SOFTWARE GUILD

Dependency Injection

.NET Cohort

Coding Bootcamp

SOFTWARE GUILD

Lesson Goals

Become familiar with what a dependency is, why we care about injecting it, and how to set it all up!

What is a Dependency?

A dependency is something that a software component requires to do its work.

For example, let’s say we have an application that reads stock information from the interment. The class that displays the stock details to the user depends on the class that retrieves the information from the internet.

That is to say, the display class can not function without it.

What is a Dependency Injection?

Dependency Injection is a software design pattern that is intended to remove “hard-coded” dependencies in a way that allows us to swap out the classes that do the work at run-time or compile-time.

Common usages of DI are loading plugins, or injecting mock/test data in test environments.

Three Elements of DI

To get dependency injection working we need three elements:

1. A *dependent* class
2. An interface defining what the dependent class needs (a contract)
3. An injector (aka *provider* or *container*) that creates instances of classes that implement the interface for the dependent class.

The Flow

The dependent class references interfaces to describe the behavior it needs.

The injector decides which concrete classes satisfy the requirements and provides them to the dependent class.

We will start by using *factory classes* to inject our dependencies. Later in the cohort we will discuss injection frameworks such as Ninject or Microsoft Unity.

What is a Factory Class?

Like Dependency Injection, a Factory is more of a concept or Design Pattern.

Using this pattern we develop a class or method that creates objects based on some criteria.

The factory relies on inheritance and creates concrete objects to satisfy the need using either interfaces or an abstract class as a base.

DEMO : FUN WITH A FACTORY

Types of Injection

There are three types of dependency injection that we will use.

1. Constructor
2. Property
3. Method

Constructor Injection

Constructor Injection is the most common pattern used in Dependency Injection.

1. Should be used when the injected dependency is required for the class to function.
2. The injected component can be used anywhere in the class.
3. Is used when the class requires one or more dependent components.

SOFTWARE GUILD

Property Injection

1. Should be used if there are optional dependencies within the class or if a dependency may need to be swapped at runtime.
2. Would require null checking within the class to ensure that an implementation has been provided prior to use.
3. Does not require adding or modifying constructors.

Method Injection

1. The least common of the injection patterns.
2. Injects the dependency into a single method for use only in that method.
3. This is useful when the entire class does not have a dependency, but is needed in only one method.

DEMO : SAMURAI

SOFTWARE GUILD