ÖZ

İÇERİK YÖNELİMLİ PROGRAMLAMA TEKNİKLERİYLE VERİTABANINDAN BAĞIMSIZ UYGULAMALAR GELİŞTİRME

Fırat HASPİŞİREN

Başkent Üniversitesi Fen Bilimleri Enstitüsü

İstatistik ve Bilgisayar Bilimleri Anabilim Dalı

Gelişen yazılım sektöründe gün geçtikçe geliştirilen uygulamaların kapsamı genişlemekte ve karmaşıklığı artmaktadır. Günümüzde uygulama geliştirilirken yeniden kullanılabilir uygulama modüllerinin geliştirilmesi gerek zaman gerekse maliyetlerin düşürülmesi açısından son derece önem kazanmıştır.

Yazılım sistemlerindeki karmaşıklık arttıkça İçerik / Bağlam Yönelimli Programlama ve Cephe Yönelimli Programlama ile geliştirilen çözümler daha popüler hale gelmiştir. Bu çözümler sayesinde yeniden kullanılabilir ve kolay yönetilebilir uygulama modülleri geliştirilebilmektedir.

Bu çalışmada bahsedilen İçerik / Bağlam Yönelimli Programlama ve Cephe Yönelimli Programlama ile geliştirilen uygulama modüllerinin avantajları ve geliştirme teknikleri, Microsoft.Net C# teknolojisi kullanılarak incelenmiş ve çeşitli modüller geliştirilmiştir.

ANAHTAR SÖZCÜKLER: İçerik, Bağlam, Cephe

Danışman: Prof. Dr. Timur KARAÇAY, Başkent Üniversitesi, İstatistik ve Bilgisayar Bilimleri Bölümü.

ABSTRACT

DEVELOPING DATABASE INDEPENDENT APPLICATIONS BY USING CONTEXT ORIENTED PROGRAMMING TECHNIQUES

Fırat HASPİŞİREN

Başkent University Graduate School of Natural and Applied Sciences

Statistics and Computer Science

Nowadays, the complexity and and the scale of the applications increases in IT sector. Therefore the Decrease in time and costs make the concept of reusability more important.

Context oriented and aspect oriented programming concepts become more popular with the increase in complexity of applications. Context oriented and aspect oriented programming concepts help to develop modules that are maintainable and reusable.

In this study, C sharp programming language and Microsoft.NET Technology was used to consider the advantages and the development techniques of all application modules that are based on Context oriented and aspect oriented programming principles.

KEYWORDS: Context, Aspect

Advisor: Prof. Dr. Timur KARAÇAY, Başkent University, Statistics and Computer Science Department.

1. GİRİŞ

İçerik / Bağlam Yönelimli Programlama (BYP) ve Cephe Yönelimli Programlama, geçmişleri 90'lı yıllara dayanmasına rağmen bu konuda geliştirilen teknolojiler son yıllarda ortaya çıkmaya başlamıştır.

Yazılım sistemlerindeki karmaşıklık arttıkça İçerik / Bağlam Yönelimli Programlama ve Cephe Yönelimli Programlama ile geliştirilen çözümler daha popüler hale gelmiştir.

Günümüz uygulamalarında yeniden kullanılabilir uygulama katmanlarının geliştirilmesi, gerek zaman, gerekse maliyetleri kısabilmek için son derece önem kazanmıştır. Bu uygulama katmanlarından en önemlilerinden bazıları:

- Veritabanı bağımsız ön bellek yönetimi (Cache Management), tez kapsamında veritabanından bağımsız bir modül olarak kullanılabilen bir önbellek modülü geliştirilmiştir
- Seyir kayıt yönetimi (Log Management)
- İstisna yönetimi (Exception Management)
- İşakış yönetimi
- Geçerlilik yönetimi

İçerik / Bağlam Yönelimli Programlama ve Cephe Yönelimli Programlama ile bahsedilen bu uygulama katmanlarının yeniden kullanılabilir modüller olarak geliştirilmesi mümkün olmaktadır. Bu modüllerin İçerik / Bağlam Yönelimli Programlama ve Cephe Yönelimli Programlama teknikleri kullanılarak sağlanan avantajlar ve kolaylıklar bu belgenin ilerleyen aşamalarında tanımlanacaktır.

Bu belge, yazılım sistemlerinde İçerik / Bağlam Yönelimli Programlama ve Cephe Yönelimli Programlama tekniklerinin nasıl kullanıldığını özellikle Cephe Yönelimli Programlama uygulamaları altında sunacak olup; üç bölümden oluşmaktadır; birinci bölümde Cephe Yönelimli Programlama nın ne olduğu ve niçin kullanıldığı tanımlanacaktır. İkinci bölümde Microsoft .Net çerçevesinde İçerik / Bağlam Yönelimli Programlama ve Cephe Yönelimli Programlamanın nasıl işlendiği

tanımlanacaktır. Üçüncü bölümde ise İçerik / Bağlam Yönelimli Programlama ve Cephe Yönelimli Programlama teknikleri kullanılarak geliştirilen yazılım yapıları tanıtılacaktır.

