Command Filter

English (United States) v1.6

Keywords: Command Filter, Global Command Filter, CommandFilterAttribute, Command

The Command Filter feature in SuperSocket looks like Action Filter in ASP.NET MVC, you can use it to intercept execution of Command, the Command Filter will be invoked before or after a command execution.

Command Filter class must inherit from Attribute CommandFilterAttribute:

```
 /// <summary>
/// Command filter attribute
/// </summary>
[AttributeUsage(AttributeTargets.Class, AllowMultiple = true)]
public abstract class CommandFilterAttribute : Attribute
{
    /// <summary>
    /// Gets or sets the execution order.
    /// </summary>
    /// <value>
    /// The order.
    /// </value>
    public int Order { get; set; }

    /// <summary>
    /// Called when [command executing].
    /// </summary>
    /// <param name="commandContext">The command context.</param>
    public abstract void OnCommandExecuting(CommandExecutingContext commandContext);

    /// <summary>
    /// Called when [command executed].
    /// </summary>
    /// <param name="commandContext">The command context.</param>
    public abstract void OnCommandExecuted(CommandExecutingContext commandContext);
}
```

There are two methods you should implement for your command filter:

OnCommandExecuting: This method is called before the execution of the Command;

OnCommandExecuted: This method is called after the execution of the Command;

Order: You also can set the Order property of the command filter to control the executing order

The following code defines a Command Filter LogTimeCommandFilterAttribute for recording the command execution time if the time is longer than 5 seconds:

```
public class LogTimeCommandFilter : CommandFilterAttribute
{
    public override void OnCommandExecuting(CommandExecutingContext commandContext)
    {
        commandContext.Session.Items["StartTime"] = DateTime.Now;
    }

    public override void OnCommandExecuted(CommandExecutingContext commandContext)
    {
        var session = commandContext.Session;
        var startTime = session.Items.GetValue<DateTime>("StartTime");
        var ts = DateTime.Now.Subtract(startTime);

        if (ts.TotalSeconds > 5 && session.Logger.IsInfoEnabled)
        {
            session.Logger.InfoFormat("A command '{0}' took {1} seconds!", commandContext.CurrentCommand.Name, ts.ToString());
        }
    }
}
```

And then apply the command filter to the command "QUERY" by adding attribute:

```
[LogTimeCommandFilter]
public class QUERY : StringCommandBase<TestSession>
{
    public override void ExecuteCommand(TestSession session, StringCommandInfo commandData)
    {
        //Your code
    }
}
```

If you want to apply this command filter to all commands, you should add this Command Filter Attribute to your AppServer class like the following code:

```
[LogTimeCommandFilter]
public class TestServer : AppServer<TestSession>
{

}
```

You also can cancel the command's execution by setting the commandContext's Cancel property to be true:

```
public class LoggedInValidationFilter : CommandFilterAttribute
{
    public override void OnCommandExecuting(CommandExecutingContext commandContext)
    {
        var session = commandContext.Session as MyAppSession;

        //If the session is not logged in, cancel the executing of the command
        if (!session.IsLoggedIn)
            commandContext.Cancel = true;
    }

    public override void OnCommandExecuted(CommandExecutingContext commandContext)
    {

    }
}
```

- Prev: Server Configuration Hot Update (/v1-6/en-US/Server-Configuration-Hot-Update)
- Next: Connection Filter (/v1-6/en-US/Connection-Filter)