[ASPeKT]

a lightweight AOP foundation

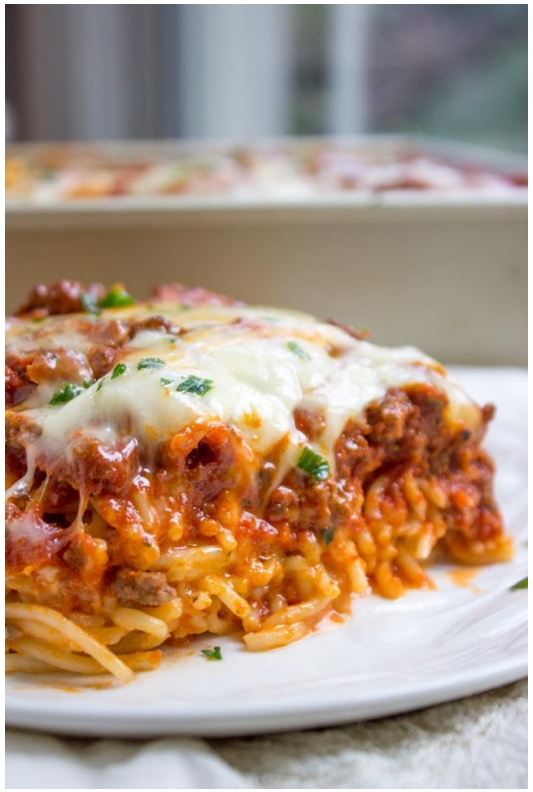
Overview

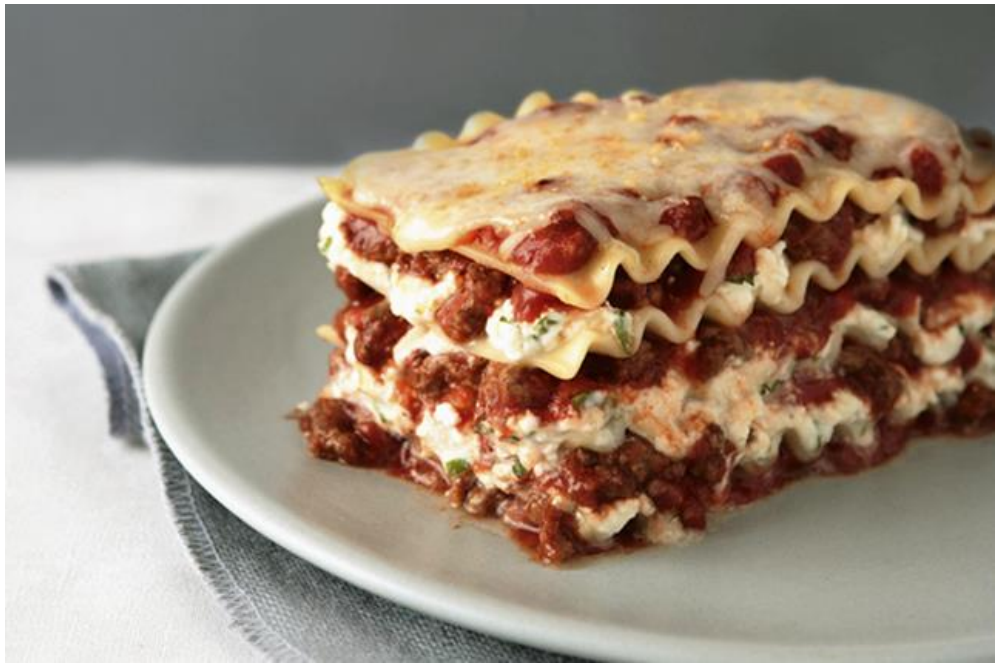
- A (quick) look at me
- A (quick) look at AOP
- A (quick) look at [ASPeKT]
- A case study
- Questions?

About Me

- Peter Lorimer
- Almost 30!
- Software scientist
- Outdoor idealist
- Tacoma owner

What is AOP?