2. CEPHE YÖNELİMLİ PROGRAMLAMA

2.1 Microsoft.Net Programlarının Çalışma Modeli

.Net programların çalışması için üç seviyeli bir model sağlamaktadır:

1. İşlem (Process): Windows işletim sistemi için kullanılan normal çalıştırma ortamı.
2. Uygulama Alanı (Application Domain): Her işlemin bir veya birden fazla Uygulama Alanı bulunur.
    - Her uygulama alanı bir diğerinden bağımsız hata izolasyonu sağlar.
    - Uygulama Alanları arası mesajların proxy (vekil) ile iletilebilmesi için kanallara ihtiyaç duyulur.
3. Her bir uygulama alanı bir veya birden çok bağlam (context) içerir.
    - Bağlam; bir modelin örneğin güvenlik modeli, işleyiş esnasında kullandığı özellikler bütünüdür.
    - Bağlamlar arasındaki iletişim işleyişin kesilmesi ile sağlanır, örneğin bir mesajın yeni servisler eklenmesi yada düzenlenmesi için durdurulması gibi.

2.2 Cephe Yönelimli Programlamanın Amacı

Cephe sistem içersinde bir özellik olup sistemin fonksiyonel bir birimi değildir. Cephe yönelimli programlamanın temel amacı, seyir kayıt (logging) yönetimi, işlem (transaction) yönetimi, önbellek yönetimi, istisna yönetimi, zaman aşımı yönetimi gibi uygulamaya yayılmış durumda bulunan çeşitli fonksiyonel özellik ve birimleri ayrı birer modül içersinde toplamaktır.

Bu sayede uygulama kodu;

1. Daha az kod tekrarı.
2. Daha amaca yönelik iş sınıfları.
3. Daha okunabilir kod.
4. Daha güvenilir ve gürbüz kod yazımı.
5. Daha az dağınık bir kod yapısı.
6. Daha gelişmiş encapsulation (kapsülleme).

Niteliklerine sahip olur.

Cephe Yönelimli Programlama neleri kolaylaştırır;

1. Cephe lerin izolasyonu.
2. Daha okunur, daha az karışık kod.
3. Daha gelişmiş encapsulation (kapsülleme).
4. Yeniden kullanılabilir kod.

3. NET UYGULAMALARINDA CEPHE YÖNELİMLİ PROGRAMLAMA

3.1 .Net Kanalları

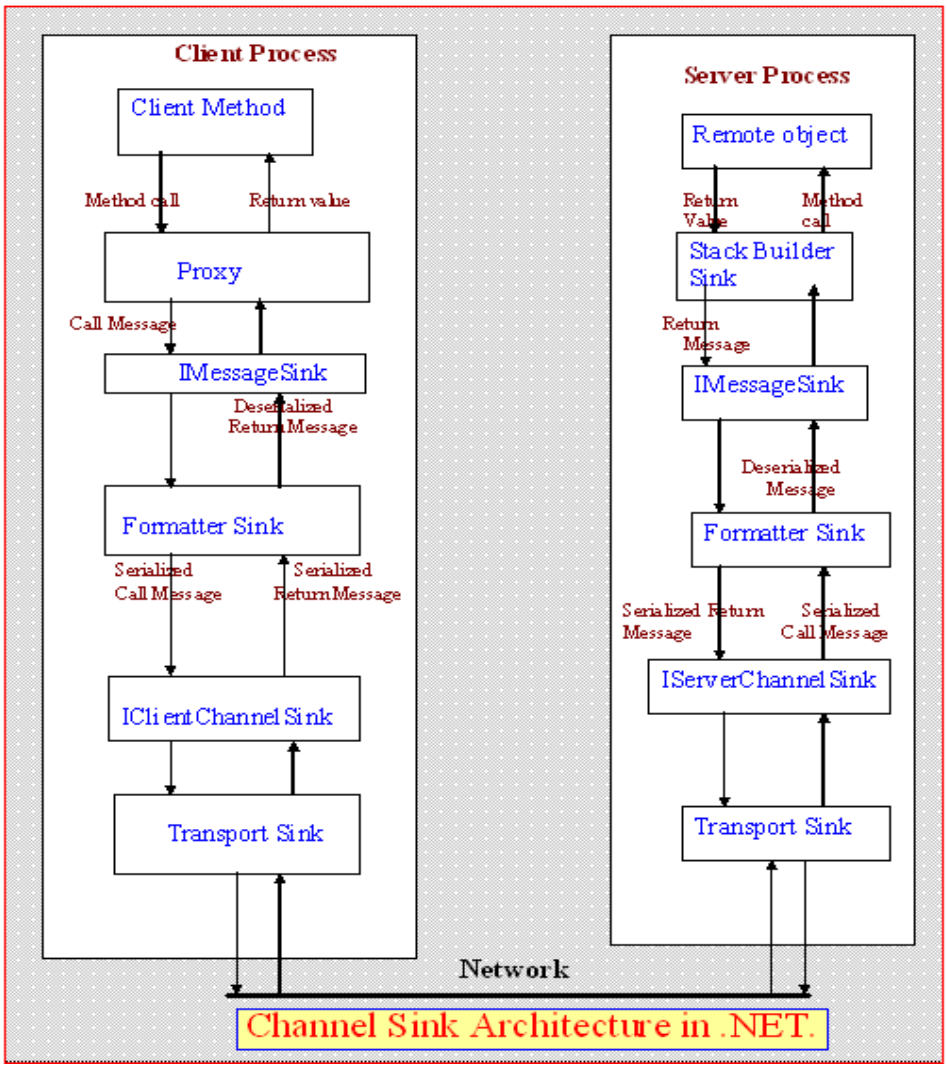
.Net uygulamalarında bağlama / içeriğe bağlı nesnelerin (ContextBoundObject) herhangi bir metodunu çağırmamız için proxy (vekil) yapısını kullanırız.

