Komme i gang med Ninject og ASP.NET Web API

1. Åpne Visual Studio 2010
2. Klikk på File -> New Project -> Visual C# -> Web og velg **ASP.NET MVC 4 Web Application**, angi et navn og trykk **OK**
3. Velg **Web API** som Project Template, trykk **OK**.
4. Nå har vi et initielt Web API project, trykker du på **F5** vil du kunne se at applikasjonen kjører.
5. Ved hjelp av Nuget kan vi nå sette opp Ninject via Package Manageren.
   Klikk på Tools -> Library Package Manager -> Package Manager Console
   Skriv inn følgende i Package Manager Console og trykk enter:

   ```
   Install-Package Ninject.MVC3
   ```

   Når den er ferdig kan man se at det har blitt gjort en del endringer i prosjektet. Det viktigste er filen **NinjectWebCommon.cs** som er lagt til under App_Start. Det er i denne filen registreringen av alle Ninject moduler må gjøres.
6. For at vi skal kunne løse avhengigheter i ASP.NET Web API ved hjelp av Ninject må man i tillegg legge til en `DependencyResolver`. Opprett en ny mappe i Solution som heter **IoC** (mappen kan hete hva som helst, dette navnet er valgt for å synliggjøre at det som skal ligge i denne mappen har med Inversion Of Control å gjøre).
7. Opprett en ny klasse i **IoC** mappen som heter `NinjectScope` og har følgende innhold:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http.Dependencies;
using Ninject.Parameters;
using Ninject.Syntax;

namespace TestNinject.IoC
{
    public class NinjectScope : IDependencyScope
    {
        protected IResolutionRoot ResolutionRoot;

        public NinjectScope(IResolutionRoot kernel)
        {
            ResolutionRoot = kernel;
        }

        public object GetService(Type serviceType)
        {
            var request = ResolutionRoot.CreateRequest(serviceType, null, new Parameter[0], true, true);
            return ResolutionRoot.Resolve(request).SingleOrDefault();
        }

        public IEnumerable<object> GetServices(Type serviceType)
        {
            var request = ResolutionRoot.CreateRequest(serviceType, null, new Parameter[0], true, true);
            return ResolutionRoot.Resolve(request).ToList();
        }

        public void Dispose()
        {
            var disposable = (IDisposable) ResolutionRoot;
            if (disposable != null) disposable.Dispose();
            ResolutionRoot = null;
        }
    }
}
```

8. Opprett en ny klasse i **IoC** mappen som heter `NinjectResolver` og har følgende innhold:

```
using System.Web.Http.Dependencies;
using Ninject;

namespace TestNinject.IoC
{
    public class NinjectResolver : NinjectScope, IDependencyResolver
    {
        private readonly IKernel _kernel;

        public NinjectResolver(IKernel kernel) : base(kernel)
        {
            _kernel = kernel;
        }

        public IDependencyScope BeginScope()
        {
            return new NinjectScope(_kernel.BeginBlock());
        }
    }
}
```

Nå har vi det vi trenger for at Ninject skal være konfigurert riktig.

9. Siste steget i konfigurasjonen er å registrere vår DependencyResolver i konfigurasjonen til applikasjonen. Det gjør man i `NinjectWebCommon` i metoden `CreateKernel` på linjen før `return kernel;` skriver man inn følgende:

```
GlobalConfiguration.Configuration.DependencyResolver = new NinjectResolver(kernel);
```

Da er konfigurasjonen av Ninject komplett.

10. For å definere bindinger mellom grensesnitt og implementasjon kan man benytte NinjectModuler. Disse modulene registreres også i `NinjectWebCommon` i metoden `RegisterServices` på følgende måte:

```
kernel.Load(new MyNinjectModule());
```

`MyNinjectModule` må arve fra `NinjectModule` og kan se f.eks slik ut dersom du har et grensesnitt `IMyInterface` som du ønsker at `MyImplementation` skal benyttes:

```
public class MyNinjectModule : NinjectModule
{
        public override void Load()
        {
            Bind<IMyInterface>().To<MyImplementation>();
        }
}
```

Nå vil Ninject sørge for at alle steder der man trenger en instans av `IMyInterface` vil implementasjonen i `MyImplementation` benyttes.

11. Lykke til!