Benefits of AOP

- Modularize functionality
- Declutter code
- Reduce signal-to-noise ratio

Components of AOP

- Cross Cutting Concerns
  - Logging
  - Auditing
  - Code contracts
  - Error handling
- Advice
- Join Point
- Point-cut
- Aspect

An AOP Model

- As per Wikipedia a Join Point Model is defined by 3 things
  1. When the Advice can run
  2. A way to specify Join Points
  3. A means of specifying code to run at the Join Point

Approaches to AOP

- Source processing i.e. Source weaving
- Runtime interpretation i.e. Run-time weaving
- Post processing i.e. Compile-time weaving
- Combination

A background on [ASPeKT]

- Light weight
- Basic
- MIT Licensed

[ASPeKT] Aspect Foundation

12 lines (10 sloc) | 305 Bytes

```
using System;

namespace Aspekt
{
    public abstract class Aspect : System.Attribute
    {
        public virtual void OnEntry(MethodArguments args) { }
        public virtual void OnExit(MethodArguments args) { }
        public virtual void OnException(MethodArguments args, Exception e) { }
    }
}
```

Creating an Aspect

```
class ConsoleLogged : Aspect
{
        public override void OnEntry(MethodArguments args)
        {
            Console.WriteLine("Entering " + args.FormattedName);
        }

        public override void OnExit(MethodArguments args)
        {
            Console.WriteLine("Exiting " + args.FormattedName);
        }
}
```

Using an Aspect

```
class Program
{
    [ConsoleLogAspect]
     static void LogMe(int a, string b, bool c)
     {
         Console.WriteLine("I was logged");
     }

    static void Main(string[] args)
    {
        LogMe(10,"Hello World", true);
        Console.ReadKey();
    }
}
```

C:\workspace\AspektPres\AspektPres\bin\Debug\AspektPres.exe

```
Entering System.Void AspektPres.Program::LogMe(10,Hello World,True)
I was logged
Exiting System.Void AspektPres.Program::LogMe(10,Hello World,True)
```

A case study

```
static class AccountManager
{

    static List<Account> Accounts { get; } = new List<Account>();

    public static void CreateAccount(string ownerName)
    {
        // No null checks
        // No authentication
        // No logging
        Accounts.Add(new Account(ownerName));
    }
}
```

Cont'd

```
static class AccountManager
{
    static List<Account> Accounts { get; } = new List<Account>();

    public static void CreateAccount(string ownerName)
    {
        if (ownerName == null)
            throw new ArgumentNullException("ownerName");
        Console.WriteLine($"Entered CreateAccount({ownerName})");

        if (!AuthorizationManager.IsAuthorized(WindowsIdentity.GetCurrent(),
Operation.Create | Operation.Account))
            throw new NotAuthorizedException("CreateAccount");

        Accounts.Add(new Account(ownerName));

        Console.WriteLine("Exiting CreateAccount()");
    }
}
```

Use Aspekt!

```
using Aspekt.Contracts; // For Null Checks
class IsAuthorized : Aspect
{
    Operation Action { get; }
    public IsAuthorized(Operation a)
    {
        Action = a;
    }

    public override void OnEntry(MethodArguments args)
    {
        if (!AuthorizationManager.IsAuthorized(WindowsIdentity.GetCurrent(), Action))
          throw new NotAuthorizedException(args.MethodName);
    }

}

class ConsoleLogged : Aspect
{
    public override void OnEntry(MethodArguments args)
    {
        Console.WriteLine("Entered " + args.FormattedName);
    }

    public override void OnExit(MethodArguments args)
    {
        Console.WriteLine("Exiting " + args.FormattedName);
    }
}
```

Cont’d

```
static class AccountManager
{

    static List<Account> Accounts { get; } = new List<Account>();

    [RequiresArgument("ownerName", typeof(string), Constraint.NotNull)]
    [ConsoleLogged]
    [IsAuthorized(Operation.Create | Operation.Account)]
    public static void CreateAccountAspect(string ownerName)
    {
        Accounts.Add(new Account(ownerName));
    }

}
```

Questions?

Thanks for listening.