Proxy (vekil) oluşturup, bir method çağırdığımız zaman, bir mesaj nesnesi yaratılır ve bu mesaj tüm katmanları geçerek server arasında çift yönlü olarak gidip gelir.

IMessage arayüzünden referans edilen Mesaj sınıfı çağrığıyla ilgili verileri tutan bir sözlüğe sahiptir.

Remoting Platformu farklı mesajlar için birçok IMessage oluşturur. Örneğin sınıftan bir nesne oluşturulurken, Constraction Call ve Construction Response, eğer bir metot çağrılıyorsa Method Call ve Method Response gibi.

Mesajlar işlemin süreci içersinde veri transfer birimleri olarak kullanılırlar.



Şekil 1.1 .Net Kanal Mimarisi

3.2 Kanal Alıcıları (Channel Sinks)

Mesajların geçtiği iki önemli alıcı (sink) bulunmaktadır.

- Formatter Sink
- Transport Sink

Formatter Sink; mesajı dizi halinde yayınlar ve transport başlıkları yaratır. Mesajların metin, xml veya veri yığınına dönüştürlmesinde veya tersi şekilde mesaj cevaplarının metin, xml veya veri yığınından geri dönüştürülmesini sağlarlar.

Transport Sink; transport başlıklarını belirli protokol başlıklarına çevirir ve diziyi belirtilmiş protokole transfer eder.

4. GELİŞTİRİLEN UYGULAMA MODÜLLERİ

İçerik / Bağlam Yönelimli Programlama ve Cephe Yönelimli Programlama teknikleri kullanılarak geliştirilen uygulama modülleri:

1. Seyir Kayıt Yönetimi (Log Management)
2. Önbellek Yönetimi (Cache Management)
3. Zaman Aşım Yönetimi (Timeout Management)
4. Geçerlilik Yönetimi (Validation Management)
5. Diğer Geliştirilen Teknikler

4.1 Seyir Kayıt Yönetimi (Log Management)

Günümüz uygulamaların da, kullanıcılar tarafından yapılan işlemlerin kayıt altına alınması, geriye dönük işlemlerin kimler tarafından, ne zaman ve hangi parametrelerle yapıldığının bulunması önem taşımaktadır. Cephe Yönelimli Programlama ve Bağlam Yönelimli Programlama teknikleri kullanılarak Microsoft .Net çerçevesinde geliştirilen bu modul ile iş sınıfları veya bağlam içersinde çalışan sınıfların metotları çalıştırıldığı durumda, doğrudan zaman, kişi, ve parametre bilgileri ile beraber kayıt altına alınmasını sağlamaktadır. Bu modul sayesinde, kayıt altına alınması gereken metotlar içersine herhangi bir metot yazılmasına gerek kalmamıştır. Kayıt altına alınması istenmeyen metotlar yine aspect ler kullanılarak işaretlenebilirler.

4.2 Önbellek Yönetimi (Cache Management)

Gelişen web uygulamaların da, kullanıcılar tarafından yapılan işlemlerin hızlı olması önem taşımaktadır. Özellikle veritabanından okunan verilerin önbellekte tutulması ve gelecek istemlerde veritabanına gitmeden doğrudan önbellekten getirlmesi önem taşımaktadır. Cephe Yönelimli Programlama ve Bağlam Yönelimli Programlama teknikleri kullanılarak Microsoft .Net çerçevesinde geliştirilen bu modul ile iş sınıfları veya bağlam içersinde çalışan sınıfların metotları çalıştırıldığı durumda, metodun döndürdüğü cevap doğrudan önbelleğe atılmata olup, gelecek çağrımlarda eğer parametreler önceki bir çağrımın parametreleri ile aynı ise metot içi hiç çalıştırılmadan sonuç önbellekten geri döndürülür. Bu modul sayesinde, özellikle web uygulamalarında ciddi bir performans sağlanmış olup bu perofrmans için metotların içersine hiçbir kod yazılmasına gerek yoktur. Bu da programcıya hiçbir ek yük getirmeden bir önbellek yönetim sistemini kullanmasını sağlar.

4.3 Zaman Aşım Yönetimi (Timeout Management)

Çeşitli uygulamalar da, yapılan işlemlerin belirli bir zamanda bitmesi aksi durumda hata verilmesi istenebilir. Örneğin RS232 iletişim protokolü ile kontrol edilen cihazlara gönderilen komutların belirli bir sürede bitmesi istenebilir. Cephe Yönelimli Programlama ve Bağlam Yönelimli Programlama teknikleri kullanılarak Microsoft .Net çerçevesinde geliştirilen bu modul ile iş sınıfları veya bağlam içersinde çalışan sınıfların metotları çalıştırıldığı durumda, eğer metot belirli bir Timeout niteliği ile tanımlanmış ise metotun çalışma süresi denetlenir, eğer işlem Timeout niteliği ile tanımlanan sürenin üzerine çıkarsa işlem durudurulu ve bir istisna fırlatılır.

