Windows Azure ile Cloud Computing Uygulamaları – 6

Şirketler, yaşamlarını sürdürebilmek amacı ile çeşitli iş süreçleri gerçekleştirmektedir. İş süreçleri, sürekli ve başarılı sonuçlar üretilmesi, iş kural ve standartları ile sağlanmaktadır. Oluşturulan iş kuralları, kalitenin korunmasını amaçlamaktadır.

Günümüzde müşteri memnuniyetinin, devamlı ve istenildiği gibi sağlanması, iş uygulamaları ile gerçekleştirilmektedir. İş standart ve kurallarının farklık göstermesi, kullanıcıları özelleştirilmiş uygulamalara yöneltmektedir.

Müşteri portföylerine, uygun üretim ve hizmetlerin sunulması, iş süreçlerinin sürekli ve başarılı olması ile sağlanmaktadır. İstenilen sonuçlar, süreçleri gerçekleştiren ve izleyen uygulamaların sağlıklı çalışması ile sağlanmaktadır.

İş uygulamaları kullanımı, gerçekleştireceği iş sürecinin yoğunluğuna bağlı olarak değişmektedir. Örneğin; yılbaşı dönemini de içerisine alan yılın son üç ayı, e-ticaret ve şirketlerin muhasebe bölümleri, hesapları kapatabilmek ve ürün satmak amacı ile yoğun olarak çalışmaktadır. Gerçekleştirilen işlemler, yılın diğer zamanlarına göre daha yoğundur. İş uygulamaları, farklı zaman birimlerinde oluşan iş yoğunlukların başarı ve performans ile çalışabilmesi için elastik çalışma ortamlarına ihtiyaç duymaktadır.

Şirketler, iş süreçlerini devamlı, en az maliyet ve başarılı olarak gerçekleştirmek amacı ile uygulamalarını ***Cloud Computing*** *sağlayıcı şirketler ve özellikle* ***Microsoft Windows Azure Platform****'a taşımaktadır.*

İş uygulamalarının **Windows Azure Platform**'a taşınması ile uygulamaların, kullanabileceği zengin alt yapı sağlanmaktadır. Uygulamaların çalışma ortamın'da **Windows Azure Platform** gibi zengin bir ortama taşınması, kullanıcıları tarafında meydana gelecek olan iş yükünün düzenlenmesine ve uygulama üzerinde bulunan stresin azaltılmasını sağlayacaktır. Ama **Windows Azure Platform** 'un sağlamış olduğu alt yapı gücün amaçlandığı gibi kullanılmasını sağlamayacaktır. İş uygulamaları, **Windows Azure Platform** içerisinde en iyi başarı ve performansı yakalayabilmesi için, uygulamaların **Windows Azure Platform** 'a uyumlu olarak geliştirilmesi gerekmektedir.

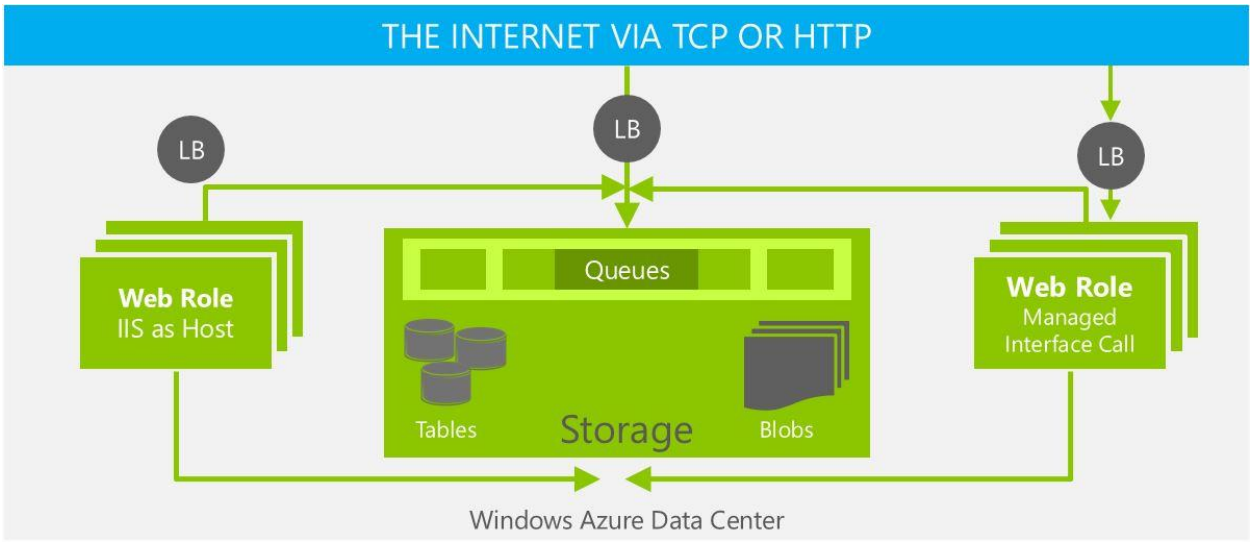
Uygulama yaşam ortamları, uygulamalar üzerinde bulunan iş yoğunluğu düzenlemesini sağlamaktadır. Kullanılan uygulamaların yazılım mimari yapısı ise, uygulamanın sağlık ve başarılı çalışmasını sağlamaktadır.

