عنوان: **مدیریت سشن‌ها در برنامه‌های وب به کمک تزریق وابستگی‌ها**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۲/۱۰ ۱۰:۴۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Design patterns, MVC, Dependency Injection

سشن‌ها در برنامه‌های وب، یکی از وابستگی‌های استاتیکی هستند که می‌توان آن‌ها را از طریق تزریق وابستگی‌ها، جهت بالا بردن قابلیت آزمون پذیری برنامه، تامین کرد. همچنین اگر از سشن‌ها برای نمونه در برنامه‌های ASP.NET MVC استفاده کنید، مقدار آن‌ها در سازنده‌ی کنترلرها نال خواهند بود ؛ از این جهت که در زمان نمونه سازی یک کنترلر توسط IoC Container، کار مدیریت سشن‌ها صورت نمی‌گیرد و اگر در این بین سرویسی نیاز به سشن داشته باشد، دیگر وهله سازی نخواهد شد؛ به این دلیل که صرفنظر از مقدار دهی متغیر سشن در صفحه‌ای دیگر، این مقدار در سازنده‌ی کلاس، نال است. در ادامه این مشکل را از طریق غنی سازی تزریق وابستگی‌ها با اطلاعات سشن جاری، برطرف خواهیم کرد.

طراحی یک تامین کننده‌ی عمومی سشن

```
public interface ISessionProvider
{
    object Get(string key);
    T Get<T>(string key) where T : class;
    void Remove(string key);
    void RemoveAll();
    void Store(string key, object value);
}
```

در اینجا یک اینترفیس عمومی را مشاهده می‌کنید که کار آن کپسوله سازی اعمال متداول کار با سشن‌ها است؛ برای مثال دریافت اطلاعات یک سشن، بر اساس کلیدی مشخص و یا ذخیره سازی اطلاعات اشیاء در سشن‌ها.
یک نمونه پیاده سازی عمومی آن نیز برای کار با سشن‌ها در برنامه‌های وب ASP.NET وب فرم و MVC، می‌تواند به صورت زیر باشد:

```
public class DefaultWebSessionProvider : ISessionProvider
{
    private readonly HttpSessionStateBase _session;

    public DefaultWebSessionProvider(HttpSessionStateBase session)
    {
        _session = session;
    }

    public object Get(string key)
    {
        return _session[key];
    }

    public T Get<T>(string key) where T : class
    {
        return _session[key] as T;
    }

    public void Remove(string key)
    {
        _session.Remove(key);
    }

    public void RemoveAll()
    {
        _session.RemoveAll();
    }

    public void Store(string key, object value)
    {
        _session[key] = value;
    }
}
```

در کلاس DefaultWebSessionProvider مستقیما از HttpContext.Current.Session برای دسترسی به سشن جاری استفاده نشده‌است. این مقدار را از سازنده‌ی خود که توسط کلاس پایه HttpSessionStateBase تامین می‌شود، دریافت خواهد کرد. این سازنده را توسط تنظیمات ابتدایی IoC Container خود وهله سازی و مقدار دهی می‌کنیم؛ زیرا HttpContext.Current.Session برای مقدار دهی، نیاز به راه اندازی یک وب سرور دارد و عملا استفاده و شبیه سازی از آن در بسیاری از آزمون‌های واحد، بسیار مشکل خواهد بود.

```
private static Container defaultContainer()
{
    return new Container(ioc =>
    {
        // session manager setup
        ioc.For<ISessionProvider>().Use<DefaultWebSessionProvider>();
        ioc.For<HttpSessionStateBase>()
           .Use(ctx => new HttpSessionStateWrapper(HttpContext.Current.Session));

        ioc.Policies.SetAllProperties(properties =>
        {
            properties.OfType<ISessionProvider>();
        });
    });
}
```

در مثال فوق یک نمونه از تنظیمات ابتدایی StructureMap را برای استفاده از مقدار HttpContext.Current.Session، جهت وهله سازی سازنده‌ی کلاس DefaultWebSessionProvider مشاهده می‌کنید.

استفاده از تامین کننده‌ی سفارشی سشن در برنامه

پس از طراحی تامین کننده‌ی سفارشی سشن و همچنین معرفی آن به IoC Container خود، اکنون استفاده‌ی از آن به سادگی ذیل است:

```
public class HomeController : Controller
{
    private readonly ISessionProvider _sessionProvider;
    public HomeController(ISessionProvider sessionProvider)
    {
        _sessionProvider = sessionProvider;
    }
```

بنابراین اگر در کلاسی، کنترلری و یا سرویسی نیاز به سشن وجود داشت، بهتر است از ISessionProvider بجای مقدار دهی و یا دسترسی مستقیم به شیء استاتیک Session استفاده کرد.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۲/۱۴ ۱۵:۲۵

یک نکته‌ی تکمیلی

شبیه به سشن، چند مورد دیگر را نیز می‌توان به همین روش تامین کرد:

```
ioc.For<IIdentity>().Use(() => HttpContext.Current.User.Identity);
ioc.For<HttpContextBase>().Use(() => new HttpContextWrapper(HttpContext.Current));
ioc.For<HttpServerUtilityBase>().Use(() => new HttpServerUtilityWrapper(HttpContext.Current.Server));
ioc.For<HttpRequestBase>().Use(ctx => ctx.GetInstance<HttpContextBase>().Request);
```