

# Advanced Patterns

---



**Miguel A. Castro**

PRINCIPAL CONSULTANT

@miguelcastro67 [www.melvicorp.com](http://www.melvicorp.com)



# Overview



**Decorators**

**ASP.NET Filters**



# Decorators

## Problem Code

- You may have constructs whose code you don't control
- Existing components where changing would break applications
- Not all constructs are DI-enabled

## Examples

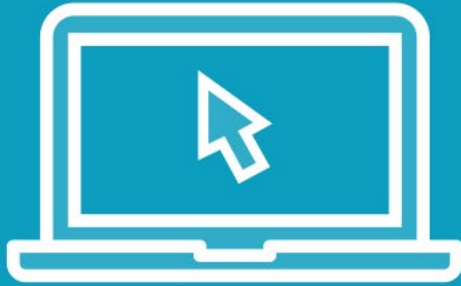
- ConfigurationManager
- Activator methods
- Static Classes
- No interface abstraction
- Locked into single implementation

## Solution

- Provide wrapper for static construct or “factory” to provide information
  - Configuration classes
  - Object instance
- Abstract factory to interface



# Demo



**Configuration Factory**

**Replacement for Activator usage**

**Registration of “others”**



# Filters

## MVC Filters

- Inherit from `System.Web.Mvc.ActionFilterAttribute`
- Override `OnActionExecuting` and/or `OnActionExecuted`
- For Autofac integration:  
    `builder.RegisterFilterProvider()`  
    Use property injection

## Web API Filters

- Inherit from `System.Web.Http.Filters.ActionFilterAttribute`
- **Override** `OnActionExecuting` and/or `OnActionExecuted`
- For Autofac integration:  
    `builder.RegisterWebApiFilterProvider()`  
    Use property injection



# Demo



Addition of logger component

MVC filter

Web API filter



# Summary



**Beyond core Autofac usage**

**Decorators can have many flavors**

**Filters are not required but useful**

**EasyBlog is done!**

