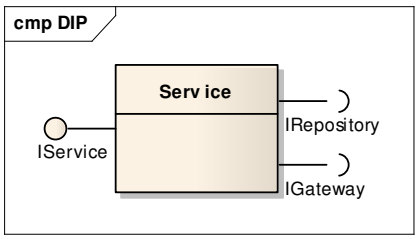
It is all about separating fat interfaces into small, specific groups of related functionality. Splits fat interfaces into separate related groups of contracts, making it easier to use and more understandable in your code.

- Dependency Inversion Principle
  - High-level modules should not depend on low-level modules. Both should depend on abstractions.
  - Abstractions should not depend upon details. Details should depend upon abstractions.




Dependency Injection (DI) is the act of supplying a low level or dependent class via a constructor, method, or property.

Dependency Inversion and Dependency Injection both refer to the same process of decoupling your code. The principle is all about isolating your classes from concrete implementations and having them depend on abstract classes or interfaces. It promotes coding to an interface.




http://www.globalnerdy.com/wordpress/wpcontent/uploads/2009/07/dependency_inversion_principle.jpg

### 4.3 Other Principles

- Keep It Simple Stupid (KISS)
  Avoid any unnecessary complexities.

- Don't Repeat Yourself (DRY)
  Variation: "One rule, one place"
  Have only one place where a rule is implemented.

  Tool to find and duplicate code:
  http://getatomiq.com/

- Tell, Don't Ask
  Tell objects what actions you want them to perform rather than asking questions about their state and then making a decision on what action to perform.

- You Ain't Gonna Need It (YAGNI )
  Only include functionality that is necessary

- Separation of Concerns (SoC)
  Is the process of divide your application into distinct features with as little overlap in functionality as possible. Layered designs are often based on separation of concerns (ex: presentation layer, business logic layer, data access layer).

Packt Publishing - ASP.NET 3.5 Application Architecture and Design @2008, Chapter 6
Wrox - Professional ASP.NET Design Patterns @September @2010, Chapter 1
Addison Wesley - Emergent Design @2008, Chapter 8

Pluralsight - Principles of Object Oriented Design

http://www.objectmentor.com/omSolutions/oops_what.html
http://www.globalnerdy.com/2009/07/15/the-solid-principles-explained-with-motivational-posters/

5. **Refactoring**

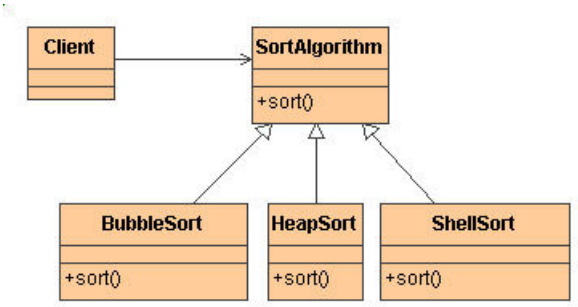
Code Smells:
http://www.codinghorror.com/blog/2006/05/code-smells.html
http://www.informit.com/articles/article.aspx?p=167891&seqNum=4

Tools:
CodeRush : VB Refactoring Add In

Wrox - Professional Enterprise .NET Chapter 3

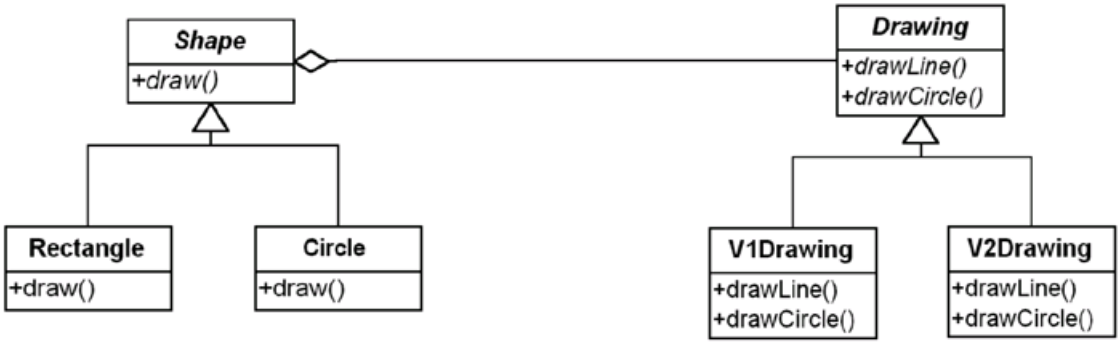
6. **Design Patterns**

6.1 Strategy
- define a family of algorithms. Conceptually, all of these algorithms do the same things. They just have different implementations
- lets the algorithm vary independently from clients that use it




6.2 Bridge

- decouple an abstraction from its implementation so that the two can vary independently




- the diagram illustrates the separation of the Shape abstraction from the Drawing implementation.
- it presents two variations present (shapes and drawing programs), each encapsulated in its own abstract class. The variations of shapes are encapsulated in the Shape class, the variations of drawing programs are encapsulated in the Drawing class. The pattern is about the relationship between these different abstractions

6.3 State

- separate behavior dependent on state from the object itself.

6.4 Adapter

- convert the interface of a class into another interface that the clients expect

6.5 Template Method

- uses inheritance to vary part of an algorithm

6.6 Mediator

- encapsulates how a set of objects interact

6.7 Facade

- to create a simpler interface in terms of method calls
- to reduce the number of objects that a client object must deal with
- to encapsulate or hide the original system

- to use the functionality of the original system and want to add some new functionality as well

### 6.8 Proxy

- The Proxy pattern acts as a surrogate for another object, enabling the proxy to control access to it and allowing it to add extra logic related to the operation.
- A virtual proxy is a placeholder for resource-intensive objects. The real object or methods on that object are called only when they are needed.
- A remote proxy provides a local representative for an object that resides in a different address space.

### 6.9 Observer

Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically

### 6.10 Factory Method

- The main objective of the Factory pattern is to hide the complexities of creating objects.
- The client will code against an interface or abstract class and leave the responsibility to the Factory class to create the concrete type.
- Typically a Factory class has a static method that returns an abstract class or interface.
- The client usually, but not always, supplies some kind of information; using the supplied information the Factory then determines which subclass to create and return.

### 6.11 Abstract Factory

- is used when you must coordinate the creation of families of objects
- factories encapsulate business rules for creating objects

### 6.12 Builder

- separtes assemblage/integration (of a complex object) from creation (of components)

### 6.13 Singleton

- ensure a class only has one instance, and provide a global point of access to it
- (we may need a single instance of a logging utility to process all logging requests in our application)

### 6.14 Object Pool

- Reuse and share objects that are expensive to create

Addison Wesley - Design Patterns Explained A New Perspective on Object-Oriented Design, 2nd Edition @2004
Addison Wesley - Emergent Design @2008
Wrox - Professional ASP.NET Design Patterns @September @2010, Chapter 5
Packt Publishing - ASP.NET 3.5 Application Architecture and Design @2008

VTC - Design Patterns
http://www.oodesign.com/

http://www.vincehuston.org/dp/all_uml.html
http://www.cours.polymtl.ca/inf3700/divers/nonSoftwareExample/patexamples.html

# 7. Enterprise Patterns

## 7.1 Presentation Layer

### 7.1.1 MVP (Model View Presenter)

Model :
- holds the data to show on the View
- see ViewModel

View :
– represents the Layout
  - WebForms : aspx and its code behind file
  - Winform : designer and its code behind file
- Instantiates the Presenter
- Request the presenter to do task
- May have a reference to the Model
- Naming Conventions
  - suffix interfaces with «View«
  - examples:
    - IPolicyAddControlView
    - ISettingEditControlView
    - ISettingListControlView

Presenter :
- Coordinates the updates between View and Model :
  - Responds to View requests
  - May know when data is updated (by subscribing to Events)
  - Updates View with data from the Model
- Naming Conventions :
  - suffix classes with «Presenter«
  - examples:
    - PolicyPresenter,
    - SettingEditPresenter
    - SettingListPresenter

Benefits :
- Presenter code can be tested
- Presenter code reuse

Pluralsight – Design Patterns Library : MVP
http://geekswithblogs.net/rajeshpillai/articles/mvp1.aspx
http://geekswithblogs.net/rajeshpillai/archive/2009/11/30/mvp2.aspx
Wrox - Professional Enterprise .NET, @2009 Chapter 10

### 7.1.2 MVC (Model View Controller)

View:
- displays the model data supplied from the Controller.

Controller:

- receives a request
- delegates tasks to be performed
- decides the view to be shown
- Naming Conventions:suffix classes with «Controller« (imposed by ASP.MVC framework)

#### 7.1.3 ViewModel

- Model objects specially designed for views. Domain objects are designed for the needs for domain model and it is representing the domain of our applications. On the other hand, View Model objects designed for our needs for views.
- Should contain only data that the View needs and nothing more
- The data should be anemic (no behaviour).
- in a Object Oriented aproch will contain Dtos (associated with Entities)
- in a Data Centeric aproch will contain DataSets/DataTables Or Xml

Naming Conventions
- Suffix with "VM"

### 7.2 **ServiceLayer**

#### 7.2.1 Services

Responsabilities :
- acts as a facade or entry point into the system.
- provides an interface that defines the operations available to the client.
- coordinates the application activity and delegates all business tasks to the domain model. This layer does not contain business logic and helps to prevent any non-business-related code from polluting the domain model project. The layer also transforms domain entities into data transfer objects that protect the inner workings of the domain and provide an easy API for the presentation layer to work with.
- It does not contain business logic and does not hold the state of any entities; however, it can store the state of a business workflow transaction.
- the service layer simply coordinates the business use case transaction and delegates work to the business objects for all the lower-level implementation details.
  many of the use cases in an enterprise application are "CRUD" (create, read, update, delete) use cases on domain objects
- coordinates the method calls to
  - domain entities
  - data access
  - gateways
- implements IService<TId, TDto, TQuery>

Naming Conventions :
- suffix classes with «Service «
- examples:
  - PolicyService
  - FormService

Wrox - Professional ASP.NET Design Patterns @September, 2010.pdf
Chapter 6, Service Layer

#### 7.2.2 The Document Message

- The Document Message pattern simplifies the communication by encapsulating all information within the body of the document (data transfer object).

