**CST-247 Activity Guide**

Contents

[Activity 5 2](#_Toc64804482)

[Part 1: IoC using Unity 2](#_Toc64804483)

[Part 2: Application of Action Filters, IoC, and Logging 4](#_Toc64804484)

# Activity 5

This activity has multiple parts/assignments. All assignments must be completed prior to documentation submission.

## Part 1: IoC using Unity

**Overview**

In this activity, students will use Inversion of Control (IoC) container to inject a Logging Service into a Controller. They will also explore how to build a Logging Service and leverage Dependency Injection using the Unity Framework.

**Execution**

Execute this activity according to the following guidelines:

1. Add Unity Framework to Solution:
   1. Right click on Project, and select Manage NuGet Packages menu to open NuGet Package Manager.
   2. Select the Browse tab, and search for Unity.
   3. Install Unity by Microsoft package.
   4. Select the Browse tab, and search for Unity.Mvc.
   5. Install Unity.Mvc by Microsoft package.
2. Create a Logger Service:
   1. Add public specification to the ILogger Interface in the Utility namespace.
   2. Copy MyLogger1 class to MyLogger2 class from the Utility namespace.
   3. Remove static from Logger class variable.
   4. Remove static MyLogger class instance variable.
   5. Remove private constructor.
   6. Remove GetInstance() method.
   7. Update GetLogger() to use Logger class variable.
3. Constructor Injection:
   1. Register the Logging Service with Unity:
      1. Open UnityConfig file located in the App\_Start folder of your project.
      2. Add the following code to the Application\_Start() to register the Logging Service with Unity.

container.RegisterType<ILogger, MyLogger2>(new ContainerControlledLifetimeManager());

* 1. Create a Test Logging Service Controller:
     1. Add a new Controller called TestLoggingService1Controller.
     2. Create a private read-only class scoped variable called logger of type ILogger.
     3. Create a public nondefault constructor that takes an ILogger method argument.
     4. Save the constructor method argument to the private logger class variables.
     5. Change the index() method to return a string.
     6. In the implementation of the Index() call the Info() on the logger, and return a test string response.
     7. Run the application and invoke the controller.
     8. Verify that the logger statement shows up in the Log file. Take a Screenshot. Label this Screenshot 1.

1. Property Injection:
   1. Create a Test Logging Service Controller:
      1. Add a new Controller called TestLoggingService2Controller.
      2. Create a class scoped property variable called logger of type ILogger.

[Dependency]

public ILogger logger { get; set; }

* + 1. Change the Index() method to return a string.
    2. In the implementation of the Index() call the info() on the logger, and return a test string response.
    3. Run the application and invoke the controller.
    4. Verity that the logger statement shows up in the Log file. Take a Screenshot. Label this Screenshot 2.

1. Method Parameter Injection:
   1. Create a Test Service:
      1. Create a new interface class ITestService in the Business namespace.
      2. Add a method Initialize(ILogger logger).
      3. Add a method TestLogger().
      4. Create an implementation class, TestService, in the Business namespace that implements the ITestService interface.
      5. Add a private class scoped variable logger of type ILogger.
      6. Implement the Initialize():

[InjectionMethod]

public void Initialize(ILogger logger)

{

this.logger = logger;

}

* + 1. Implement the Test():

public void TestLogger()

{

logger.Info("Test Logging in TestService.TestLogger() invoked.");

}

* 1. Register the Test Service with Unity:
     1. Open UnityConfig file located in the App\_Start folder of your project.
     2. Add the following code to the Application\_Start() to register the Logging Service with Unity.

container.RegisterType<ITestService, TestService>();

container.RegisterType<ITestService>(new InjectionMethod("Initialize", new MyLogger2()));

* 1. Create a Test Logging Service Controller:
     1. Add a new Controller called TestLoggingService3Controller.
     2. Create a private read-only class scoped variable called logger of type ILogger.
     3. Create a private read-only class scoped variable called service of type ITestService.
     4. Create a public non-default constructor that takes an ILogger method argument and a ITestService method argument.
     5. Save the constructor method arguments to the private logger class variables.
     6. Change the Index() method to return a string.
     7. In the implementation of the Index() call the Info() on the logger, then call the Test() on the service, and return a test string response.
     8. Run the application, and invoke the controller.
     9. Verify that the logger statement shows up in the Log file. Take a Screenshot. Label this Screenshot 3.

**Documentation**

All documentation will be submitted at the end of the activity to the learning management system. Ensure documentation of the following:

1. Create a project report using a GCU standard project header/cover page to include a header, your name, course, assignment name, and date.
2. Add screenshots of the following activities to your project report:
   1. Screenshot 1 of the Log message from Step 3
   2. Screenshot 2 of the Log message from Step 4.
   3. Screenshot 3 of the Log message from Step 5.
3. Upload your code to GIT, and include the URL of your GIT repository in the Project Report.

## Part 2: Application of Action Filters, IoC, and Logging

**Overview**

In this activity, students will apply previous activities to their team project. They will build a Security Filter, Logging Service, and use the Unity (IoC) Framework. These components will be integrated with their team project.

**Execution**

Execute this assignment according to the following guidelines:

1. Security Filter (reference Activity 4 Part 3: Action Filters):
   1. Implement a Security Authorization Filter to handle a redirect to your login page when a user has not been authenticated—this can be done by running a check on your security token in the session, and if not valid, simply calling “filterContext.Result = new RedirectResult("/Login");” from within your Filter.
2. Logging Service (reference Activity 4 Part 1 and Activity 5 Part 1):
   1. Create a Logging Service Interface class (with debug(), info(), warn(), and error() methods).
   2. Create a Logging Service Implementation class (use NLog).
3. Register Logging Service (reference Activity 5 Part 1):
   1. Bind a singleton to the Logging Service.
4. Team Project Integration:
   1. Add the Security Filter to your team project.
   2. Add the Logging Service to your team project.
   3. Create nondefault constructor() for your Login Controller and inject the Logging Service into the Controller. Add try catch exception blocks to the Login Controller. In the exception block, use the Logging Service to log the exception message at the error log level. Note, you should repeat this for all Controllers when you submit your final team project.

**Submission**

Submit the following to the learning management system:

**Part 1**

1. Project Report that includes:
   1. A GCU standard project header/cover page to include your name, course, assignment name, and date
   2. Screenshots 1-3
   3. URL of your GIT repository
2. Upload your code to the GIT repository.

**Part 2**

1. Zip up the code for the following activities to your project report:
   1. Security Filter
   2. Logging Service
   3. Login Controller