4.4 Geçerlilik Yönetimi (Validation Management)

Ugulamalar da, sınıfların gerek özelliklerine gerekse metotların aldığı parametrelerin belirli bir değer aralığında veya değerde olması istenebilir. Cephe Yönelimli Programlama ve Bağlam Yönelimli Programlama teknikleri kullanılarak Microsoft .Net çerçevesinde geliştirilen bu modul ile iş sınıfları veya bağlam içersinde çalışan sınıfların metotlarına veya özelliklerine değer atanmaya

çalışıldığı durumlarda değerin uygun kriterlere sahip olup olmadığı kontrol edilir. Eğer atanmaya çalışılan değer uygun değil ise bir hata mesajı verilir veya istisna fırlatılabilir. Bu modul sayesinde programcılar kod içersine sadece kontrol edilmesini istedikleri özellik yada parametrelerin kod yazmadan kontrol edilmesini sağlarlar.

4.5 Diğer Geliştirilen Teknikler

Yukarıdaki tekniklere ek olarak geliştirilen uygulama özellikleri:

- Metot çalıştırıma onayı; bir metot çalıştırlmaya kalktığında öncellikle kullanıcıdan onay ister.

5. GELİŞTİRİLEN UYGULAMA KODLARI

5.1 Nitelikler (Attributes)

5.1.1 Ön bellek niteliği

Niteliğin amacı iş sınıfları katmanında yada veri erişim katmanındaki tez bağlamı içinde çalışan sınıfların istenen metotlarının bu niteliği kullanılarak işaretlenmesi ve bu sayede metodun gelecek çağrımlarında sonucun doğrudan ön bellekten getirilmesidir.

TEZCacheMethodAttribute ile işaretlenmiş metot ilk çağrıldığında normal çalışacak ve sonucu döndürecektir bu sonuç çağıran nesneye döndürülmeden önce bağlam tarafından ön belleğe atılır. Devam eden çağrılarda ise, metot daha önceden çağrılmış olduğu parametre değerleri ile çağrılıyor ise, bu parametre değerleriyle ilişkili sonuç metodun çmasına izin vermeyerek sonucu doğrudan önbellekten yollayacaktır.

Kod:

```
using System;
using System.Collections.Generic;
```

```
using System.Text;

namespace Meteksan.TEZ.Library.TEZContext.Attributes
{
    [AttributeUsage(AttributeTargets.Method)]
    public class TEZCacheMethodAttribute: Attribute
    {
        private string[] _DependencyTypes = new string[] { };

        public string[] DependencyTypes
        {
            get { return _DependencyTypes; }
        }
        public TEZCacheMethodAttribute()
        {

        }
        public TEZCacheMethodAttribute(Type[] types)
        {
            _DependencyTypes = new string[types.Length];

            for (int i = 0; i < types.Length; i++)
            {
                _DependencyTypes[i] = types[i].FullName;
            }
        }
    }
}
```

Örnek Kullanım:

```
    public class OkumaGrup : TEZBusinessBase
    {
        [TEZCacheMethod]
        public OkumaGrupListesiDataTable GetirOkumaGrupListesi(string bolge_ID)
        {
            using (TableAdapterOkumaGrup adapter = new
TableAdapterOkumaGrup())
            {
                return adapter.GetirOkumaGrupListesi(bolge_ID);
            }
        }
    }
```

5.1.2 Ön belleği düşürme niteliği

Bağlam tarafından önbelleğe atılmış bilgilerin geçerliliğini yitirmesine neden olan metotların çalıştırılması durumunda bağlamın önbellekte verileri silmesini isteriz.

Niteliğin amacı iş sınıfları katmanında yada veri erişim katmanındaki tez bağlamı içinde çalışan sınıfların istenen metotlarının bu niteliği kullanılarak işaretlenmesi durumunda işaretli metot çalıştırılırsa, metodun bulunduğu sınıf ile ilişkili ve bu sınıfa bağımlı tüm önbellek nesneleri önbellekten silinir.

Kod:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Meteksan.TEZ.Library.TEZContext.Attributes
{
   [AttributeUsage(AttributeTargets.Method)]
   public class TEZDropCacheAttribute : Attribute
   {
      public TEZDropCacheAttribute()
      {
         _Dependency = string.Empty;
      }
   }
}
```

Örnek Kullanım:

```
   public class OkumaGrup : TEZBusinessBase
   {
      [TEZDropCache(DropAllCachedObjects = true)]
      public void EkleOkumaGrubu(string Grup_Adi, string bolge_ID)
      {
         using (TableAdapterOkumaGrup adapter = new
TableAdapterOkumaGrup())
         {
            adapter.EkleOkumaGrubu(Grup_Adi, bolge_ID,
TEZIdentity.CurrentUserName);
         }
```