- By using the Document Message pattern for all communication, you make it easy for the service method to evolve and include additional parameters without needing to change the signature of the method.

### 7.2.3 The Request-Response

- ensures that responses as well as requests use the Document Message pattern
- as with the Request object, the Response can also inherit from a base class, which can provide access to common properties like a generic message and success flag

```
CustomerSearchResponse RetrieveCustomers(CustomerSearchRequest request);
```

### 7.2.4 The Reservation Pattern

- There are times, when it is necessary to maintain the state of a long-running process during a complex transaction that requires several messages to be sent to complete a unit of work. For these situations, you can assign a reservation number to the first response. The client can use this reservation number in subsequent requests to allow the service layer to pick up a transaction. Typically, an expiration date is used to allow the reserved state to expire after a given time so it doesn't hold onto resources for an undefined amount of time.

### 7.2.5 The Idempotent Pattern

- an idempotent operation is one that has no additional effect if it is called more than once with the same input parameters
- ensure that repeat calls do not have undesirable effects on the state of a system

### 7.2.6 Data Tranfer Objects (DTOs)

- An object used to transfer data between subsystems
- Use DTOs to transfer data between Presentation and Services
- Use Automapper for custom mapping between DTOs and Entities
- Should be serializable
- The fields in a DTO are simple, being primitives, simple classes like strings and dates, or other DTOs objects. It should be a hierarchy.

### 7.2.7 Translator / Mapper

- Maps data between objects
- Keeps them independent of each other

### 7.2.8 Assembler

Use a Transfer Object Assembler to build an application model as a composite Transfer Object. The Transfer Object Assembler aggregates multiple Transfer Objects from various business components and services, and returns it to the client.

http://www.corej2eepatterns.com/Patterns2ndEd/TransferObjectAssembler.htm

## 7.3 **Domain Layer**

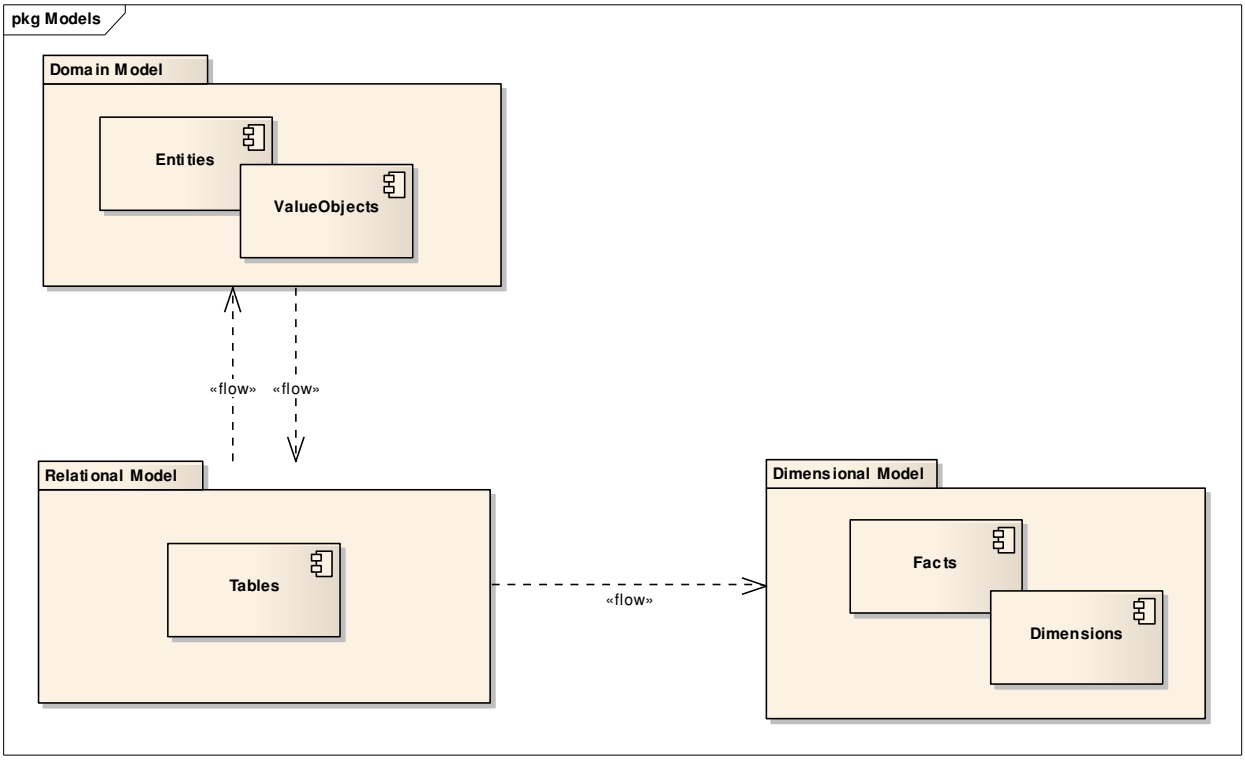
### 7.3.1 Entities

Responsabilities :

- a layer of objects that model the business area you're working in
- will handle all the application's business logic
- Object Data Modeling

http://www.agiledata.org/essays/mappingObjects.html

- are the things that require an identity, which will remain with it throughout its lifetime
- an object that is not defined by its attributes, but rather by its identity




Code Generation:

- For Entity Frameworkgenerate using ADO.NET Entity Data Model template (.edmx file). Use partial class to add behavior
- For NHibernategenerate using NHibernate Mapping Generator http://nmg.codeplex.com/

Recommendations:

- analyze and model the logical business entities of your application, rather than defining a separate business entity for every table.
- do not define separate business entities to represent many-to-many tables in the database;

Examples :

Policy
Form
Vehicle

### 7.3.2 Value Objects

- An object that contains attributes but has no conceptual identity

### 7.3.3 Aggregate

- groups logical entities and value objects
- a group of associated objects that are treated as a unit for the purpose of data changes

### 7.3.4 Aggregate Root

- an entity, which is the only member of the aggregate that any object outside the aggregate is allowed to hold a reference to
- an entity that acts as the logical way into the aggregate.
- implements IAggregateRoot

### 7.3.5 Domain Events

- are events that occur in your model that can be published for other services inside and outside the model to respond to.When the change in stateoccurs, a domain event is fired. A handler in the service layer responds.

- Involved clases:

| | |
|---|---|
| IDomainEvent | • identify domain events within the model |
| IDomainEventHandler<T> | • is the interface that handlers of the events must implement |
| IDomainEventHandlerFactory | • use this to obtain the collection of domain event handlers for a given domain event<br>• use StructureMap |
| DomainEvents.Raise<T>(T domainEvent) | • to raise an event, you will create a static DomainEvents class that is called by the domain entities from within the Model project.<br>• After you obtain the collection of IDomainEventHandlers, their handle method is called and given the domain event as an argument to action |

Wrox - Professional ASP.NET Design Patterns @September, 2010.pdf, Pag 617

## 7.4 Resource Access Layer / Data Access Layer

### 7.4.1 Repository / Data Access Object

Responsabilités :

- allow to retrieve and persist your business entities
- typically includes all the create, read, update, and delete (CRUD) methods, transaction management, data concurrency, as well as a querying mechanism to enable your business logic layer to retrieve objects for any given criteria.
- should not contain business logic and should be accessed via the business logic layer through interfaces; this adheres to the separation of concerns principle and ensures that the business layer remains unaware of the underlying data access implementation strategy
- Inherits BaseDao

ORMs (Object Relational Mapper) :

- Entity Framework
- NHibernate

Helpers

- Enterprise Library Data Access Application Block

Naming Conventions :

- suffix with "Dao"
- examples :
    - UserDao
    - FormDao

Recommendations:
If you have methods that return a particular type of business entity, place these methods in the Data Access Logic Component for that type. For example, if you are retrieving all orders for a customer, implement that function in the Order Data Access Logic Component because your return value is of the type Order. Conversely, if you are retrieving all customers that have ordered a specific product, implement that function in the Customer Data Access Logic Component.

Apress - Pro Entity Framework 4.0 @March 2010
Wrox - Professional ASP.NET Design Patterns @September, 2010.pdf

### 7.4.2 Query Object

Query :
- an object that represent a database query

Responsabilities :
- keep
    - all filter criteria
    - order by
    - top

Implementation :
- LINQ
- http://litequery.codeplex.com/

When to use ?
- use to make a query over the database and return a List<Entity>

Naming conventions :
- suffix with "Query"
- examples:
    - PolicyQuery
    - FormQuery

http://martinfowler.com/eaaCatalog/queryObject.html
http://lostechies.com/chadmyers/2008/08/02/query-objects-with-repository-pattern-part-2/
http://fabiomaulo.blogspot.com/2010/07/enhanced-query-object.html
http://psandler.wordpress.com/2009/11/10/draftdynamic-search-objects-part-1-introduction/

Wrox - Professional ASP.NET Design Patterns @September, 2010.pdf
Page 215

### 7.4.3 Lazy Loading

- used to defer the loading of object properties until needed

#### 7.4.4 Gateway

- An object that encapsulates access to an external system or resource.

