Service Locator

Design Patterns

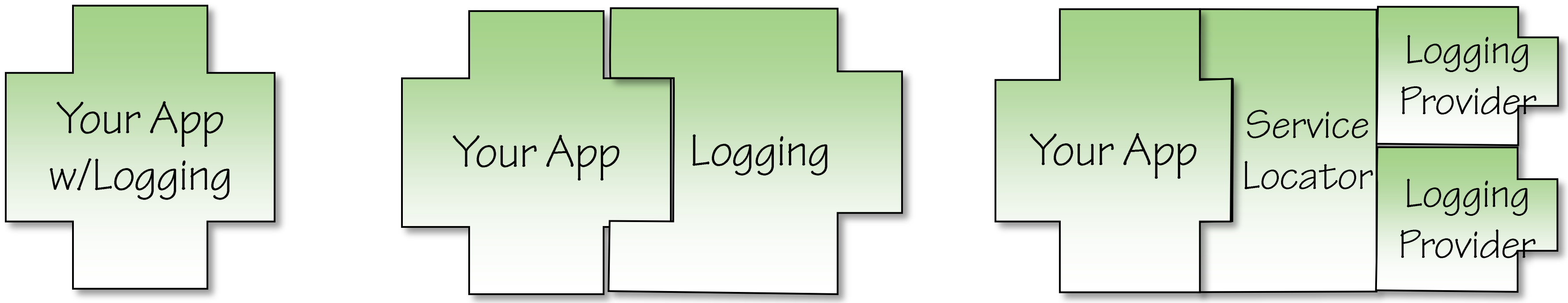
Creational Pattern

Service Locator

- **Gang of Four**
  - Abstract Factory
  - Builder
  - Factory Method
  - Prototype
  - Singleton
- **Core J2EE – Business Tier**
  - Session Façade
  - Service Locator
  - Value List Handler
- **Inversion of Control Containers**
  - Spring .NET
  - Structure Map
  - Ninject
  - Castle Windsor
- **Implementation Method**
  - Service Locator
  - Dependency Injection
  - Factory Pattern

Motivating Example

- **Logging**
  - Used throughout the application
  - Independent of any particular business function
  - Easy to implement for each application for simple needs
  - Often is changed based on environment or deployment needs



pluralsight
see what you can learn

Intent

Service Locator

- Abstract the application from the services it uses
- Play the Middleman
- Change the service without recompilation
- Identify the service through configuration
- Non-Programmers

Real World Examples

Service Locator

Operator

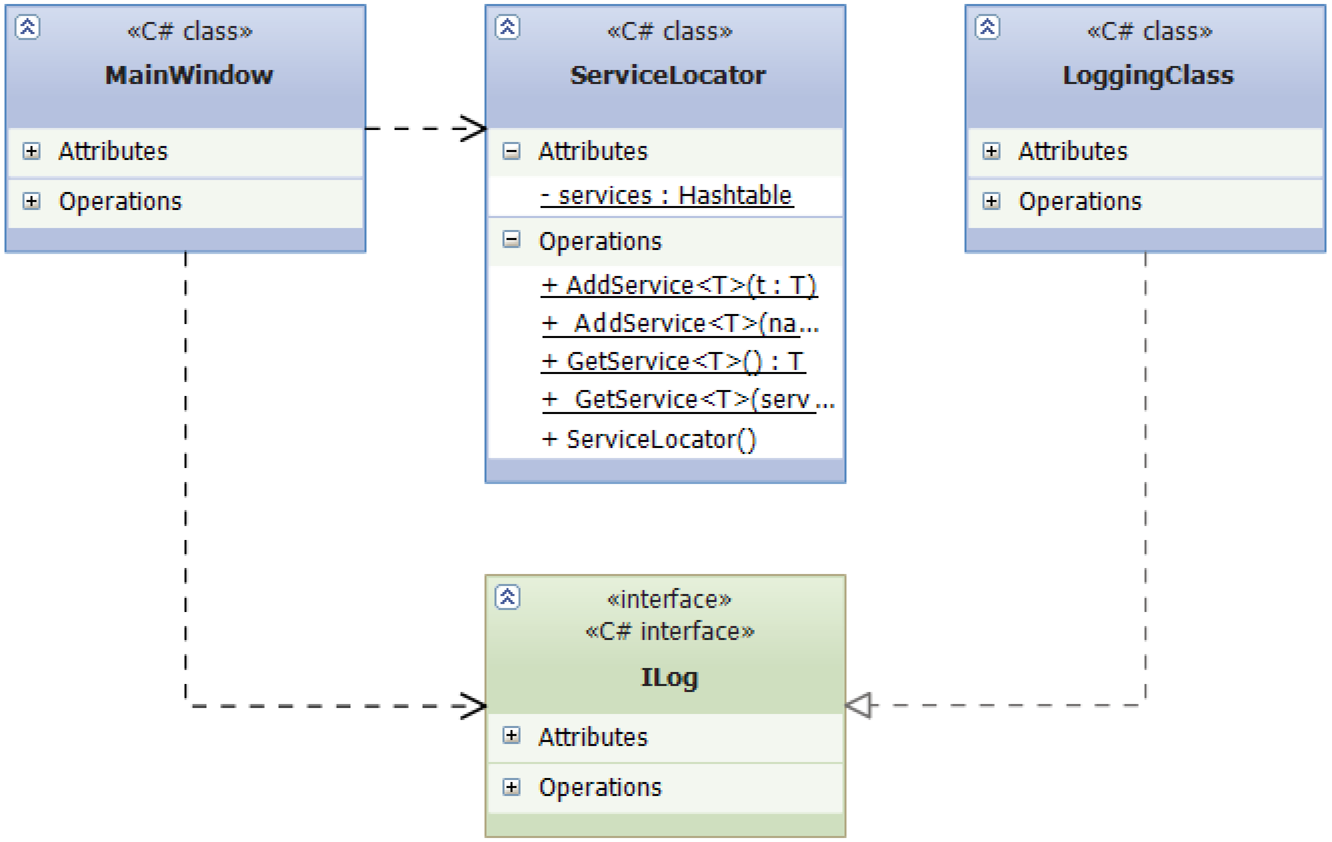
- Ask for Information
- Given a name, receive a telephone #
- Places a call

Domain Name Service

- Replace the hosts file
- Each producer independently registers
- Consumer trusts the DNS to resolve the request

Structure

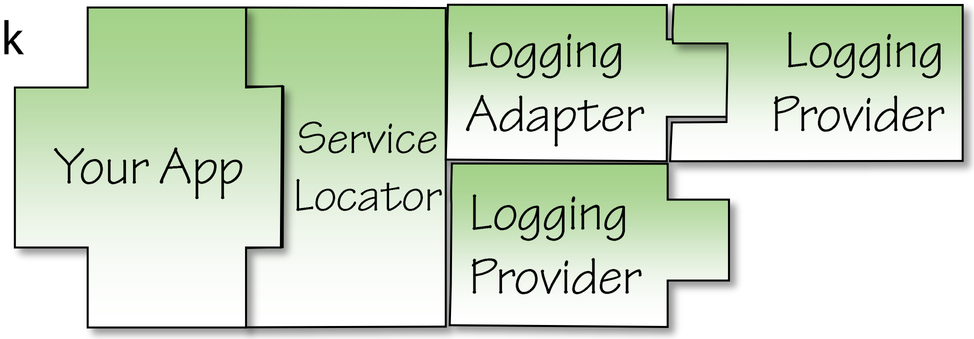
Service Locator



Other Providers

- **Why do we want other providers?**
  - Leverage the work done by other groups
  - Concentrate on our application and let other specialize in logging
  - Choose a service provider that is specific to the deployed environment

- **Samples**
  - Log4net
  - NLOG
  - Enterprise Application Block
  - …



Benefits

Service Locator

SOLID Principles

Single Responsibility	• Each component can concentrate on the specific service
Open / Closed	• Modifications to the consumer is not necessary when the producer changes
Liskov Substitution	• Services can be reduced to their interfaces
Interface Segregation	• Each interface can be specific to the type of service
Dependency Inversion	• Consumer is only dependent on the service interfaces and the service locator, never the service

Consequences

Service Locator

Global	• Any client can access
Availability	• Service may not be loaded
Failure	• Caller must know how to handle failure
Lifecycle	• Does not handle inherently the object lifecycle

Related Patterns

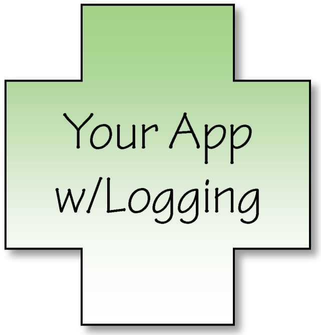
Service Locator

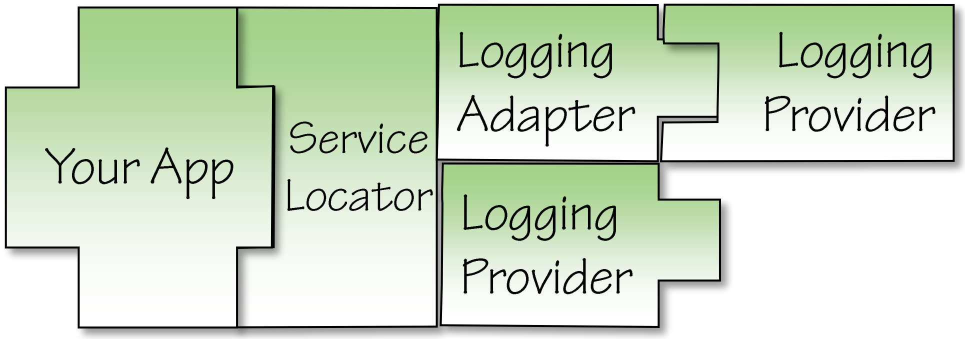
Observer Design Pattern	• Locate new services when they become available
Proxy Design Pattern	• Will often use a service locator to match proxies to implementations
Adapter Design Pattern	• Used to adapt services to a common interface
Dependency Injection	• Pass in the services when we create the clients

Summary

- **Shown how**
  - Inject a service locator between the application and the logging services
  - Use adapter pattern to allow an external library to be recognized

- **Other Methods**
  - IoC Container - Uses many design patterns to achieve robust object creation
  - Managed Extensibility Framework - Create and bind the services as part of the class declaration





For more in-depth online developer training visit

pluralsight
see what you can learn

on-demand content from authors you trust