```
    }
}
```

5.1.3 Seyir kaydı niteliği

Niteliğin amacı iş sınıfları katmanında yada veri erişim katmanındaki tez bağlamı içinde çalışan sınıfların metotlarının çalıştırıldığında bağlam tarafından doğrudan seyir kayıt sistemine aktarılmasını kontrol etmektir.

Varsayılan olarak, bağlama bağlı sınıfların herhangi bir metodu çalıştırıldığında bağlam tarafından metot çağrı bilgileri seyir kayıt altına alınır. Ancak sadece okumaya yönelik, örneğin veritabanından veri okuyup döndüren metotların seyir kayıt altına alınması istenmeyebilir. Bu durumda seyir kayıt altına alınması istenmeyen metotlar bu nitelik kullanılarak işaretlenebilir. Bağlam işaretlenmiş metot çalıştırıldığında bunun seyir kayıt altına alınmasını engeller.

Kod:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Meteksan.TEZ.Library.TEZContext.Attributes
{
    [AttributeUsage(AttributeTargets.Method)]
    public class TEZDoNotLogAttribute : Attribute
    {
        public TEZDoNotLogAttribute()
        {
        }
    }
}
```

Örnek Kullanım:

```
    public class OkumaGrup : TEZBusinessBase
    {
        [TEZDoNotLog]
        [TEZCacheMethod]
        public OkumaGrupListesiDataTable GetirOkumaGrupListesi(string bolge_ID)
        {
```

```
            using (TableAdapterOkumaGrup adapter = new
TableAdapterOkumaGrup())
            {
                return adapter.GetirOkumaGrupListesi(bolge_ID);
            }
        }
    }
```

5.1.4 Bağlam niteliği

Niteliğin amacı iş sınıfları katmanında yada veri erişim katmanındaki tez bağlamına bağımlı çalışması istenen sınıflar bu nitelik kullanılarak bağlama bağımlı çalışmaları ve metotlarınında bu bağlam içersinde çalışması sağlanır.

Kod:

```
using System;
using System.Runtime.Remoting.Contexts;
using System.Runtime.Remoting.Activation;
using Meteksan.TEZ.Library.TEZContext.Property;

namespace Meteksan.TEZ.Library.TEZContext.Attributes
{
    [AttributeUsage(AttributeTargets.Class)]
    public class TEZContextAttribute : Attribute, IContextAttribute
    {
        public TEZContextAttribute()
        {
        }

        public void GetPropertiesForNewContext(IConstructionCallMessage msg)
        {
            TEZProperty property = new TEZProperty();
            msg.ContextProperties.Add(property);
        }

        public bool IsContextOK(Context ctx, IConstructionCallMessage msg)
        {
            return false;
        }
    }
}
```

Örnek Kullanım:

```
using System;
using System.Collections.Generic;
using System.Text;
using Meteksan.TEIAS.TEZ.Library.TEZContext.Attributes;

namespace Meteksan.TEIAS.TEZ.Business
{
   [TEZContext]
   public class TEZBusinessBase : ContextBoundObject
   {

   }
}
```

5.1.5 Boş veya tanımsız geçerlilik niteliği

Niteliğin amacı iş sınıfları katmanında yada veri erişim katmanındaki tez bağlamı içinde çalışan sınıfların istenen metotlarının bu niteliği kullanılarak işaretlenmesi ve bu sayede metod çağrıldığında nitelikte ile tanımlanmış metot parametreleri boş veya tanımsız mı diye kontrolünün yapılması.

Kontrol edilen parametrelerden herhangi biri boş veya tanımsız ise bağlam metodun çalışmasına izin vermeden bir istisna fırlatır.

Kod:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Meteksan.TEZ.Library.TEZContext.Attributes
{
   [AttributeUsage(AttributeTargets.Method, AllowMultiple=true)]
   public class TEZValidateNullOrEmptyAttribute : Attribute
   {
      private string _Parameter = string.Empty;
      private string _Definition = string.Empty;
```

```
        public string Parameter
        {
            get { return _Parameter; }
        }

        public string Definition
        {
            get { return _Definition; }
        }

        public TEZValidateNullOrEmptyAttribute(string parameter, string definition)
        {
            _Parameter = parameter;
            _Definition = definition;
        }
    }
}
```

Örnek Kullanım:

```
using System;
using System.Collections.Generic;
using System.Text;
using Meteksan.TEIAS.TEZ.Library.TEZContext.Attributes;

namespace Meteksan.TEIAS.TEZ.Business
{
    public class Kullanici : TEZBusinessBase
    {
        [TEZValidateNullOrEmpty("kullanici_Adi", "Kullanıcı Adı")]
        [TEZValidateNullOrEmpty("sifre", "Şifre")]
        [TEZValidateNullOrEmpty("adi", "Adı")]
        [TEZValidateNullOrEmpty("soyadi", "Soyadı")]
        [DataObjectMethod(DataObjectMethodType.Insert, true)]
        public void EkleYeniKullanici(string kullanici_Adi, string sifre, string adi, string
soyadi,
            string aciklama, int rol_ID)
        {
            using (TableAdapterKullaniciBilgisi adapter = new
TableAdapterKullaniciBilgisi())
            {
                adapter.EkleYeniKullanici(kullanici_Adi, sifre, adi, soyadi, aciklama,
rol_ID, TEZIdentity.CurrentUserName);
            }
        }
    }
}
```

5.1.6 Aralık kontrolü niteliği

Niteliğin amacı iş sınıfları katmanında yada veri erişim katmanındaki tez bağlamı içinde çalışan sınıfların istenen özelliklerinin bu niteliği kullanılarak işaretlenmesi sonucunda özelliğe değer atanmak istendiğinde, değerin nitelik ile tanımlanmış aralık içersinde olup olmadığı kontrol edilir.

Değer istenen aralıkta değil ise bağlam özelliğe değer atanmasına izin vermez ve bir istisna fırlatır.

Kod:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Meteksan.TEZ.Library.TEZContext.Attributes
{
   [AttributeUsage(AttributeTargets.Method | AttributeTargets.Property)]
   public class TEZMinMaxAttribute : Attribute
   {
      public double Minimum;
      public double Maximum;

      public TEZMinMaxAttribute(double Min, double Max)
      {
         Minimum = Min;
         Maximum = Max;
      }
   }
}
```

Örnek Kullanım:

```
using System;
using System.Collections.Generic;
using System.Text;
using Meteksan.TEIAS.TEZ.Library.TEZContext.Attributes;

namespace Meteksan.TEIAS.TEZ.Business
```

```
{
   public class Kullanici : TEZBusinessBase
   {
         private int? _Yasi;

      public int? Yasi
      {
         get
         {
            return _Yasi;
         }
         [TEZMinMax(18, 120)]
         set
         {
            _Yasi = value;
         }
      }
   }
}
```

5.1.7 Zaman aşımı niteliği

Niteliğin amacı iş sınıfları katmanında yada veri erişim katmanındaki tez bağlamı içinde çalışan sınıfların istenen metotlarının bu niteliği kullanılarak işaretlenmesi ve bu sayede metod çağrıldığıktan sonra nitelik ile tanımlanmış zaman sınırı içersinde metodun çalışması sona ermez ise, bağlamın metodun çalıştığı thread'i durdurması ve bir istisna fırlatması.

Kod:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Meteksan.TEZ.Library.TEZContext.Attributes
{
   [AttributeUsage(AttributeTargets.Method | AttributeTargets.Property)]
   public class TEZTimeoutAttribute : Attribute
   {
      private TimeSpan _Timeout;
      private string _ExceptionMessage;

      public string ExceptionMessage
```

```
        {
            get { return _ExceptionMessage; }
            set { _ExceptionMessage = value; }
        }

        public TimeSpan Timeout
        {
            get
            {
                return _Timeout;
            }
            set
            {
                _Timeout = value;
            }
        }

        public bool Disabled = false;

        public TEZTimeoutAttribute(int timeout)
        {
            if (timeout > 0)
                _Timeout = TimeSpan.FromMilliseconds((double)timeout);
            else
                Disabled = true;
        }

        public TEZTimeoutAttribute(int timeout, string exception)
        {
            if (timeout > 0)
                _Timeout = TimeSpan.FromMilliseconds((double)timeout);
            else
                Disabled = true;
            _ExceptionMessage = exception;
        }

    }
}
```

Örnek Kullanım:

```
using System;
using System.Collections.Generic;
using System.Text;
using Meteksan.TEIAS.TEZ.Library.TEZContext.Attributes;

namespace Meteksan.TEIAS.TEZ.Business
```

```
{
   public class CoatingRoutine : TEZBusinessBase
   {
      [TEZTimeout(5 * 60 * 1000, "No Fill Gas")]
      private void LL_ATM_OFF_Routine()
      {
         using (AutoMessage message = new AutoMessage("Looping ATM Switch
Check"))
         {
            while (CoatingSystem.LL_ATM.Status !=
DidoDeviceBase.DidoStatus.ON)
            {
               if (CoatingSystem.LL_VBV.Status != DidoDeviceBase.DidoStatus.ON)
               {
                  CoatingSystem.TCU.SetTemprature = 0;
                  CoatingSystem.LL_GV.Off();
                  CoatingSystem.LL_BV.Off();
                  CoatingSystem.LL_TP_VBV.Off();
                  CoatingSystem.MultiGauge.LL_CC_EMISSION = false;
                  Helper.TransferArmCheck();
                  CoatingSystem.TGV.Off();
                  CoatingSystem.LL_TP.Stop();
                  Sleep("Load Lock Turbo Pump Slowing Down", LL_VBVOffWait1 *
OneSecond);

                  while (!(CoatingSystem.TCU.CurrentTemprature < 50))
                  {
                     Helper.ShowMessageWait("Waiting until Temprature is lower
than 50C!", LL_VBVOffWait1);
                  }

                  CoatingSystem.LL_TP_ESV.Off();
                  CoatingSystem.LL_VBV.On();
               }
               Sleep("", 2 * OneSecond);
            }
         }
      }
   }
}
```

5.2 Bağlam Mesaj Kontrolü

Server sink bağlam içersinde çalışan nesnelerin ve metotların çalıştığı süreci belirler ve kontrol eder. Bağlam içersinde kullanılan tüm nitelikler server sink

içersinde kontrol edilir ve işlenir. Niteliklerin anlamlı karşılığı olan tüm işlemler bu sınıfta yapılır.

Kod:

```
public class TEZServerSink : IMessageSink
{
    #region Özel Değişkenler
    private IMessageSink _NextSink;
    #endregion

    #region Özellikler
    public IMessageSink NextSink
    {
        get
        {
            return this._NextSink;
        }
    }
    #endregion

    #region Yapıcılar
    public TEZServerSink(IMessageSink next)
    {
        this._NextSink = next;
    }
    #endregion

    #region Process Chain
    protected virtual void BeforeProcessMessage(IMessage msg)
    {
        if (msg is IMethodMessage)
        {
            MethodBase IMethodInfo = (msg as IMethodMessage).MethodBase;

            object[] validateNullOrEmptyAttributes =
IMethodInfo.GetCustomAttributes(typeof(TEZValidateNullOrEmptyAttribute), true);
            object[] validateMinimumAttributes =
IMethodInfo.GetCustomAttributes(typeof(TEZValidateMinimumAttribute), true);
            object[] doNotLogAttributes =
IMethodInfo.GetCustomAttributes(typeof(TEZDoNotLogAttribute), true);

            if (validateNullOrEmptyAttributes.Length > 0 ||
validateMinimumAttributes.Length > 0)
                ValidateParameters(msg, validateNullOrEmptyAttributes,
validateMinimumAttributes);
```

```
            if (doNotLogAttributes.Length == 0)
                LogManager.Log(msg);
        }
    }
    protected virtual IMessage ProcessMessage(IMessage msg)
    {
        IMessage replyMessage = null;

        if (msg is IMethodMessage)
        {
            MethodBase IMethodInfo = (msg as IMethodMessage).MethodBase;

            object[] cacheAttributes =
IMethodInfo.GetCustomAttributes(typeof(TEZCacheMethodAttribute), true);
            object[] dropCacheAttributes =
IMethodInfo.GetCustomAttributes(typeof(TEZDropCacheAttribute), true);

            // drop cache
            if (dropCacheAttributes.Length > 0)
                TEZCache.SilBenzerAnahtarli(IMethodInfo.ReflectedType.FullName);

            // return value cached
            if (cacheAttributes.Length > 0)
            {
                string key = GetCacheKey(msg);

                // get return value from cache
                if (!TEZCache.VarMi(key))
                {
                    replyMessage = NextSink.SyncProcessMessage(msg);
                    if ((replyMessage as IMethodReturnMessage).Exception == null)
                        TEZCache.Ekle(key, replyMessage.Properties["__Return"],
IMethodInfo.ReflectedType.FullName, (cacheAttributes[0] as
TEZCacheMethodAttribute).DependencyTypes);
                }
                else
                    replyMessage = new ReturnMessage(TEZCache.Getir(key), new
object[0], 0, (msg as IMethodCallMessage).LogicalCallContext, msg as
IMethodCallMessage);
            }
            else
                replyMessage = NextSink.SyncProcessMessage(msg);
        }
        else
            replyMessage = NextSink.SyncProcessMessage(msg);

        return replyMessage;
```

```
        }
        protected virtual void AfterProcessMessage(IMessage msg)
        {

        }
        #endregion

        #region Helpers
        private void ValidateParameters(IMessage msg, object[]
validateNullOrEmptyAttributes, object[] validateMinimumAttributes)
        {
            Dictionary<string, string> ParametersNullOrEmpty = new Dictionary<string,
string>();
            Dictionary<string, TEZValidateMinimumAttribute> ParametersMinimum =
new Dictionary<string, TEZValidateMinimumAttribute>();

            MethodBase lMethodInfo = (msg as IMethodMessage).MethodBase;

            foreach (object objAttr in validateNullOrEmptyAttributes)
            {
                TEZValidateNullOrEmptyAttribute attribute = objAttr as
TEZValidateNullOrEmptyAttribute;
                ParametersNullOrEmpty.Add(attribute.Parameter, attribute.Definition);
            }

            foreach (object objAttr in validateMinimumAttributes)
            {
                TEZValidateMinimumAttribute attribute = objAttr as
TEZValidateMinimumAttribute;
                ParametersMinimum.Add(attribute.Parameter, attribute);
            }

            StringBuilder parameters = new StringBuilder();
            IDictionaryEnumerator propertyEnumerator =
(IDictionaryEnumerator)msg.Properties.GetEnumerator();
            string methodName = lMethodInfo.ReflectedType.FullName;

            while (propertyEnumerator.MoveNext())
            {
                object pKey = propertyEnumerator.Key;
                string pName = pKey.ToString();
                object pValue = propertyEnumerator.Value;
                ParameterInfo[] methodParameters = lMethodInfo.GetParameters();

                if (pName == "__Args")
                {
                    object[] args = (object[])pValue;
```

```
                string definition = string.Empty;
                TEZValidateMinimumAttribute minAttr;
                for (int i = 0; i < args.Length; i++)
                {
                    if
(ParametersNullOrEmpty.TryGetValue(methodParameters[i].Name, out definition))
                    {
                        if (args[i] == null)
                            throw new TEZBusinessException(definition + " boş veya
tanımsız olamaz");

                        if (args[i].GetType() == typeof(DateTime))
                        {
                            if (((DateTime)args[i]).ToString() == new
DateTime().ToString())
                                throw new TEZBusinessException(definition + " boş veya
tanımsız olamaz");
                            continue;
                        }

                        if (String.IsNullOrEmpty(args[i] as string))
                            throw new TEZBusinessException(definition + " boş veya
tanımsız olamaz");
                    }
                    if (ParametersMinimum.TryGetValue(methodParameters[i].Name,
out minAttr))
                    {
                        if (args[i] == null)
                            throw new TEZBusinessException(definition + " boş veya
tanımsız olamaz");

                        if (Convert.ToInt32(args[i]) < minAttr.Minimum)
                            throw new TEZBusinessException(minAttr.Definition + " " +
minAttr.Minimum.ToString() + " den küçük olamaz");
                    }
                }
                break;
            }
        }
    }
    private string GetCacheKey(IMessage msg)
    {
        MethodBase lMethodInfo = (msg as IMethodMessage).MethodBase;
        StringBuilder parameters = new StringBuilder();
        IDictionaryEnumerator propertyEnumerator =
(IDictionaryEnumerator)msg.Properties.GetEnumerator();
```

```
        object[] cacheAttributes =
IMethodInfo.GetCustomAttributes(typeof(TEZCacheMethodAttribute), true);
        string typeName = IMethodInfo.ReflectedType.FullName;

        //TEZCacheMethodAttribute attr = (cacheAttributes[0] as
TEZCacheMethodAttribute);
        //string typeName = attr.DependencyType == null ?
IMethodInfo.ReflectedType.FullName : attr.DependencyType.FullName;
        //parameters.Append(attr.Dependency);

        parameters.Append(typeName + "." + IMethodInfo.Name);
        parameters.Append(";");

        while (propertyEnumerator.MoveNext())
        {
            object pKey = propertyEnumerator.Key;
            string pName = pKey.ToString();
            object pValue = propertyEnumerator.Value;
            ParameterInfo[] methodParameters = IMethodInfo.GetParameters();
            if (pName == "__Args")
            {
                try
                {
                    object[] args = (object[])pValue;
                    string value = String.Empty;
                    for (int i = 0; i < args.Length; i++)
                    {
                        value = args[i].ToString();
                        if (args[i].GetType() == typeof(System.Object[]))
                        {
                            StringBuilder val = new StringBuilder();
                            val.Append("(");
                            foreach (object obj in ((object[])args[i]))
                            {
                                if (obj != null)
                                    val.Append(obj.ToString()).Append("; ");
                                else
                                    val.Append("null; ");
                            }
                            val.Append(")");

                            value = val.ToString();
                        }
                        parameters.Append((string.Format("{0}={1};",
methodParameters[i].Name, value)));
                    }
                }
```

```
                catch { }
            }
        }

        return parameters.ToString();
    }

    private void TEZCacheMethodAttribute(TEZCacheMethodAttribute
TEZCacheMethodAttribute)
    {
        throw new Exception("The method or operation is not implemented.");
    }
    #endregion

    #region IMessageSink Metotları
    public IMessage SyncProcessMessage(IMessage msg)
    {
        IMethodReturnMessage replyMessage = null;

        BeforeProcessMessage(msg);
        replyMessage = ProcessMessage(msg) as IMethodReturnMessage;

        if (replyMessage.Exception is SqlException)
            if (((replyMessage.Exception) as SqlException).Class == 16)
                throw new TEZBusinessException(replyMessage.Exception.Message,
replyMessage.Exception);

        AfterProcessMessage(msg);

        return replyMessage;
    }
    public IMessageCtrl AsyncProcessMessage(IMessage msg, IMessageSink
replySink)
    {
        return null;
    }
    #endregion
}
```

KAYNAKLAR LİSTESİ

Tam Metni Basılı Tebliğ:

[1] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin, “Aspect-Oriented Programming,” European Conference on Object-Oriented Programming (ECOOP), LNCS 1241, Springer-Verlag, Jyväskylä, Finland, June, S. 220-242, 1997

Raporlar:

[2] Wolfgang Schult ve Andreas Polze. Aspect-oriented programming with c# and .net. Technical report, Hasso-Plattner-Institute at University Potsdam, 2002

Kitaplar:

[3] Don Box ve Chris Sells. Essential .NET, Volume 1: The Common Language Runtime. Addison Wesley, 2002.

[4] Advanced .NET Remoting – Ingo Rammer; Apress; 2002

[5] Microsoft .NET Remoting - Kim Williams, James Naftel, Scott McLean; Microsoft Press; 2003

[6] Remoting with C# and .NET: Remote Objects for Distributed Applications, David Conger; John Wiley & Sons Canada, Ltd.;2003

[7] Visual C# .NET – Jason Price, Mike Gunderloy, Sybex; 2002

İnternet (www/ftp) Kaynakları:

[8] Edward GARSON. Aspect-Oriented Programming in C#/.NET. Dunstan Thomas Consulting, http://www.oberon2005.ru/paper/eg2003.pdf