Suffix: "Gateway"

#### 7.4.5 Service Agent

- a component acting as the front-end of communications towards Web Services. It should be solely responsible for actions of direct consumption of Web Services.

Suffix: "ServiceAgent" / "SvcAgent"

### 7.5 ALL Layers

#### 7.5.1 Layer Supertype

- defines an object that acts as the base class for all types in its layer, and is very much based around inheritance.
- acts as a common base class from all objects in the business layer providing implementation for common logic.

#### 7.5.2 Separated Interface

- defines an interface in a separate package from its implementation

#### 7.5.3 Registry

- a well-known object that other objects can use to find common objects and services.

#### 7.5.4 Plugin Factory

- links classes during configuration rather than compilation.

Addison Wesley - Patterns of Enterprise Application Architecture @November 2002.chm
Wrox - Professional ASP.NET Design Patterns @September, 2010

## 8. **Deployment Patterns**

Deployment Type :

- Non-Distributed Deployment
  - A non-distributed deployment is where all of the functionality and layers reside on a single server except for data storage functionality
- Distributed Deployment
  - A distributed deployment is where the layers of the application reside on separate physical tiers. Distributed deployment allows you to separate the layers of an application on different physical tiers.

Scaling :

- Scale up: get a bigger box.

With this approach, you add hardware such as processors, RAM, and network interface cards to
your existing servers to support increased capacity.

- Scale out: get more boxes.

To scale out, you add more servers and use load balancing and clustering solutions. In addition
to handling additional load, the scale-out scenario also protects against hardware failures. If one server fails, there are additional servers in the cluster that can take over the load.
If your application is I/O-constrained and you must support an extremely large database, you might partition your database across multiple database servers.

"Load-Balanced Elements" (aka "Load Balanced Cluster")

- a distribution pattern where multiple servers are configured to share the workload. Load balancing provides both improvements in performance by spreading the work across multiple servers, and reliability where one server can fail and the others will continue to handle the workload.

"Active–Redundant Elements" (aka "Fail-Over Cluster")

- a set of servers that are configured so that if one server becomes unavailable, another server automatically takes over for the failed server and continues processing.
- for Session Failover keep session on SQL Server, AppFabric Caching

Ms Press - Web Application Architecture Pocket Guide @2008.pdf
Wiley - Architecting Enterprise Solutions Patterns for High Capability Internet Based Systems Chapter 7
Packt - ASP.NET 4 Social Networking @2011 Chapter 11

## 9. **N-Layered Architecture**

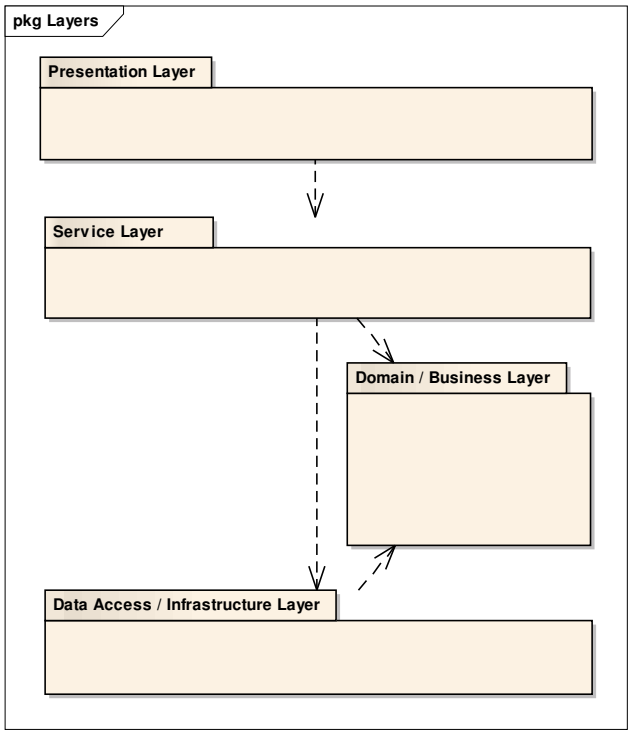
### 9.1 Networking Concepts

http://www.techiwarehouse.com/cms/engine.php?page_id=d9e99072
http://www.wirelessforums.org/how-tos/basic-network-knowledge-troubleshooting-96070.html

### 9.2 Layers

- Presentation Layer (aka User Interface Layer, View Layer or UI Layer)
- Service Layer (aka Application Layer)
- Business Layer (aka Domain Layer, Business Logic Layer or Model Layer)
- Resource Access (aka Data Access Layer, Infrastructure Layer, or Integration Layer)

http://en.wikipedia.org/wiki/Common_layers_in_an_information_system_logical_architecture




- **Core**
  - Interfaces

- **Presentation layer**
  - Views
  - Controllers\Presenters
  - ViewModels
  - Proxies (Service Agents)

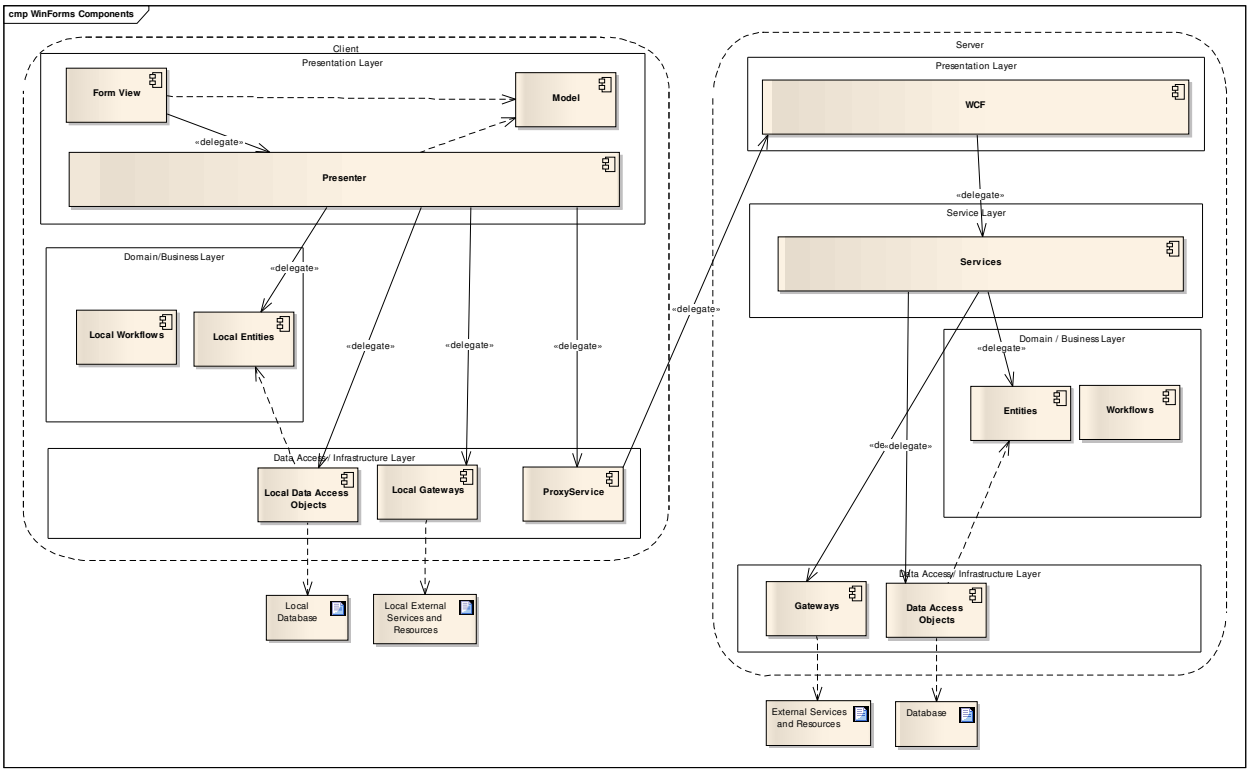
- **Service Layer**
  - Services
  - Messages (Request / Response)

- **Business / Domain Layer**
  - Entities
  - Value Objects
  - Factories
  - Assemblers
  - Mappers
  - Workflows

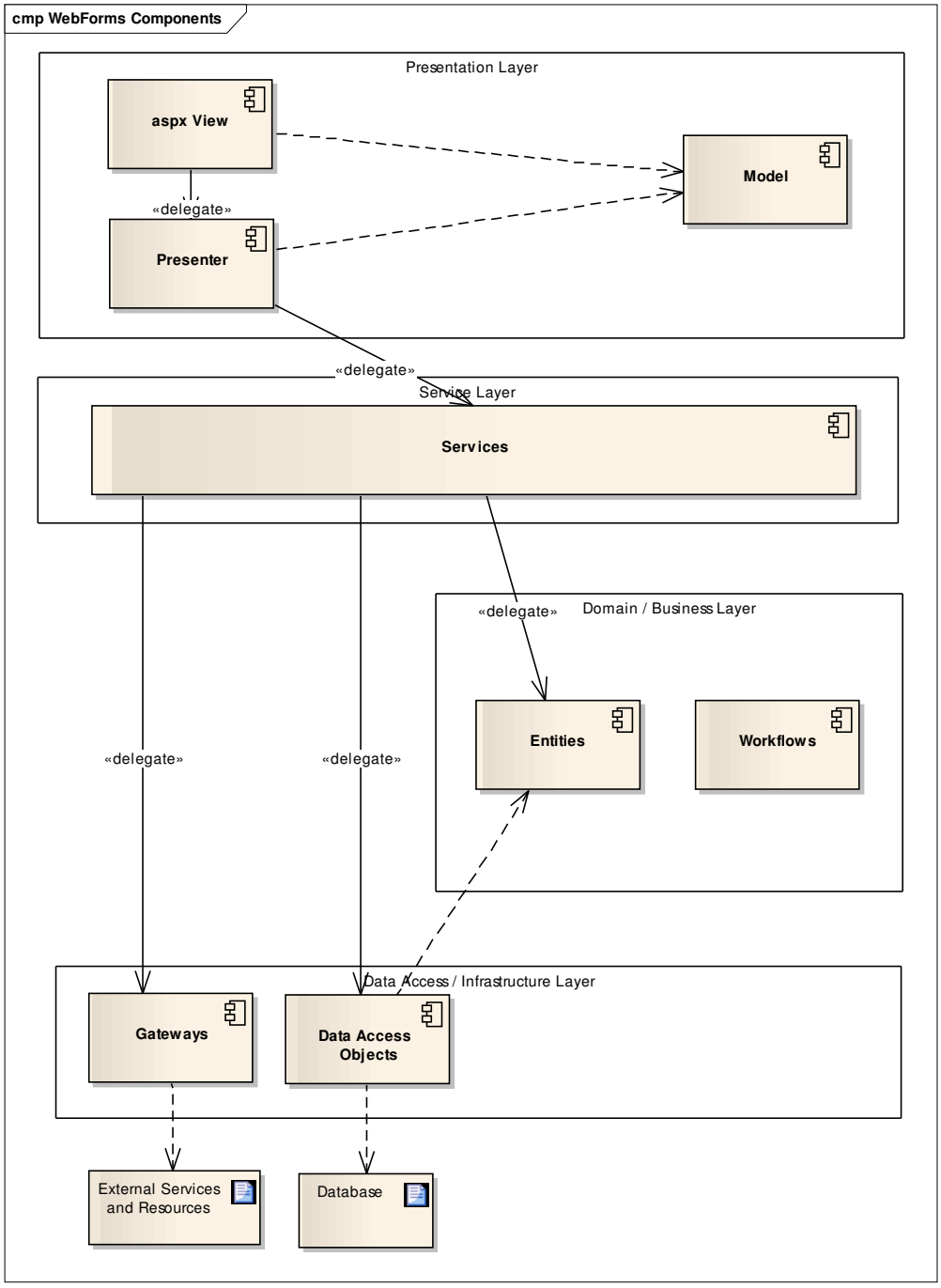
- **Resource Access Layer / Data Access Layer**
  - Repositories/Dao
  - ORM technology infrastructure
  - Gateways
  - Service Agents

- **Cross-Cutting Components**
  - DTOs for Entities objects
  - DTOs for Entities PKs
  - Queries

9.3 Client-Server Application LayeredDesign


cmp WinForms Components
Client
Presentation Layer
Form View
Model
«delegate»
Presenter
Domain/Business Layer
«delegate»
Local Workflows
Local Entities
«delegate»
«delegate»
«delegate»
Data Access/ Infrastructure Layer
Local Data Access Objects
Local Gateways
ProxyService
Local Database
Local External Services and Resources
«delegate»
Server
Presentation Layer
WCF
«delegate»
Service Layer
Services
Domain / Business Layer
«delegate»
Entities
Workflows
«de«delegate»
Data Access/ Infrastructure Layer
Gateways
Data Access Objects
External Services and Resources
Database

## 9.4 Web Application Layered Design




Ms Press - N-Layered Domain Oriented Architecture Guide with .NET 4.0 @2011

http://msdn.microsoft.com/es-es/architecture/en
Domain Oriented N-Layered .NET 4.0 Sample App

## 10. **Crosscutting Concerns**

- Logging mechanism :
    - Use ILogger interface
    - I has 2 implementations using Log4Net and Enterprise Library Logging Block

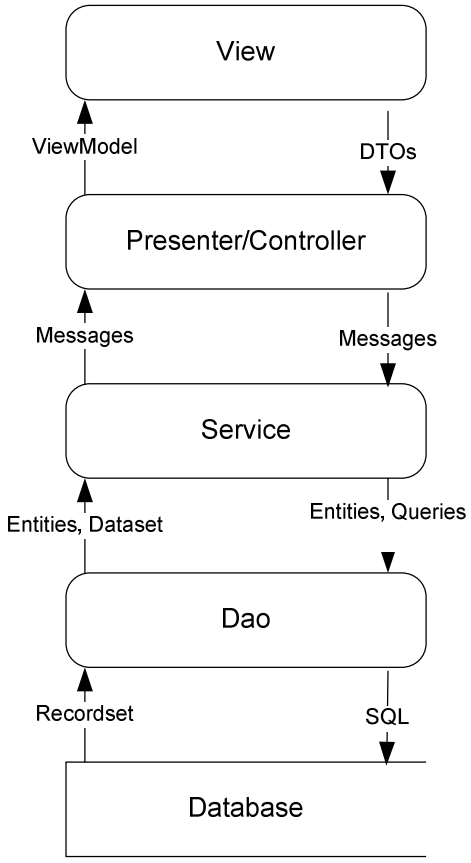
- Authentication and Authorization :
    - User Services
    - authenticated identities are pass across layers using SessionContextInfo dto class. SessionContextInfo is the first parameter for all methods in the Service Layer.

- Exception management
    - Use Exception Handling Application Block
      http://msdn.microsoft.com/en-us/library/dd140113.aspx
    - In UI Layer use try/catch for every page or control events
    - In Service Layer a single Try/Catch block surrounds the method to ensure that any exception thrown will not reveal the internal structure of the service.

http://msdn.microsoft.com/en-us/library/ff649923.aspx
Shielding Exceptions at WCF Service Boundaries

- Create error id
- Log error details + Log Context (ip, username, etc)
- Tell caller „Something went wrong" + Error ID + Contact Information

Communication between the layers:




| | |
|---|---|
| Controllers -> Views | - Use ViewModels as models for Views<br>- Use Dtos as models for Views when working with a single entity |
| Views -> Controllers | - Use Dtos when necesary as parameters for add/edit/delete/find methods |
| Controllers -> Services | - Use Request Messages:<br>o Request<DTO> to add/edit/delete an entity<br>o Request<Query> to query entities |

| | |
|---|---|
| | ○ Request<type> to find one entity by primary key |
| Services -> Controllers | - Use Response Messages :<br>○ BaseResponse to return result for add/edit/delete<br>○ Response<Dto> to return a single entity<br>○ Response<VM> to return a ViewModel<br>○ ListResponse<Dto> to return result for query |
| Services -> Dao | - Use Entity to add/edit/delete entity<br>- Use Query to query entities |
| Dao -> Services | - Use Entity / List<Entity> |
| Dao-> Database | - Query using ORM |
| Database -> Dao | - Recordset |

| | |
|---|---|
| View -> Presenters | - Use IViews to get user input in Presenters by calling Build() methods that will build DTOs. |
| Presenters - > View | - Use IViews to set result in user interface by calling Bind() methods. Pass the ViewModel to the view. |

- Caching

Typically, to cache is to take some data or objects from backend database and store them in memory for later use. When we need the data/object, we can simply retrieve it from the cache (memory).

Cache storage providers :

- Memcached : distributed memory object caching systems
  http://memcached.org/
  https://github.com/enyim/EnyimMemcached/downloads
  Membase : http://www.couchbase.com/downloads

- Windows Server AppFabric is a Microsoft server for hosting and managing WCF and Workflow Foundation (WF) services and also provides distributed caching services. Velocity was incorporated in AppFabric.
- ASP.NET's default caching mechanism (HttpContext.Current.Cache)

The caching system will be used at the Service Layer.

- Validation
  - make validation in Entity classes
  - take in consideration to use Enterprise Library Validation Application Block at the Service Layer

- Configuration Management

Settings/Parameters can be stored in web.config, xml, database, etc.
Parameters stored in database can be accesed from web app and sql. Loading changes from database will not require web app restart. Parameters stored in web.config can be accessed from web app only. Modifications to web.config will restart the web.app. If Session is kept in the same memory as web app, the Session will be restarted.

http://msdn.microsoft.com/en-us/library/ee658124.aspx
http://msdn.microsoft.com/en-us/library/ee658105.aspx

## 11. **Frontend Technologies**

### 11.1 ASP MVC

ASP.NET MVC framework will impose the site organization.

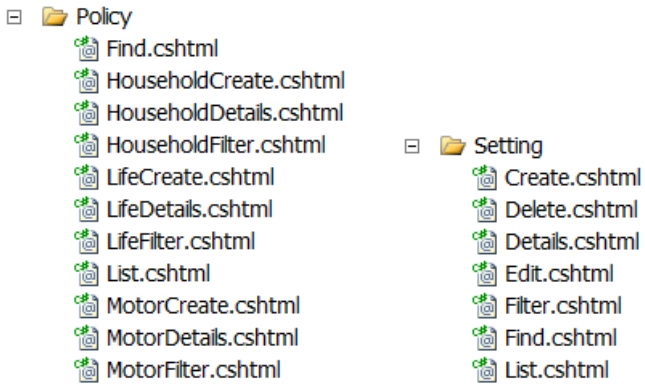
Controllers

- Inherits from BaseController

Views

- Generate Views using « Scaffold » templates
- Naming Conventions :

  Suffix views with
  - Find
  - Filter
  - List
  - Details
  - Edit
  - Create




Models

- Usually are represented by ViewModel
- Can be represented by a single DTO if the view is working with only one entity