Geliştirilen iş uygulamalarının, **Windows Azure Platform** alt yapı zenginliklerini en iyi şekilde kullanılabilmesi için **Windows Azure Platform** 'un desteklemiş olduğu, **Windows Azure Cloud Service** uygulama modelinin kullanılması önerilmektedir. Uygulamalar, **Windows Azure Cloud Service** modeli ile kolay yönetilebilir, güvenli ve elastiki yaşam ortamı kazanmaktadır.

Uygulama geliştiricilerinin, Windows Azure Platform zenginleri ve Windows Azure Cloud Service uygulama modelinin hızlı ve kolay öğrenebilmesi amacı ile "WindowsAzure.FunnyApp" uygulama örneği hazırlanmıştır. Uygulama örneği ile temsili olarak iş yoğunlukların dağıtılması ve katmanlı çalışma süreçleri incelenmiştir. Uygulama örneğinde aşağıda belirtilen, senaryolar gerçekleştirilmiştir.

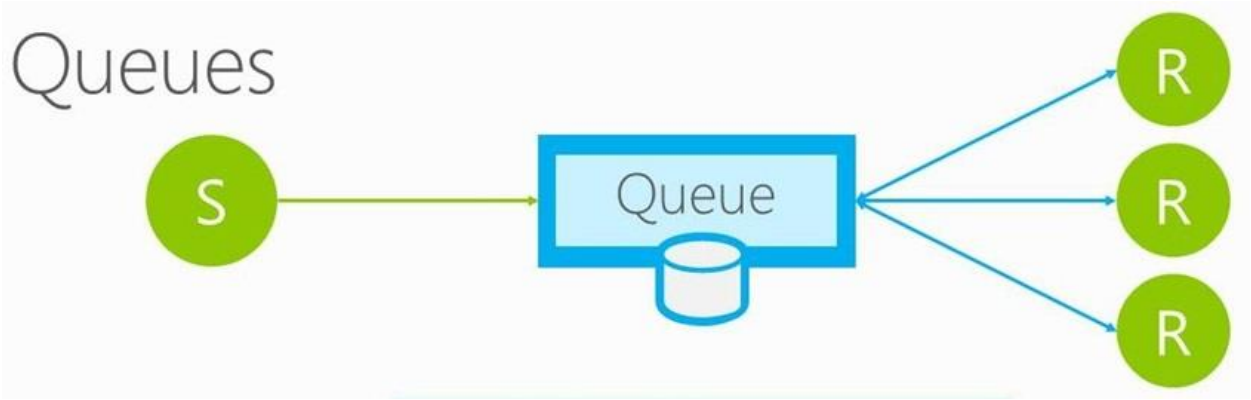
- Kullanıcı hesabının işlemleri (kullanıcı adı, parola, eposta)
- Resim yükleme alanının oluşturulması (istenen resmin yüklenmesi, açıklama, ilgili etiketler)
- Yüklenen resimlerin thumbnail boyutlarında şekillendirilmesi (Windows Azure Worker Role)
- Yüklenen resimlerin görüntülenmesi (thumbnail boyutunda resimlerin listelenmesi)
- Resminde detay gösterilmesi (açıklama, etiketler ve yüklene boyutlarda resim)
- Yorum giriş alanın oluşturulması (kullanıcı adı, eposta ve yorum)
- Yorumların listelenmesi (Kullanıcı adı ve yorum gösterilmesi)

İş uygulamaları içerisinde kullanıcı işlemlerinin gerçekleştirildiği alanlar olduğu gibi yoğun iş yükünün olduğu parçalar da bulunmaktadır. İş uygulamalarında örneğin; resim ya da video işleme, yoğun muhasebe ve arama işlemleri gerçekleştirilebilmektedir. Söz konusu işlemler, **Windows Service** uygulamaları ya da kullanıcı işlemlerinin devam ettirebileceği şekilde planlanmaktadır.



