­

A Research Paper on

Framework Development

What we will see

In this paper we will discuss the major points and issues related to the design of an application framework. This paper will cover this problem by NOT discussing the details of implementation, but will instead focus on the abstract design and the issues involved with that design. This paper emphasizes on conceptually part of framework , not on implementation part.

What is framework?

Building up a Framework Steps

* Establishing Requirements
* Determining the Scope of the framework
* Abstract Design
* Identifying the Design Pattern
* Designing Class Hierarchy
* Making a Framework Flexible

Qualities of Good Framework

What is Framework ?

* A set of cooperating classes that makes up a reusable design for a specific class of software. A framework provides architectural guidance by partitioning the design into abstract classes and defining their responsibilities and collaborations.
* A framework has certain responsibilities in the functioning of an application. These responsibilities are;
* Menu Management
* Form Management
* Security Management
* Communication Management
* Data Access Management

**Build up a Framework**

**Establishing Requirements**

* First step is identifying the problem you want to solve.
* If you set out to solve the wrong problem you will, most assuredly, end up with a solution that is wrong for your needs.
* **Trying to fit an application into a framework** that is working against you is not a pleasant task.
* Regardless to whether you are building your own framework or evaluating one of the many commercial frameworks available, you **need to know your requirements** clearly if you want any chance of being successful.

What kind of applications do you build?

What kind of data sources do you require?

What kind of user interface do you use?

Above questions helps to identify the requirement.

**What Kind Of Applications Do You Build?**

* Since the framework supplies the skeleton for your application, the type of application will influence the way the framework should be structured.
* **Specify(Identify) the application Domain**
* Ex. accounting, decision support, line of business, research, marketing, scheduling, and others.
* Each of these application types would require a different type of interaction with the user.
* Each type of application would have its own major requirements on the framework and the desired behavior for one may be a detriment to another.
* If you are building a decision support application the requirement for managing data saving and editing versus viewing modes for forms may just get in your way.

**What Kind Of Data Sources Do You Require?**

* You may be using one or more of the following data sources in your applications
* local DBFs
* local views
* remote views
* Internet/Intranet data sources.

**What Kind Of User Interface Do You Use?**

* Your applications may present themselves to your users in one of a variety of ways.
* There are process centric, data centric, and goal centric approaches to this presentation.

**Determining the Scope**

* **A framework is NOT an application wizard!**
* You need to determine for what the framework is responsible and for what the developer is responsible.
* Don’t use or build a framework that makes the developer work harder to get the result they want.

**Abstract Design**

Above example showing the abstract design of login framework. Abstract design will help to give basic idea of the proposed framework. This is Blueprint or Skelton of framework.

NOTE:

* A framework does NOT provide the functionality for the application, it provides the skeleton upon which that functionality is built. The application functionality is the responsibility of the developer who uses the framework.

**Qualities of a Good Framework:**

**Simplicity**

* This simplicity is achieved by giving the parts of the framework clear and consistent interfaces.
* All objects of the same lineage should have the same interface.
* The overall structure of the framework must be easy to understand. A well designed framework can be taught to a new developer in a few days at most. Obviously, the details of all of the methods and properties involved are not taught in a few days, but the new developer should be able to understand how the thing works very quickly and then fill in the details as they use it.

**Clarity**

* The behavioral aspects of the framework must be encapsulated.
* If the developer must understand the actual code in the framework then there is a definite design problem in that framework.
* The public interface for the classes in the framework should be as simple as possible.

**Boundaries**

* A framework has clear and succinct responsibilities
* it should meet those requirements and nothing more.
* All functionality outside the frameworks boundaries should be handled by the developer.
* When a framework crosses the boundary it becomes overly complex .
* A framework does NOT provide the functionality for the application, it provides the skeleton upon which that functionality is built. The application functionality is the responsibility of the developer who uses the framework and the framework should not get in their way.

**Expandability**

* It should be easy for the developer to expand the framework by either adding new classes or by subclassing the existing classes.
* A framework whose behavior is not easily modified restricts the developer rather than empowering them.
* The only sure way that a framework can provide expandability is by including it in the design

**Hooks**

* One way to provide for expandability is by providing the developer with methods in which they can write code that will affect the behavior of the framework.

**Designing Class Hierarchy**

* The design of class hierarchies is beyond the scope of this paper, but some discussion is necessary.
* In order for a framework to be simple to use and to provide for flexibility in the use of the framework the class definitions should meet at least certain basic design goals.
* Every branch in the class tree should start with a single abstract class definition that I will call the root class.
* This root class should have no code in it at all, but rather should be the class where we add the properties and methods that the subclasses will use to provide the code.
* This approach provides the structure for having a clear and consistent public interface to the classes in our class libraries.