## 11.2 ASP Web Forms

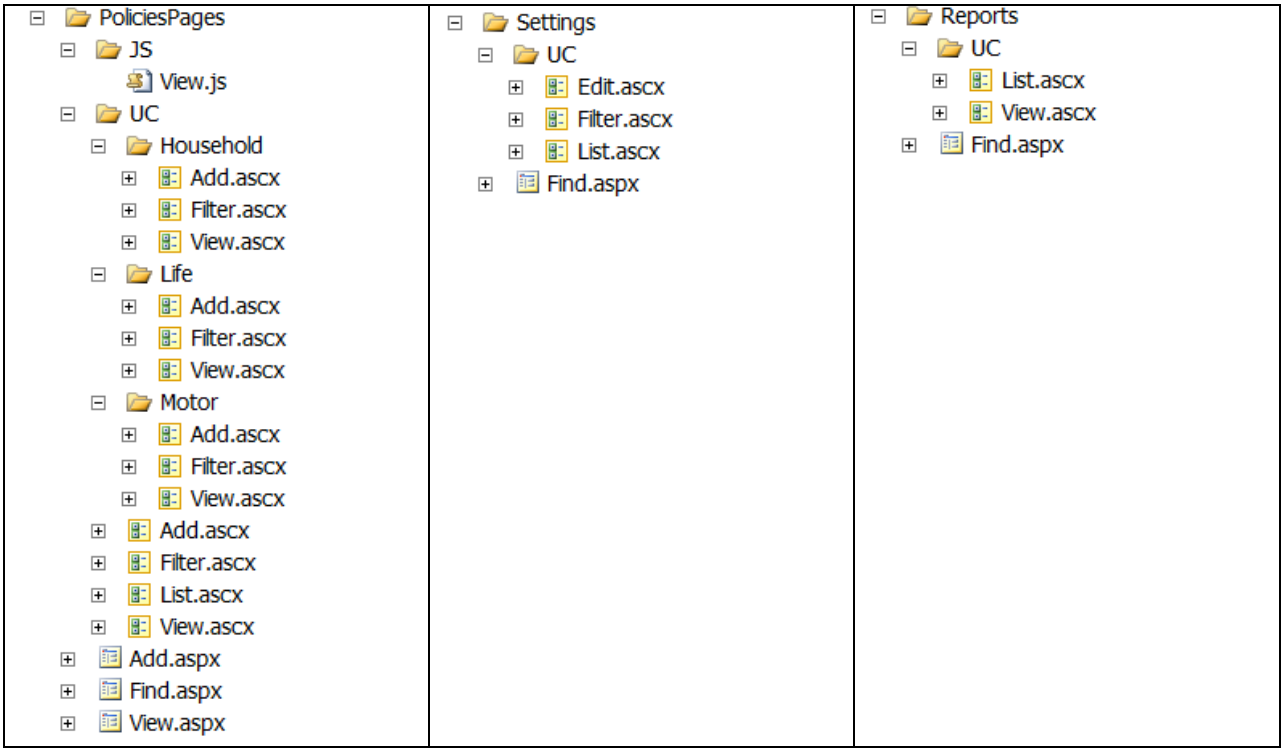
### 11.2.1 Directories and Files organisation

RootDirectory

Module1
Module2
…
ModuleN
UC
Scripts
Images
Default.aspx
Login.aspx
Index.aspx

| | |
|---|---|
| Uc | user controls common to all modules |
| Scripts | java script libraries used in all modules<br>All JavaScript methods should be included in js files. |
| Images | images common to all modules |

For every module on the interface we create one folder to keep all the files that are used only for that module.




Module

\Uc

\Images

\Js

.aspx

| | |
|---|---|
| Uc | user controls related to this module |
| Scripts | java script files related to this module |
| Images | images related only to this module |
| .aspx | .aspx pages related to this module |

Pages and User Controls Files naming conventions:

| | |
|---|---|
| Add.aspx | Controller Page to add entity |
| Find.aspx | Controller Page to find entity |
| View.aspx | Controller Page to view entity |
| Edit.aspx | View Control |
| Add.ascx | Add Control |
| Filter.ascx | Filter Control |
| List.ascx | List Control |
| View.ascx | View Control |
| Edit.ascx | Edit Control |

### 11.2.2 User Controls

Practices :

- Split pages in user controls
- Advantages :

- well defined responsabilities
- code reuse
- easy to debug (smaller parts to manage)

Types of Controls and their responsabilities :

Controller Page/Window :

- Transfers responsabilities to user controls
- Manage user control communications
- Inherits from BasePage

Filter Control :

- Builds Filter object
- Validates user input
- Inherits BaseFilterControl

List Control :

- Calls Service passing Filter as parameter and displays results in grid
- Paging
- Sort data by columns
- Export data to excel/csv
- Inherits BaseListControl

View Control :

- Displays entity information
- Calls Service to get all information for an entity
- Inherits BaseViewControl

Edit Control

- Add/Edit/Delete entity
- Calls Service for CRUD operations on entity
- Inherits BaseEditControl

http://techrepublic.com.com/5100-22-5034451.html

### 11.3 **HTML**

Visual Studio Extensions and Updates :

- Web Standards Update for Visual Studio 2010 SP1

http://blogs.msdn.com/b/webdevtools/archive/2011/06/15/web-standards-update-for-visual-studio-2010-sp1.aspx

Resources:
Pluralsight - HTML5 Fundamentals
http://www.w3schools.com/html5/default.asp

### 11.4 **CSS**

Libraires :

- Bootstrap from Twiter

http://twitter.github.com/bootstrap/

Online Tools & Services:
http://css3generator.com/
http://www.ajaxload.info/

http://jigsaw.w3.org/css-validator/

Resources :
Lynda - CSS Fundamentals
Lynda - CSS: Page Layouts
Lynda - Responsive Design Fundamentals
Sams - Teach Yourself CSS in 10 Minutes

## 11.5 **Java Script**

### 11.5.1 Folders and Files Organization

All javascript should be put in the "Scripts" folder.
Generic libraries, usually created using jQuery plugins or custom functions will be put in the "Scripts\Lib".

For every module on the interface we create one sub-folder to keep all the files that are used only for that module. The module name is the same as the controller name.
Example "Scripts\Policy"

### 11.5.2 Debugging

- IE
  - Tools\Internet Options\Advanced\Disable Script Debugging -> uncheck
    - Visual Studio 2010 Debugger
    - Web Development Helper
      http://projects.nikhilk.net/WebDevHelper
    - Tools\Developer Tools
    - IE Developer Toolbar
      http://www.microsoft.com/download/en/details.aspx?id=18359

- Mozilla Firefox
  - Tools\Error Console
  - Tools\Web Developer
  - Firebug Addon for Mozilla Firefox
    https://addons.mozilla.org/en-US/firefox/addon/firebug/
  - Web Development Bookmarklets
    https://www.squarefree.com/bookmarklets/webdevel.html
  - Tamper Data

- Chrome
  - Tools\Developer tools (WebInspector)

- Fidller 2 (use « localhost.«  in URL instead of «localhost«)
  It seems it doesn't work with Mozilla Firefox
  http://www.fiddler2.com/fiddler2/

- Alternative to using alert("")
  - Sys.Debug.trace("")in MS Ajax
    <textarea id="TraceConsoale"
  - console.log(« «) in Mozilla & Chrome

### 11.5.3 Libraries / Frameworks

- JQuery

http://jquery.com/
http://www.codylindley.com/jqueryselectors/

- JQuery Templates
  http://jtemplates.tpython.com/
- JQuery UI
  http://jqueryui.com/demos/
- JqGrid
  http://www.trirand.com/blog/?page_id=6
  http://www.webpirates.nl/webpirates/robin-van-der-knaap/47-fluent-jqgrid-html-helper-for-aspnet-mvc
- YUI
  http://developer.yahoo.com/yui/
- AngularJs

### 11.5.4 Tools

- http://www.favbrowser.com/internet-explorer-developer-tools-vs-firefox-firebug-vs-safari-web-inspector-vs-opera-dragonfly/
- http://amiworks.co.in/talk/11-best-web-developer-tools-for-firefox-and-internet-explorer/
- JSLint (Quality Tool)
  http://www.jslint.com/
- JsMin
  http://www.crockford.com/javascript/jsmin.html
- YUI Compressor

| | |
|---|---|
| JqGrid | |

Browser Statistics :
http://www.w3schools.com/browsers/browsers_stats.asp

Screen Resolution
http://www.w3schools.com/browsers/browsers_display.asp

Lynda - JavaScript Essential 2011
PluralSight - Structuring JavaScript Code
Pluralsight - jQuery Fundamentals
Pluralsight - jQuery Advanced Topics
Pluralsight - ASP.NET Ajax Advanced Topics
Pluralsight - Using The Chrome Developer Tools

Rockable - Getting Good with JavaScript @2011
OReilly - JavaScript The Definitive Guide, 6th Edition @Apr 2011
OReilly - Javascript The Good Parts @May 2008
OReilly - jQuery Pocket Reference @2010

## 11.6 **WCF**

Style:

- SOAP (Simple Object Access Protocol)
- REST (Representational State Transfer)

Message Formats :

- XML
- RSS, Atom (subsets of XML)
- JSON (usually for AJAX applications)
- Csv (usually for exporting tabular data)
- binary
-

Syndication Feed Formats: RSS 2.0, Atom 1.0

Debug WCF Services
Use Service Trace Viewer Tool
http://msdn.microsoft.com/en-us/library/ms732023.aspx

```
<system.diagnostics>
<sources>
<sourcename="System.ServiceModel"
switchValue="Information, ActivityTracing"
propagateActivity="true">
<listeners>
<addname="traceListener"
type="System.Diagnostics.XmlWriterTraceListener"
initializeData= "c:\WcfTraces.svclog" />
</listeners>
</source>
</sources>
</system.diagnostics>
```

```
<serviceDebugincludeExceptionDetailInFaults="true" />
```

Use a « Specific Port » in Visual Studio not an « Auto-Assign Port ».

## 12. Inversion of Control (IoC)

IoC container is an "assembler" that makes object coupling a run-time
IoC container is an implementation of DI principle, whose purpose is to inject services into client code without having the client code specifying the concrete implementation.