Uygulama örneği olan, ***"WindowsAzure.FunnyApp"*** çalışması içerisinde kullanıcı tarafında yüklenen resim içeriğin, içerik bilgisi alınarak işleme katmanına taşınmaktadır. Yapılan işlem ile kullanıcı çalışmalarına devam ederken, resim işleme süreçleri farklı katmanda tamamlanmaktadır. İşlem sürecinde **Windows Azure** nesneleri kullanarak, katmanlar arasında mesajlaşma işlemleri ile gerçekleşmektedir.

Windows Azure Platform içerisinde birbirinden bağımsız olan, uygulama katmanları iletişiminin sağlanması amacı ile **Queues** nesnelerinin kullanılması önerilmektedir. **Queues** nesneleri, **Windows Azure Storage** içerisinde bulunduğu gibi **Windows Azure ServiceBus Queues** yaklaşımı ile de kullanılmaktadır.



Windows Azure Platform içerisinde aynı uygulama domaininde çalışan, katmanlar arası iletişimin sağlanması amacı ile **Windows Azure Storage** içerisinde bulunan **Queues** nesnesi kullanılması önerilmektedir. Uygulama parçalarının dışı ya da iç sistemler ile iletişim sağlaması için **Windows Azure ServiceBus Queues** yaklaşımın kullanılması önerilmektedir. **Windows Azure Platform** 'un desteklemiş olduğu **Queues** nesneleri ile ilgili bilgi ve karşılaştırma aşağıda bulunmaktadır.

Özellik	Windows Azure Queues	Windows Azure Service Bus Queues
Toplu mesaj gönderme	Desteklemiyor	Destekliyor
Toplu mesaj alma	Destekliyor	Destekliyor

WCF Entegrasyonu	Desteklemiyor	Destekliyor
WF Entegrasyonu	Geliştirme Gerektiriyor	Destekliyor
Server-Side Transaction Log	Destekliyor	Desteklemiyor
Zamanlanmış Gönderim	Destekliyor	Destekliyor
En az mesaj boyuttu	64KB	256KB
En fazla mesaj boyuttu	100TB	1,2,3,4 ya da 5GB
Kuyruk Algoritması		First-In-First-Out (FIFO)
Kullanılan iletişim protokollü	HTTP/HTTPS üzerinde Rest	HTTPS üzerinde Rest
Peek fonksiyon desteği	Destekliyor	Desteklemiyor
Authentication	Symmetric key	ACS claims
Yetki tabanlı iletişim	Desteklemiyor	Destekliyor

Gerçekleştirilen uygulama örneği, dış herhangi sistem ile iletişim kurmaması nedeni ile **Windows Azure Queues** kullanılmıştır. Uygulama içerisinde mesaj içeriğinin gönderilmesi ve dinleyici tarafından alınması ile ilgili süreçler gerçekleştirilmiştir. Uygulama senaryoları, aşağıda kaynak kodlar ile incelenmektedir.

Aşağıda anlatılan süreçler de kullanılmak üzere bazı sabitler tanımlanmıştır. Tanımlanan değişkenler, yazılan kod içerisinde anahtar içeriklerin tek noktada kontrolünü sağlamaktadır. **Blob**, **Queue** nesneleri için tanımlanan değişken değerlerinin, küçük harf ile yazılmasına dikkat edilmelidir.

```
public class Utils
{
    public const string ConfigurationString = "DataConnectionString";

    public const string CloudQueueKey = "imagequeue";
    public const string CloudBlobKey = "imageblob";
}
```

Uygulama Senaryosu – I

Kullanıcı tarafında resim ve ilgili bilgilerin girilmesi işlemleri gerçekleştirilmektedir. Gerçekleştirilen işlemler ile ilgili bilgiler mesaj kuyruğuna girmesini sağlayacaktır. Konu ile ilgili kaynak kodlar, yorumlar ile aşağıda anlatılmıştır.

```
private static readonly object _look = new object();
private static bool _storageInitialized = false;

// işlem gerçekleştirlecek olan nesnelerin tanımlanması
private static CloudBlobClient _blobClient;
private static CloudQueueClient _queueClient;

protected void Page_Load(object sender, EventArgs e)
{
    this.Page.Title = "Image Uploads";
    if (IsPostBack) return;
    InitializeStorage();
}

protected void ButtonSave_Click(object sender, EventArgs e)
{
```

```
    if (FileUploadImage.HasFiles & Page.IsValid)
    {
        string uniqueBobName = string.Format("{0}/funnyimage_{1}{2}", Utils.CloudBlob
Key,
                                             Guid.NewGuid().ToString(),
                                             Path.GetExtension(FileUploadImage.FileNa
me));

        CloudBlockBlob blob = _blobClient.GetBlockBlobReference(uniqueBobName);
        blob.Properties.ContentType = FileUploadImage.PostedFile.ContentType;
        blob.UploadFromStream(FileUploadImage.FileContent);

        FunnyAppRepository<Post> postRepository = new FunnyAppRepository<Post>();
        FunnyAppRepository<Tag> tagRepository = new FunnyAppRepository<Tag>();

        MembershipUser user = Membership.GetUser(Page.User.Identity.Name);
        if (user != null)
        {
            Post post = new Post
                {
                    PostContent = TextBoxDescription.Text,
                    PostTitle = TextBoxTitle.Text,
                    State = false,
                    UserId = user.ProviderUserKey.ToString()
                };

            string[] tags = TextBoxTag.Text.Split(';');
            foreach (string tag in tags)
            {
                if (!string.IsNullOrEmpty(tag))
                {
                    tagRepository.Create(new Tag()
                        {
                            PostRowKey = post.RowKey,
                            PostPartitionKey = post.PartitionKey,
                            TagName = tag,
                        });
                    tagRepository.SubmitChange();
                }
            }

            postRepository.Create(post);
            postRepository.SubmitChange();

            // Kuyruk nesneleri
            CloudQueue queue = _queueClient.GetQueueReference(Utils.CloudQueueKey);
            // mesaj içeriğinin oluşturulması
            // mesaj içerisinden birden fazla bilgi olması sebebi ile "," karakteri i
le bilgiler birbirinden ayrılmıştır.
            CloudQueueMessage message =
                new CloudQueueMessage(string.Format("{0},{1},{2}", blob.Uri, post.Par
titionKey, post.RowKey));
            // Mesaj kuyruğa eklenmiştir.
            queue.AddMessage(message);
```

```
            LabelResult.Text = "Uploaded";
        }
        else
        {
            LabelResult.Text = "Failed";
        }
    }
}

private void InitializeStorage()
{
    if (_storageInitialized)
    {
        return;
    }

    lock (_look)
    {
        if (_storageInitialized)
        {
            return;
        }

        try
        {
            // hesap bilgilerinin alınması
            CloudStorageAccount storageAccount = CloudStorageAccount.FromConfiguratio
nSetting(Utils.ConfigurationString);

            // image blob taşıyıcısının oluşturulması
            _blobClient = storageAccount.CreateCloudBlobClient();
            CloudBlobContainer container = _blobClient.GetContainerReference(Utils.Cl
oudBlobKey);
            container.CreateIfNotExist();

            // Blob taşıyıcısına ile ilgili erişim ayarlarının tanımlanması
            var permissions = container.GetPermissions();
            permissions.PublicAccess = BlobContainerPublicAccessType.Container;
            container.SetPermissions(permissions);

            // create queue to communicate with worker role
            _queueClient = storageAccount.CreateCloudQueueClient();
            CloudQueue queue = _queueClient.GetQueueReference(Utils.CloudQueueKey);
            queue.CreateIfNotExist();
        }
        catch (WebException exception)
        {
            Trace.Write(exception.Message);
        }

        _storageInitialized = true;
    }
}
```