Responsabilities :

- create all instances in the project for objects defined under the GUI
- set objects life-cycle

Libraries:

- StructureMap

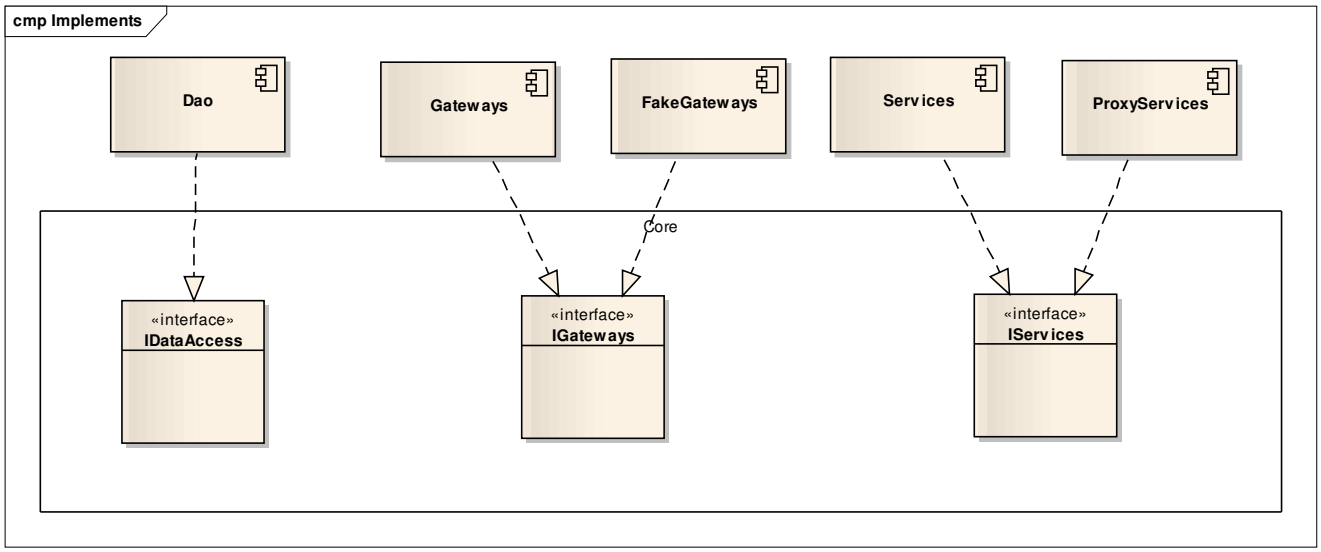
http://www.hanselman.com/blog/ListOfNETDependencyInjectionContainersIOC.aspx
http://weblogs.asp.net/shijuvarghese/archive/2008/10/10/asp-net-mvc-tip-dependency-injection-with-structuremap.aspx

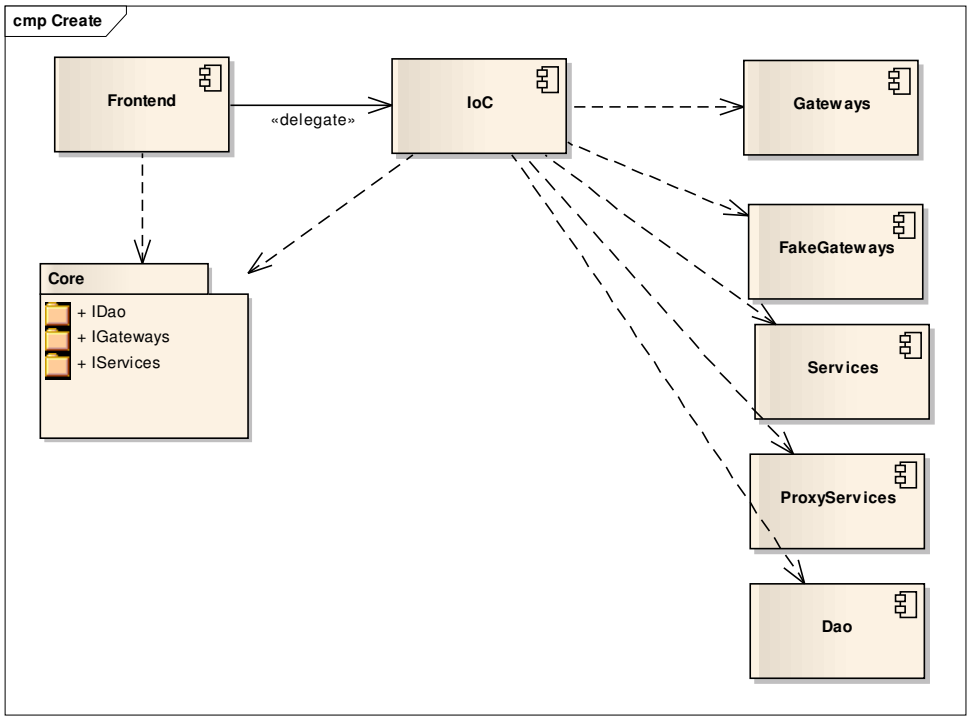
A project (ex : "Factories") is used for IoC related classes.
The Bootstrapper class is responsible for the initialization of an application. Is used in Global.asax.cs Application_Start().
Common stuff to initialize:

- current site name
- session context life-cycle : Singleton, Session, Http Request, Thread

- service call type : Direct Call / Proxy Call
- orm type : EF4 / NHibernate
- IoC

The Session context information can be kept on the IoC. Set the life-cycle depending on the UI: Session for Web, Singleton for Win, HttpRequest for WCF.
The "Registry" classes are used for registering types into the IoC
A static Facade can be build over IoC GetInstance methods. (see "Registry" pattern).
IoC / Object Creation Diagram




13. **Library**

Contains classes reusable for any other solution you may have
Example "helper" classes.

14. **Unit Tests**

Methodologies :

- Test Driven Development (TDD)
- Acceptance Test Driven Development (ATTD)
- Behaviour Driven Development (BDD)

**Practices for testability :**

- use MVC/MVP pattern in the UI layer
- abstract behind interfaces dependencies like :
  - HttpContext.Current (Session)

Automated Tests ⇔ Developer Tests
Manual Tests ⇔ Exploratory Testing

Unit Test Frameworks ca be used to write : Unit Tests or Integration Test
Unit Test Frameworks :

- MS Test
- NUNIT
- XUnit
  http://xunit.codeplex.com/
- http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks#.NET_programming_languages

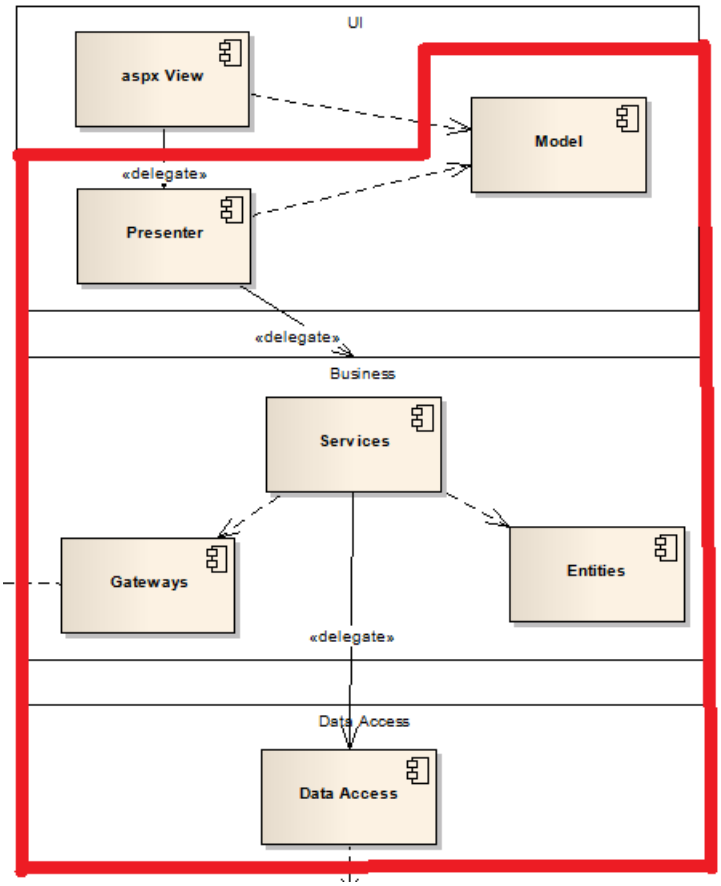
Data-Driven Unit Tests
http://msdn.microsoft.com/en-us/library/ms182527.aspx

Mocking libraries :

- Rhino Mocks
- Moq

Acceptance Tests :

- Objective : Tests should become Executable specification
- Are full integration tests
  - At UI level
  - Or Just under the UI ⇔ at the Presenter/Controller Level
- A functionality(Story) will have 1 ore more Scenario (Test Cases)




ATDD/BDD frameworks :

- SpecFlow
- StoryQ
- MSpec

Language :

- Given / When /Then (Gherkin)
- Arrange / Act / Assert

Test Results :

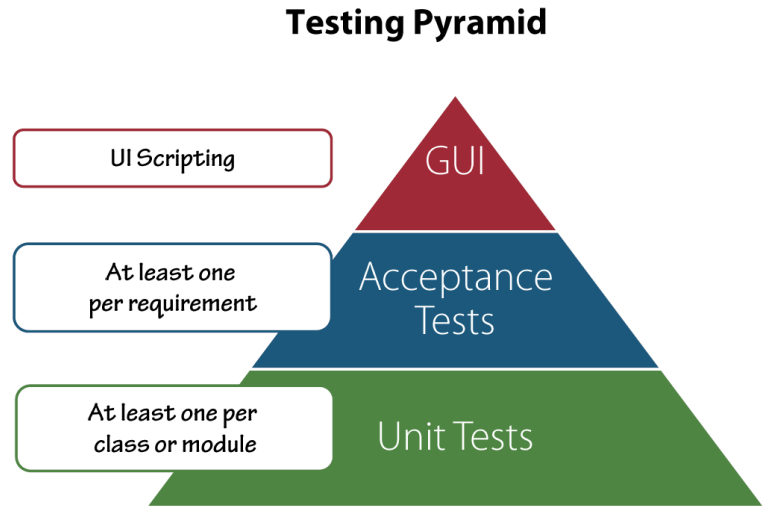
- http://trx2html.codeplex.com/
- SpecFlow.exe mstestexecutionreport Specs\Specs.csproj /testResult:TestResult.trx

UI Test / Click/Record/Playback Tools :

- Visual Studio Tools : Web Tests
- Selenium
- WatiN

http://blog.gfader.com/2010/02/smack-down-watin-vs-selenium-vs-vs2010.html

Testing Pyramid :




http://vallista.idyll.org/~grig/testing-pyramid-big.jpg

Code Coverage
Code Analysis

### 14.1 Resources

Pluralsight - Test First Development - Part 1
Pluralsight - Test First Development - Part 2

## 15. Templates / Code Generation

Use T4Templates to generate:

- Entities
  http://visualstudiogallery.msdn.microsoft.com/23df0450-5677-4926-96cc-173d02752313

- DTOs
  For Entity Framework : Generate DTO from Entities using Loef .tt templates
  http://restcake.net/Loef/
  http://loef.codeplex.com/

- Queries
- Views
- Parts from Presenters
- Parts from User Controls
- Parts from Controllers
- Parts from Views

Visual Studio Extensions:

- Tangible T4 Editor plus modeling tools for VS2010http://visualstudiogallery.msdn.microsoft.com/60297607-5fd4-4da4-97e1-3715e90c1a23?SRC=VSIDE

## 16. Performance

### 16.1 Practices

- Frontend
  - Make fewer HTTP requests.
  - Use a content delivery network (CDN)
    - Example for jQuery :
      http://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js
      http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.6.2.min.js
  - Add a far future Expires header to your components.
  - Gzip your scripts and stylesheets
  - Put your stylesheets in the document HEAD using the LINK tag.
  - Move scripts to the bottom of the page.
  - Put your JavaScript and CSS in external files.
  - Minify your JavaScript source code.

  http://developer.yahoo.com/performance/rules.html
- Memory
  - Acquire late
  - Release early
  - Use StringBuilder to concatenate strings
  - If an object implements IDisposable, be sure to call its Dispose ()
  - Reducing session state life time
  - Reducing space taken by session state
- CPU
  - Use generic List instead of DataSets
  - Returning multiple result sets
  - Sending multiple inserts in one go
  - Using native data providers
  - Do not use exceptions for anything other than trueexceptions
  - Avoid DataBinder.Eval
  - Use "for" instead of "foreach" for very high numbers of iterations
  - Avoid unnecessary processing
- Use Caching
- Minimize ViewState
- Use Compression (IIS)
- Queuing up system-to-system communications.
  - This can be done with almost any communication that does not need real-time feedback. For your queue you can use : Database, MSMQ.

Packt - ASP.NET Site Performance Secrets @2010
O Reilly - High Performance Web Sites @Sep 2007

### 16.2 Performance Testing

What we measure : Total Page Response Time (Time to Last Byte)
Load :

- User load
- Requests/Sec
- Errors/Sec

Resource :

- CPU
- Memory
- Disk / Network

Web Performance Tools :

- Firebug
- Chrome Developer Tools
- YSlow
- Chrome Speed Tracer
- IE9 Developer Tools

Load Testing Tools

- Visual Studio 2005 Team Suite (180 Days)
  http://msdn.microsoft.com/en-us/vstudio/aa718822.aspx
- Visual Studio 2008 Team Suite (90 Days)
  http://www.microsoft.com/downloads/details.aspx?FamilyID=d95598d7-aa6e-4f24-82e3-81570c5384cb&displaylang=en
- Visual Studio Team System 2008 Test Load Agent (90 Days)
  http://www.microsoft.com/download/en/details.aspx?id=8447
- Visual Studio 2010 Ultimate (30 Days, 250 Users Limit)
  http://www.microsoft.com/downloads/details.aspx?familyid=06A32B1C-80E9-41DF-BA0C-79D56CB823F7&displaylang=en
- Perfmon

Performance Test Tools :

- « Web Test » (comercial)
- « Load Test » (comercial)
- Fiddler can record requests and export them as Visual Studio .webtest

Performance Requirments

- Specific request(s) (URL)
- Response Time
- Load
- Resource Constraints

Example :

- Specific request : Search request for a random article (Search.aspx?articleName=)
- Response Time : 2s
- Load
  - 10 requests per seconds for this URL and no other requests
  - DB : 100 000 articles
- Resource Contraints :
  - Web CPU under 10%
  - DB CPU under 20%

Recommended Performance Counters :

- Processor
  - Processor \ %Processor Time (minimize, problematic over 85% consistently)
  - Process \ %Processor Time
- Memory
  - Process \ Working Set (minimize)
  - Memory \ Committed Bytes in Use (minimize)
  - Memory \ Available Mbytes (maximize)
- Disk
  - Physical Disk\ %Idle Time (higher is better)
  - Physical Disk\ % Disk Time(over 85% problematic)
- Network
  - Network Interface \ Output Queue Length (should be 0)
  - Network Interface \ Current Bandwidth
  - Network Interface \ Bytes Total/sec
- ASP.NET Applications
  - Errors Total/Sec
  - Requests / Sec
- SQLServer:SQL Statistics

- Transactions/Sec

Visual Studio 2010 Agents
http://www.microsoft.com/download/en/details.aspx?id=1334#overview
Installing

- http://stevesmithblog.com/blog/installing-visual-studio-load-test-agents-and-controllers/

Troubleshooting:

- Configure firewall on remote machines
- Start Remote Registry service on remote machines
- Add Controller user to the Performance Monitor Users group on remote machines

Pluralsight - Web Application Performance and Scalability Testing
http://vsptqrg.codeplex.com/

17. **3rd Party Libraries**

NHibernate
Loef Utility T4 templates
http://loef.codeplex.com/

Enterprise Library 5.0

- Data Access Application Block
- Exception Application Block
- Logging Application Block

http://entlib.codeplex.com/releases/view/43135

Rolling Log File :
http://bloggingabout.net/blogs/erwyn/pages/rolling-file-trace-listener.aspx

Different Configs :
http://dotnetslackers.com/ASP_NET/re-25499_External_configuration_files_in_Enterprise_Library_for_NET_Framework_2_0.aspx

Exception :
http://msdn.microsoft.com/en-us/library/cc309212.aspx

Oracle Data Access Components (ODAC)
http://www.oracle.com/technetwork/topics/dotnet/utilsoft-086879.html

Automapper
http://automapper.codeplex.com/

StructureMap
http://structuremap.net/structuremap/index.html

Memcached
http://memcached.org/
https://github.com/enyim/EnyimMemcached/downloads

App Fabric
http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=15848

jQuery
http://jquery.com/
http://jtemplates.tpython.com/

SquishIt

http://www.codethinked.com/squishit-the-friendly-aspnet-javascript-and-css-squisher

JqGrid
http://www.trirand.com/blog/?page_id=6

Fluent JqGrid Helper
http://www.webpirates.nl/webpirates/robin-van-der-knaap/47-fluent-jqgrid-html-helper-for-aspnet-mvc

Ajax Control Toolkit
http://ajaxcontroltoolkit.codeplex.com/releases/view/43475
http://www.asp.net/ajax/AjaxControlToolkit/Samples/

Rhino Mocks

Always check in the open source community before starting to develop a new component.
http://www.codeplex.com/

Links to ASP.NET Open Source Projects
http://wiki.asp.net/page.aspx/388/aspnet-open-source-projects/

18. Development Tools

UML :

- Enterprise Architect (commercial)

http://www.sparxsystems.com/

NHibernate :

- Nhibernate Visual Designer (commercial)
- Mindscape NHibernate Designer (commercial)
- NHibernate Profiler (commercial)
- NHibernate Mapping Generator

http://nmg.codeplex.com/

Distributed Systems Technologies :

- WCF
- Microsoft Message Queuing (MSMQ)
- WF
- Microsoft Distributed Transaction Coordinator

Diagnostic:

- Fiddler 2

Fiddler is a Web Debugging Proxy which logs all HTTP(S) traffic between your computer and the Internet.
http://www.fiddler2.com/Fiddler2/version.asp

- Debug View
  monitor debug output Debug.Print(« « )
  http://technet.microsoft.com/en-us/sysinternals/bb896647
  http://blog.aggregatedintelligence.com/2010/12/debugview-doesnt-work-with-aspnet-app.html
- Trace Spy
  http://tracespy.codeplex.com/
- CLR Profiler for .NET Framework 4
  http://www.microsoft.com/download/en/details.aspx?id=16273
- Reflector

Expresso
Regular expression development tool
http://www.ultrapico.com/ExpressoDownload.htm

**Visual Studio Features:**

- Code Metrics
  Analyze\Calculate Code Metrics for Solution
  http://msdn.microsoft.com/en-us/library/bb385914.aspx
- Code Analysis
- Profiler
  Analyze\Profiler
  Call Tree \ See Hot Path

Building and Deployment :

- Project\Project Build Order
- Debug\Release

Visual StudioProducts and Extensions

- NuGet

Use NuGet to get open source packages.
http://visualstudiogallery.msdn.microsoft.com/

Microsoft Web Platform Installer 3.0
http://www.microsoft.com/web/downloads/platform.aspx

Source Controls
Tools

- VSS (Visual Source Safe 2005)
- TFS

VSS Features

- Label (label a version that was deployed)
- Shared (use shared functionality to share the same user control between 2 or more projects)

19. **Continuous Integration**

Server Tools:
Free :

- Team City The Professional Edition

http://www.jetbrains.com/teamcity/download/
- CruiseControl.NET
  http://sourceforge.net/projects/ccnet/

Comercial :
- Team Foundation Build

Command Tools :
- MsBuild
- MsTest
- MsDeploy
- NCover
  www.ncover.com/
- FxCop

Build Steps:

- Compile
- Run Tests and Generate Reports
- Run Code Analysis (FxCop, Duplicates Finder)
- Create Deploy Packages
- Deploy Packages

## 20. **Coding Style**

### 20.1 Comments

Too many comments are not a good thing, because they can get in your way : they make the code longer.

Comments, especially excessive comments, should really be unnecessary if the code is truly readable. Put another way, if you need a lot of comments to make your code clear, one might rightly ask why the code is not clearer in the first place.A comment that is very old may contain out-of-date information, which can be worsethan no information at all.

Comments that tell you *what* the code is doing are an indication that the code is not clear enough: the methods are too weakly cohesive, and/or are not named expressively (named to reveal their intent).

Comments that tell you *why* the code is doing what it is doing are helpful, because they express what the code cannot: arbitrary business rules.

## 21. **Naming Conventions**

### 21.1 Definitions

*Pascal case*

The first letter in the identifier and the first letter of each subsequent concatenated word are capitalized. ex: BackColor

*Camel case*

The first letter of an identifier is lowercase and the first letter of each subsequent concatenated word is capitalized. ex: getString

*Uppercase*

All letters in the identifier are capitalized. Use this convention only for identifiers that consist of two or fewer letters. ex: System.UI

*Hungarian notation*

Identifiers are named with a standardized prefix that makes it easy to identify the type of object, and they follow the CamelCase. ex: btnValidate

### 21.2 .NET (C#) Types

| *Identifier capitalization conventions* | | | |
|---|---|---|---|
| **Identifier** | **Case** | **Example** | **Description** |
| Class | Pascal | **AppDomain** | |
| Enum type | Pascal | **ErrorLevel** | |
| Enum values | Pascal | **FatalError** | |
| Event | Pascal | **ValueChange** | |
| Exception class | Pascal | **WebException** | Always ends with the suffix **Exception**. |
| Read-only Static field | Pascal | **RedValue** | |
| Interface | Pascal | **IDisposable** | Always begins with the prefix **I**. |
| Method | Pascal | **ToString** | |
| Namespace | Pascal | **System.Drawing** | |
| Parameter | Camel | **typeName** | |
| Property | Pascal | **BackColor** | |
| Protected instance field | Camel | **redValue** | Rarely used. A property is preferable to using a protected instance field. |
| Public instance field | Pascal | **RedValue** | Rarely used. A property is preferable to using a public instance field. |

### 21.3 .NET (C#) Database Access Objects

For database objects DO NOT use hungarian notation.

| **Object type** | **New style** |
|---|---|
| SqlConnection | connection |
| SqlCommand | command |
| SqlDataAdapter | dataAdapter |
| SqlParameter | parameter |
| SqlDataReader | reader |
| OleDbConnection | connection |
| OleDbCommand | command |
| OleDbDataAdapter | dataAdapter |
| OleDbDataReader | reader |
| OleDbParameter | parameter |
| SQL String | sqlString |

### 21.4 ASP.NET UI Controls

For Web and HTML Controls use hungarian notation.

| **Control type** | **Prefix** | **Example** |
|---|---|---|

| | | |
|---|---|---|
| AdRotator | art | artCustomer |
| Button | btn | btnCustomer |
| Calendar | cld | cldCustomer |
| CheckBox | chk | chkCustomer |
| CheckBoxList | chkL | chkLCustomer |
| ComboBox | cbo | cboCustomer |
| CrystalReportViewer | crv | crvCustomer |
| DataGrid | grd | grdCustomer |
| DataList | dl | dlCustomer |
| DropDownList | ddl | ddlCustomer |
| Form | frm | |
| Frame | fra | |
| GroupBox | grp | grpCustomer |
| GridView | grd | grdCustomers |
| HyperLink | hyp | hypCustomer |
| Image | img | imgCustomer |
| ImageButton | ibtn | ibtnCustomer |
| Label | lbl | lblCustomer |
| LinkButton | lbtn | lbtnCustomer |
| LinkLabel | llbl | llblCustomer |
| ListBox | lst | lstCustomer |
| Literal | lit | litCustomer |
| OptionButton | opt | optCustomer |
| Panel | pnl | pnlCustomer |
| Picture | pic | picCustomer |
| PlaceHolder | plh | plhCustomer |
| ProgressBar | prg | prgCustomer |
| RadioButton | rbtn | rbtnCustomer |
| RadioButtonList | rbtnl | rbtnlCustomer |
| Repeater | rep | repCustomer |
| Table | tbl | tblCustomer |
| TableCell | td | tdCustomer |
| TableRow | tr | trCustomer |
| TabControl | tab | tabCustomer |
| TextBox | txt | txtCustomer |
| RequiredFieldValidator | rfv | rfvCustomer |
| CompareValidator | cmv | cmvCustomer |

| | | |
|---|---|---|
| RangeValidator | rgv | rgvCustomer |
| RegularExpressionValidator | rev | revCustomer |
| CustomValidator | ctv | ctvCustomer |
| ValidationSummary | vsm | vsmCustomer |

### 21.5 HTML

- tag names must be in lower-case
- all elements must be closed
- attribute names must be in lower case
- attribute values must be quoted

### 21.6 Java Script

- Use Hungarian notation
- Always use descriptive, relevant variable names. Make sure that the variable name gives an indication of the variables purpose.

| Subtype | Description | Prefix | Example |
|---|---|---|---|
| | | | |
| | | | |
| Boolean | Contains either True or False. | bln | blnIsUSCititzen |
| Integer | Contains an integer in the range –32,768 to 32,767. | int | intNumberOfDirectReports |
| Double | Contains a double-precision floating-point number in the range -1.79769313486232E308 to –4.94065645841247E–324 for negative values; 4.94065645841247E–324 to 1.79769313486232E308 for positive values. | dbl | dblMeritPayMulitplier |
| String | Contains a variable-length string. Strings can be made up of any alphanumeric characters. | str | strUserLastName |
| Object | Contains an object reference. An object variable represents an Automation object. | obj | objExcelSpreadsheet |

## 22. Bibliography

Books :
Wrox - Professional ASP.NET Design Patterns, @September 2010
Wrox - Professional Enterprise .NET, @2009
Addison Wesley - Design Patterns Explained A New Perspective on Object-Oriented Design, @2004
Addison Wesley - Emergent Design @2008
Addison Wesley - Patterns of Enterprise Application Architecture @November 2002
Prentice Hall - Core J2EE Patterns - Best Practices And Design Strategies @2003
Ms Press - Application Architecture Guide, 2nd Edition @2009

http://msdn.microsoft.com/en-us/library/ee658093.aspx
http://msdn.microsoft.com/en-us/library/ff650706.aspx
Ms Press - Web Application Architecture Pocket Guide @2008.pdf
Packt Publishing - ASP.NET 3.5 Application Architecture and Design @2008
Ms Press - Designing Data Tier Components and Passing Data Through Tiers 1.0 @2002
Addison Wesley - Essential Skills for the Agile Developer: A Guide to Better Programming and Design @2011
Packt - ASP.NET 4 Social Networking @2011

Video Training :
Pluralsight - Principles of Object Oriented Design
Pluralsight - Test First Development - Part 1
Pluralsight - Test First Development - Part 2
Pluralsight - HTML5 Fundamentals
Pluralsight - jQuery Fundamentals
Pluralsight - jQuery Advanced Topics
PluralSight - Structuring JavaScript Code
Pluralsight - Design Patterns Library

VTC - Design Patterns

Articles :
http://www.globalnerdy.com/2009/07/15/the-solid-principles-explained-with-motivational-posters/
http://www.martinfowler.com/eaaCatalog/
http://martinfowler.com/articles/injection.html

Study Projects :

Wrox ASP.NET Design Patterns - ASP.NET MVC 2 Case Study Starter Kit
http://aspnetdesignpatterns.codeplex.com/releases/view/52918

Wrox Professional Enterprise .NET Case Study
http://proent.codeplex.com/

Wrox NHibernate with ASP.NET Problem-Design-Solution Case Study
http://nhibernateasp.codeplex.com/

ProDinner - ASP.NET MVC EF4.1 Code First SOLID N-Tier Arch
http://prodinner.codeplex.com/

nopCommerce
http://www.nopcommerce.com/downloads.aspx

Podcasts :
http://www.asp.net/learn/podcasts
http://www.dotnetrocks.com/archives.aspx
http://www.hanselminutes.com/
http://www.runasradio.com/
http://feeds.feedburner.com/pluralcast
http://www.pluralsight-training.net/community/blogs/pluralcast/default.aspx
http://feeds2.feedburner.com/PolymorphicPodcast

http://feeds.feedburner.com/TalkingShopDownUnder
http://feeds.feedburner.com/se-radio
http://feeds.feedburner.com/deepfriedbytes

Screencasts :
http://www.dnrtv.com/
http://www.dimecasts.net/
http://msdn.microsoft.com/en-us/bb629407.aspx/

Blogs
http://www.elbandit.co.uk/blog/ Scott Millett
http://www.pluralsight-training.net/community/blogs/starr/default.aspx David Starr