Uygulama Senaryosu – II

Mesaj kuyruğunun dinlenmesi ve ilgili bilgilerin alınması gerekmektedir. Söz konusu bilgilerin alınması ile **Worker Role** içerisinde resim işleme süreci istenilen boyutlara şekillendirilecektir. Dinleme işlemi, uygulama yaşam süresi boyunca, kullanıcı taleplerinin gerçekleştirilmesi amacı ile sonsuz bir döngü içerisinde gerçekleştirilmektedir.

```
// işlem gerçekleştirlecek olan nesnelerin tanımlanması
private CloudQueue _queue;
private CloudBlobContainer _container;

public override void Run()
{
    // Worker nesnesi içerisinde sürecin sürekli olarak gerçekleştirilmesi amacı ile
    // sonsuz döngüye alınması gerekmektedir.
    while (true)
    {
        try
        {
            // İşlem sürecinde kuyruk içerisinde bulunan mesajın dinleme işlemi
            CloudQueueMessage message = _queue.GetMessage();
            if (message != null)
            {
                string[] messageArray = message.AsString.Split(new char[] { ',' });
                string outputBlobUri = messageArray[0];
                string partitionKey = messageArray[1];
                string rowkey = messageArray[2];

                // Kuyruk içerisinde okunan mesajı, döngü içerisinde tekrar okuması ö
nemek amacı ile
                // mesaj Peek edilmektedir.
                _queue.PeekMessage();

                string inputBlobUri = Regex.Replace(outputBlobUri, "([^\\.]+)(\\.[^\\
.]+)?$", "$1-myimage$2");

                _container.CreateIfNotExist();
                CloudBlob inputBlob = _container.GetBlobReference(outputBlobUri);
                CloudBlob outputBlob = _container.GetBlobReference(inputBlobUri);

                var u = inputBlob.Uri.AbsolutePath;

                // Blob nesnesi içerisinde bulunan resim içeriği okunması
                using (BlobStream input = inputBlob.OpenRead())
                using (BlobStream output = outputBlob.OpenWrite())
                {
                    ProcessImage(input, output);

                    output.Commit();
                    outputBlob.Properties.ContentType = "image/jpeg";
                    outputBlob.SetProperties();
```

```
                    FunnyAppRepository<Post> postRepository = new FunnyAppRepository<
Post>();
                    Post post = postRepository.Find(partitionKey, rowkey);

                    post.PostImage = inputBlobUri;
                    post.State = true;
                    postRepository.Update(post);
                    postRepository.SubmitChange();

                    _queue.DeleteMessage(message);
                }
            }
        }
        catch (StorageClientException e)
        {
            Trace.Write(e);
        }
    }
}

public override bool OnStart()
{
    // Worker üzerinde başlatılması amaçlanan varsayılan bağlantı sayısı
    ServicePointManager.DefaultConnectionLimit = 12;
    // Uygulama içerisinde bağlantı bilgilerinin alınma işlemi
    CloudStorageAccount.SetConfigurationSettingPublisher((configName, configSetter) =
>
        configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)));

    // Uygulama içerisinde bağlantı bilgilerinin alınma işlemi
    var storageAccount = CloudStorageAccount.FromConfigurationSetting(Utils.Configura
tionString);

    // kuyruk nesnesi ile ilgili refeasnların oluşturulması
    CloudQueueClient queueStorage = storageAccount.CreateCloudQueueClient();
    _queue = queueStorage.GetQueueReference(Utils.CloudQueueKey);

    // kuyruk nesnesinin bulunup-bulunmadığı kontrollünün yapılması
    _queue.CreateIfNotExist();

    CloudBlobClient blobStorage = storageAccount.CreateCloudBlobClient();
    _container = blobStorage.GetContainerReference(Utils.CloudBlobKey);

    _container.CreateIfNotExist();

    return base.OnStart();
}

// Resim boyutlandırma işlemleri ile fonksiyon
public void ProcessImage(Stream input, Stream output){ ... }
```

İş uygulamalarının, her yeni gün daha fazla kullanılması ile iş yükleri artırılmaktadır. Gerçekleştirilen her sürecin, devamlı ve başarılı sonuçlar üretmesi beklenmektedir. Süreçlerin planlandığı gibi gerçekleşmesi, süreç içerisinde kullanılan uygulamanın sağlıklı çalışması büyük önem taşımaktadır.

Not: Yapılan anlatımın örneklenmesi amacı ile *"**WindowsAzure.FunnyApp**"* uygulaması hazırlanmıştır. Aşağıdaki bağlantı kullanarak, uygulama kaynak kodlarına erişebilirsiniz.

Github / https://github.com/ibrahimatay/WindowsAzure.FunnyApp

İş uygulamaların başarılı ve performanslı çalışması, uygulama yazılım mimarisi ve konumlandırıldığı platform ile ilişkili olarak gerçekleşmektedir. **Windows Azure Cloud Service** uygulama geliştirme modeli ile geliştirilen uygulamalar, iş sürecine bağlı olarak parçalarına ayrılarak, beklenen başarı ve performans elde edilebilmektedir. Konu ile ilgili sorularınızı info@ibrahimatay.org eposta adresine yöneltebilirsiniz.

İbrahim ATAY

Windows Azure Queues and Windows Azure Service Bus Queues - Compared and Contrasted

http://msdn.microsoft.com/en-us/library/hh767287(VS.103).aspx

What is a cloud service?

http://www.windowsazure.com/en-us/manage/services/cloud-services/what-is-a-cloud-service/

The NIST Definition of Cloud Computing

http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf