

Daron Yöndem

IE8



Windows®
**Internet
Explorer®8**

Hızlı Kolay ve Daha Güvenli!

INTERNET EXPLORER 8

**DARON
YÖNDEM**

KODLAB

Yayın Dağıtım Yazılım ve Eğitim
Hizmetleri San. ve Tic. Ltd. Şti.

KODLAB 3

INTERNET EXPLORER 8

DARON YÖNDEM

ISBN 978-605-4205-03-5

Yayincılık Sertifika No: 13206

1. Baskı: Mart 2009

Yayın Yönetmeni: Suat Özdemirci

Kapak Tasarım: Nebi Yiğiroğlu

Sayfa Düzeni: Fikret Eldem

Satış: Hüseyin Üstünel

Baskı: Şefik Matbaası, Tel: (212) 671 59 81

Bu kitabın bütün yayın hakları Kodlab Yayın Dağıtım Yazılım ve Eğitim Hizmetleri San. ve Tic. Ltd. Şti.'ne aittir. Yayınevimizin yazılı izni olmaksızın kısmen veya tamamen alıntı yapılamaz, kopya çekilemez, çoğaltılamaz ve yayınlanamaz.

KODLAB Yayın Dağıtım Yazılım ve Eğitim Hizmetleri San ve Tic Ltd.Şti

Yerebatan Caddesi Salkım Söğüt Sokak Keskinler İş Hanı No: 8
D: 401 Sultanahmet / İSTANBUL

tel: (212) 514 55 66 | **fax:** (212) 514 66 61

e-posta: bilgi@kodlab.com | **web:** www.kodlab.com

DARON YÖNDEM

DEVELOLOAD Yazılım kurucusu Daron Yöndem Microsoft tarafından 2008 ve 2009 yıllarında ASP.NET alanında Most Valuable Professional olarak seçilmiştir. Uluslararası bir konuşmacı olarak Daron Yöndem aynı zamanda Microsoft Regional Director ünvanına sahiptir. Türkiye'de iki kitabı olan Daron Yöndem, International .NET Association'da Türkiye Başkanlığı ve Ortadoğu Afrika bölgesi konuşmacı ofisi başkanlığı yapmaktadır. Kendisine <http://daron.yondem.com/tr/> adresinden blogundan ulaşabilirsiniz.

ÖNSÖZ

Bundan yıllar önce bana çıkışip “Bir gün Internet Explorer kitabı yazacağım” deseler sanırım güzel bir esprî olarak karşılardım. Nasıl olur da bir tarayıcının kitabı yazılır? Ne gerek var? Değil mi?

Oysa bugün kendimi bu kitapçıyı yazarken buluyorsam aslında bu bazı şeylerin değiştiğinin de kanıtı. Belki de tarayıcıları artık haklarında kitap yazılabilecek işlevsellikte yazılımlar olarak görmenin zamanı geldi. Bugün Internet Explorer 8'e baktığında yazılımcılar ve tasarımcılara anlatılabilecek o kadar güzel şeyler görüyorum ki bunların hepsini derleyip sizlere minik bir kitapçık olarak ulaştırmayı uygun gördüm.

Kitapçığın yazım süreci Internet Explorer 8'in Beta 1 günlerinde başladi ve yayındığı şu anda Internet Explorer 8 RC1 sürümü ile karşımızda. Son sürümde doğru ilerleyen bu yolda kitabın yayınından sonra ulaşabilecek tüm gelişmeleri <http://daron.yondem.com/tr/> adresinden, blogumdan takip edebilirsiniz. Ayrıca bana her türlü yorumunuza daron@yondem.com adresinden ulaşabilirsiniz.

Son olarak, unutmadan gönülleri rahatlatmak açısından kitapçığımıza tamamen ücretsiz olduğunu ve aynı sizin elinize geçtiği gibi sizin de ücretsiz olarak dostlarınıza ister dijital ister baskı kopyasını alıp gönderebileceğinizi belirtmek isterim.

Görüşmek üzere...

DARON YÖNDEM
Microsoft Regional Director, MVP

Internet Explorer; her zaman internet deneyimini en üst seviyede, hızlı ve güvenli yaşayabilmemiz için Microsoft olarak geliştirmiş olduğumuz ve günlük online dünyamızın önemli bir parçası olan tarayıcımız. Web 2.0 ile birlikte değişen kullanıcı beklenileri ve Web 3.0 için önemli bir altyapı oluşturacak Anlamsal Web için tasarlanmış olan en yeni sürümü olan Internet Explorer 8 hakkında yazılmış elinizde tuttugunuz bu kitap tüm teknik gelişmeleri detaylarıyla içermektedir. Microsoft Regional Director ve MVP ünvanlarına sahip olan Daron Yöndem tarafından yazılmış bu kitap Türkiye'de Internet Explorer 8'e özel yeni içeriklerin nasıl geliştirilebileceğini ve yeni gelen teknik özelliklerle yeni nesil web teknolojilerine nasıl adapte olunabileceğini detaylarıyla anlatmaktadır. Daron Yöndem'i bu başarılı çalışmasından dolayı kutluyor, bu kitaptaki paylaşımların yeni fikirler ve projeler geliştirecek birçok internet girişimcисine öncülük edeceğini düşünüyorum.

MEHMET NURI ÇANKAYA

Windows Pazarlama Grup Müdürü

Microsoft Türkiye

içindekiler

ÖNSÖZ

1 NEDEN INTERNET EXPLORER 8?	1
Performans	1
Aynı Anda 2 Yerine 6 Bağlantı	2
Script Performansları	2
Kolay Kullanım	2
Yeni Trendler: Accelerator, Web Slice, Search Suggestion	2
Güvenlik	3
Yazılım Geliştirici ve Tasarımcı Dostu	3
Sistem Yöneticileri	4
2 INTERNET EXPLORER 8 DEVRİMİ	7
Ekstra Güvenlik	8
InPrivate Pencereleri	8
InPrivate Blocing	9
Phishing Koruması ve SmartScreen	9
Kolay Kullanım	11
ActiveX Yüklemeleri	11
Sekme Renklendirme	12
Akıllı Adres Çubuğu	14
Yeni Trendler	15
Hızlandırıcı (Accelerator)	15
Görsel Arama (Visual Search)	18
Web Slice	20
3 WEB'DE YENİ TRENDLER	25
Bir Hızlandırıcı (Accelerator) Nasıl hazırlanır?	25
Son Adımlar...	30
Artık Hazırız	32
Görsel Arama Nasıl Hazırlanır?	32
Görsel Arama ve Arama Sağlayıcılarının İşbirliği	35
Sunucu Tarafında Neler Yapmalı?	36
Sunucuda Gerçek Zamanlı XML Üretimevi	39
C# ve ASP.NET ile XML Üretmek	39
VB ve ASP.NET ile XML Üretmek	41

PHP ile XML Üretmek	44
Görsel Aramalar için JSON Veri Kaynağı	46
Web Dilimleri Nasıl Hazırlanır	47
Çalışma Yapısı	48
Display Source Değişikliği	48
Farklı Web Dilimleri’ni Birbirine Bağlamak	50
Web Dilimi’ne Kaynak Olarak RSS Kullanmak	50
4 YAZILIM GELİŞTİRİCİLER İÇİN INTERNET EXPLORER 8	53
JavaScript Yenilikleri	53
tostaticHTML Metodu	53
Dahili JSON Sınıfları...	55
AJAX Navigasyon	57
Online ve Offline Çalışma Yapısı	58
Cross Domain Request Nesnesi	60
DOM Veri Ambarı	64
Session Storage	64
Local Storage	65
Yazılımcı Araçları	65
CSS ve HTML DOM Gerçek Zamanlı Düzenleme	65
JavaScript Debuging	67
Profiler ile Optimizasyon	68
Image Optimizasyonu	69
Pratik Araçlar	70
Önce Güvenlik	71
Data Execution Prevention	71
Kullanıcıya Özel ActiveX	72
Siteye Özel ActiveX	73
XSS Saldırıları	74
MIME TYPE Kararları	75
O Uygulamayı Sitemde Çalıştırma!	75
DOM Elementlerini Sorgulayın	75
5 TASARIMCILAR İÇİN INTERNET EXPLORER 8	77
Compatibility View	77
Sitemizi Nasıl Ayarlarız?	79
Özel CSS Filtreleri	80

NEDEN INTERNET EXPLORER 8?

Performans	1
Aynı Anda 2 Yerine 6 Bağlantı	2
Script Performansları	2
Kolay Kullanım	2
Yeni Trendler: Accelerator, Web Slice, Search Suggestion	2
Güvenlik	3
Yazılım Geliştirici ve Tasarımcı Dostu	3
Sistem Yöneticileri	4

Tarayıcı savaşları gün geçtikçe kızışıyor. Global anlamda kullanım oranlarına baktığımızda Internet Explorer bu savaşın galibi gibi gözükse de bu durum savaşın devam etmediği anlamına tabi ki gelmiyor. Önümüzdeki dönemde Internet Explorer 8.0, bu savaşın en ağır toplarından biri diyebiliriz. Peki, neden **Internet Explorer 8?**

PERFORMANS

Bir internet tarayıcısından tek beklediğimiz özellik performans olmasa da, bu durum performansın önemini yadsiyacağımız anlamına gelmiyor. Internet Explorer 8.0 ile beraber performans noktasında yapılan belki de en büyük yenilik artık her bir tarayıcı sekmesinin arka planda ayrı birer uygulama olarak çalışıyor olması. Böylece tek bir uygulama içerisinde thread sayısı azaltılabilirken sekmelerin birbirinden bağımsız olarak hata alması halinde birbirlerini kurtarabilmeleri gibi avantajlar da ortaya çıkıyor. Tüm bunların detaylarından kitabımın ileriki bölümlerinde kullanıcı taraflı yenilikleri ele alırken detaylı olarak bahsedeceğiz.



THREAD

Herhangi bir uygulama içerisinde aynı anda çalışabilen ayrı kod bloklarını tanımlar. Bir uygulama içerisinde birden çok **Thread** bulunup aynı anda çalışabilibiliği üzere aynı uygulamanın kaynaklarını Thread'ler beraber de kullanabilirler.

AYNI ANDA 2 YERİNE 6 BAĞLANTI

Yıllar önce, 1999 senesinde, HTTP 1.1 standartlarına göre tarayıcıların aynı anda bir sunucuya sadece iki bağlantı kurabileceklerine dair bir kural belirlenmiş. Aradan geçen 9 yılda artık bu sınırın günümüz geniş bant internet altyapısında çok da anlamlı olmadığını söyleyebiliriz. IE 8.0 ile beraber bu sınır altya yükseltiliyor. Aynı anda 2 bağlantı yerine 6 bağlantı teorik olarak yaklaşık üç kat hızlanma anlamına geliyor. Bir yazılımcı olarak sayfanıza eklediğiniz 6 harici JavaScript dosyasının, birbirlerini beklemek yerine hep beraber aynı anda kullanıcı taraflına indirilmeye başlanması ciddi performans artışı sağlayacaktır.

SCRIPT PERFORMANSLARI

AJAX’ın kullanılmadığı web sitesi neredeyse kalmadı. İstemci tarafında Script’ler kullanarak artık çoğu işlemi sunucu yerine istemcide yapmayı tercih ediyoruz. Bu çerçevede Script’lerin çalışma hızı bir tarayıcının genel performansı ile çok yakından alakalı. Haberler güzel; IE 8.0 JavaScript **SunSpider** testinde IE 7.0'a kıyasla %400 daha hızlı!

Kendi testinizi yapmak isterseniz SunSpider testine aşağıdaki adresden ulaşabilirsiniz;

<http://www2.webkit.org/perf/sunspider-0.9/sunspider.html>

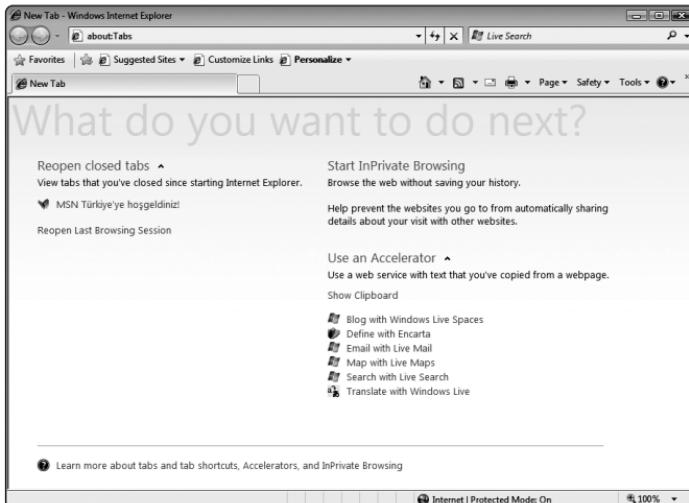
KOLAY KULLANIM

Çok hızlı bir tarayıcıya sahip olalım derken tabii ki işlevselliklerden de vazgeçmemeyiz. IE 8.0 ile beraber belki de Internet Explorer tarihindeki en güzel araçlar geliyor. Bu araçlar hem doğrudan kullanıcıların hayatını kolaylaştırırken hem de yazılım geliştiricileri ve tasarımcıları da mutlu ediyor.

YENİ TRENDLER: ACCELERATOR, WEB SLICE, SEARCH SUGGESTION

Internet Explorer’ı kullanım şekliniz değişecek. Kitabımızın ileriki bölgümlerinde hem kullanıcı taraflı hem de yazılımcı ve tasarımcılar için detaylı olarak inceleyeceğimiz **Accelerator**, **Web Slice** ve **Search Suggestion** yapıları siz Internet Explorer’da gönülden bağlayacak. **Accelerator’lar** ile sürekli internette gezinirken tekrarladığınız işlemlerin farkına varacak ve kendinize daha hızlı bir çalışma ortamı sağlayacaksınız. **Web Slice’lar** ise sizin yerinize interneti takip edecek. **Search Suggestion** sistemini gördüğünüzde aradığınızı bulmanın ne kadar ko-

laylaştığını göreceksiniz. Detayları girmek için sabırsızlandığınızı biliyorum, az kaldı.



Internet Explorer ilk açılışında yeni özelliklere hızlıca ulaşabileceğiniz linkler karşınıza geliyor.

GÜVENLİK

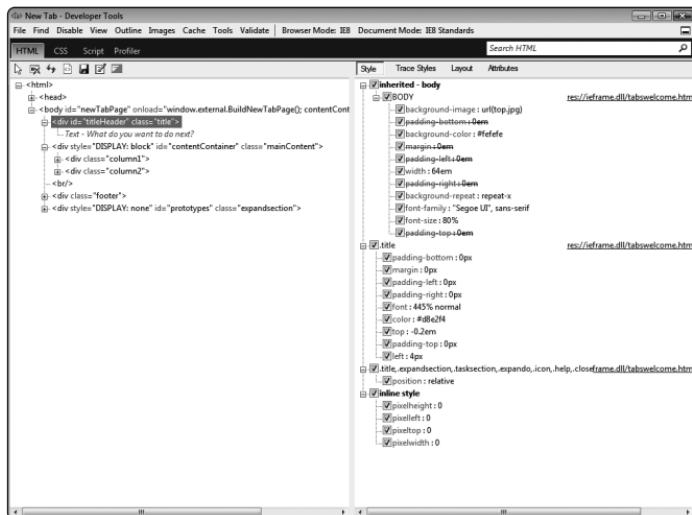
InPrivate tarayıcı özelliği ile tanıtıığınızda gönlünüz o kadar rahat edecek ki kullandığınız bilgisayarlarda başka bir internet tarayıcıaramayacaksınız. Sadece size özel bir Internet Explorer oturumu düşünün ki siz kapattığınızda sizinle ilgili hiçbir iz kalıyor. Ne tarayıcı geçmiş, ne **cookie** (*cerezler*) ne de yazdığınız bilgiler ve şifreleriniz. Tarayıcıyı kapattığınız anda her şey uçup gidiyor. Gönül daha ne ister?

Phishing koruması, **XSS** engelleme sistemleri ve daha birçok yenilik var. Tüm bunların detaylarını da ileriki bölümlerde inceleyeceğiz.

YAZILIM GELİŞTİRİCİ VE TASARIMCI DOSTU

IE 8.0 ile beraber yazılım geliştiriciler ve tasarımcılar olarak da mutluluğun yollarında ilerliyoruz. CSS 2.1 desteği, HTML 5, AJAX ve JavaScript yenilikleri, Font Embedding düzenlemeleri ile artık IE 8.0'da çalışacak web sitelerini tasarlamak ve programlamak çok daha zevkli olacaktır. Kitabımızın özellikle yazılımcılara ve tasarımcılarla

hitap eden bölümlerde teknik detaylardan da bahsederek Internet Explorer 8.0'ın bir yazılımcının veya tasarımcının hayatını da nasıl kolaylaştırdığını göz atacağız.



Internet Explorer 8.0 ile beraber gelen **Developer Tools** paketi yazılımcılar ve tasarımcıların işini kolaylaştırmayı hedefliyor.

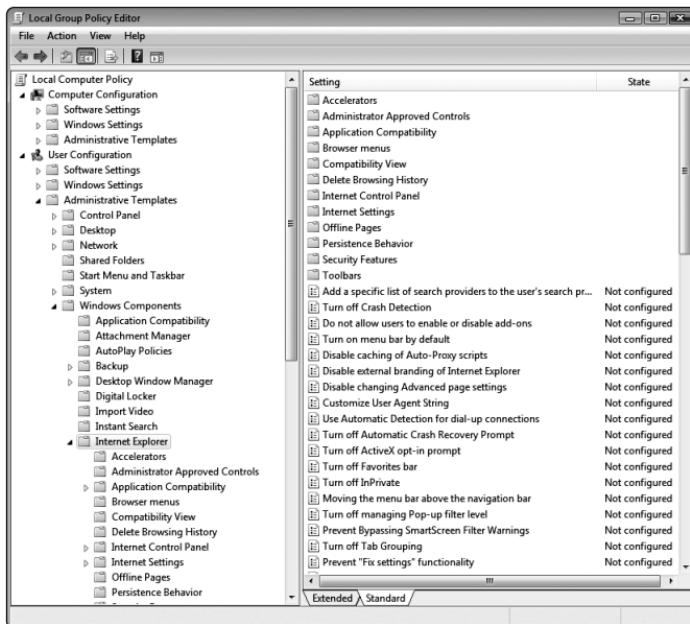
SİSTEM YÖNETİCİLERİ

Birçok yeni özellik geliyor, tüm bunların tabii ki güvenlik tarafında da kontrol mekanizmaları ile ele alınabilmesi şart. Windows Server 2008 üzerinde Internet Explorer 8 ile beraber IE çerçeveli **Group Policy** sayısı toplamda 1300'ü geçiyor. Tabii tüm bunları ancak istemcilere IE 8.0 yükledikten sonra kullanabilirsiniz, bunun için de toplu yükleme seçeneklerinin yanı sıra kendinize özel IE8.0 yükleme paketleri oluşturmanız için gerekli detayları da kitabımızda inceliyoruz.



GROUP POLICY

Windows Server işletim sistemlerinde Active Directory üzerinde bilgisyarlarında ve uzak kullanıcıların merkezi yönetim ve merkezi konfigürasyonunu sağlar.



Internet Explorer 8.0 ile beraber gelen **GPO**'lardan bazıları

İşte tüm bu yenilikler ve eşsiz kolaylıklar nedeniyle Internet Explorer 8.0! Daha fazlası için sayfayı çevirin...

6 INTERNET EXPLORER 8

INTERNET EXPLORER 8 DEVRİMİ

Ekstra Güvenlik	8	Yeni Trendler	15
InPrivate Pencereleri	8	Hızlandırıcı (Accelerator)	15
InPrivate Blocking	9	Görsel Arama (Visual Search)	18
Phishing Koruması ve SmartScreen	9	Web Slice	20
Kolay Kullanım	11		
ActiveX Yüklemeleri	11		
Sekme Renklendirme	12		
Akıllı Adres Çubuğu	14		

Internet Explorer’ın tarihine baktığımızda “Devrim” kelimesi artık özlediğimiz sözcükler arasında yerini almıştı. Peki, nasıl oluyor da 8.0 sürümü ile bu özlem bitiyor? Kitabımızın bu bölümünde Internet Explorer 8’i bir son kullanıcının gözünden inceleyerek ilk bakışta göze çarpan yeniliklere değineceğiz. Tüm bu yenilikler önmüzdeki dönemde günü birlik işlerimizi kolaylaştıracak ve hayatımızın bir parçası haline gelecek yeni işlevsellikleri kapsıyor.

EKSTRA GÜVENLİK

Normal hayatı olduğu gibi dijital hayatı da her şeyin yanı sıra en çok önem verdığımız detaylardan biri de **Güvenlik** konusu. Özellikle internette gezerken bizim bilgimiz dahilinde olsun veya olmasın sisteminizin güvenliği ile ilgili riskler belki de canımızı en çok sikan noktalar arasında.

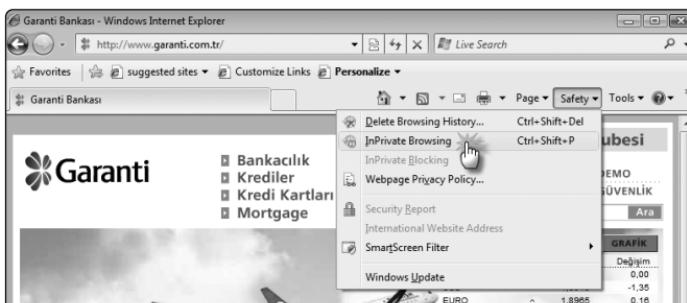
Bir internet tarayıcısının tüm “hayatımıza kolaylaştıran” özelliklerini bir kenara koyup söz konusu tarayıcıyı kullanıp kullanmayacağı konusunu kararını belki de sadece sağladığı güvenlik özellikleri çerçevesinde bile verebiliriz. Özellikle E-Ticaret’in gün geçtikçe hayatımızın göbeğine oturduğunu düşünürsek IE 8.0 ile beraber gelen kullanıcı tarafındaki güvenlik yenilikleri çok önemli.

INPRIVATE PENCERELERİ

Bir yazılımcı olarak incelediğimiz yazılımlara gelen yeni özelliklerini bazılarını gördüğümde “Neden bu kimsenin aklına bugüne kadar

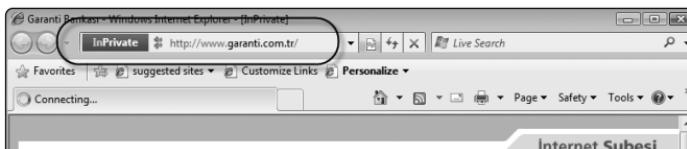
gelmedi?” sorusunu kendime soruyorum. IE 8.0’ın **InPrivate Browsing** özelliği de bu özelliklerden biri.

Bir Internet Explorer penceresi düşünün ki kapattığınızda her şey es-kisi gibi olacak. Tarayıcı geçmiş, cookie, Temp dosyaları ve geri kalan her şey tarayıcıyı kapattığınız anda otomatik olarak yok olsa, silinse ne kadar güzel olurdu değil mi? Tabi tüm bunların varsayılan ayarlıarda olmaması gerekiyor. Sadece istediğimizde böyle geçici bir pencere oluşturabilmeliyiz.



InPrivate pencereleri açabilmek için tarayıcı içerisindeki **Safety** menüsünden **InPrivate Browsing** komutunu verebilirsiniz.

InPrivate tarayıcı pencereleri bizim tüm bu isteklerimizi karşılıyor. InPrivate olarak açtığımız bir pencereyi kapattığımızda o pencere ile ilgili her şey yok ediliyor. IE 8.0 içerisinde **Safety→InPrivate Browsing** menüsünden açabildiğimiz InPrivate pencerelerini adres çubuğuundaki InPrivate yazısından da tanıyalıyoruz.



InPrivate pencerelerinde adres çubuğuunda **InPrivate** logosunu görebilirsiniz.

InPrivate pencerelerini özellikle internet cafe gibi ortamlarda hatırlamak ve kendi bilgisayarınızda olmamanıza rağmen özel bilgilerinizi girmek veya izlemek zorunda olduğunuz durumlarda sıkça kullanmakta güvenlik açısından büyük fayda olacaktır.

INPRIVATE BLOCKING

Doğrudan son kullanıcının hissedeceği bir sistem olmasa da InPrivate Browsing'e ek olarak ayrıca bir de **InPrivate Blocking** mekanizması var. Bu sistemi anlatmanın en kolay yolu sistemin engellediği güvenlik açığını anlatmak geçiyor.

Varsayılmı **birinci.com** sitesini geziyoruz ve bu esnada **sinsi.com** sitesinden bir JavaScript dosyası çağrıldı. Artık **sinsi.com** bizi tanıyor, orada da bir oturumumuz var. Bir sonraki adımda **ikinci.com** adında bir siteyi ziyaret ettik ve şans eseri bu sitede de yine **sinsi.com**'dan aynı JavaScript dosyası eklenmiş. An itibarı ile **sinsi.com** bizim **ikinci.com**'a gelmeden önce hangi sitede bulunduğumuzu biliyor. Hatta bu senaryo daha da uzatılabilir ve aynı tarayıcı oturumu içerisinde 20 önceki siteye kadar nereleri gezdigimiz **sinsi.com** tarafından takip ediliyor olabilir. Tabi bunların hepsinin gerçekleşmesi için aynı JavaScript'in tüm bu sitelerden çağrılmış olması şart.

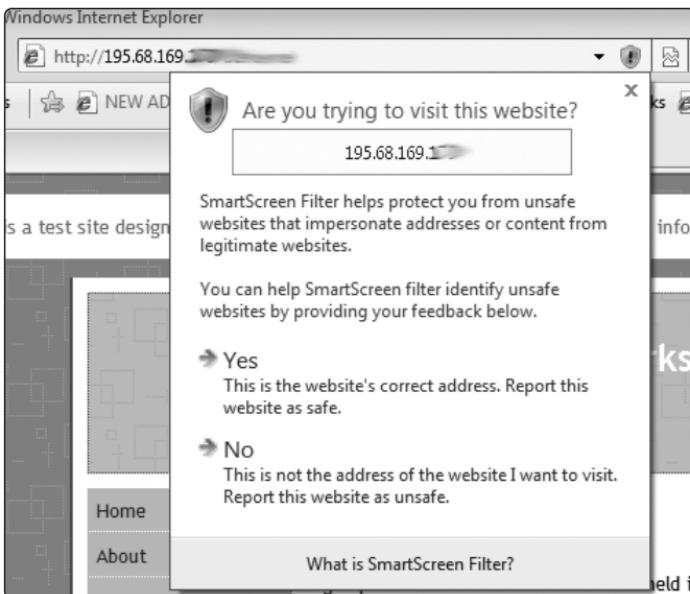
İşte yukarıda anlattığımız güvenlik açığını engellemek amacıyla InPrivate Blocking geliştirilmiştir. Sistemin amacı kullanıcıya özel bilgilerin farklı sitelere kullanıcı farkında olmadan ulaşmasını engellemek. Eğer InPrivate Blocking aktif ise tarayıcı her sitede harici olarak yüklenen script'lerin listesini saklıyor ve durumuna göre bazı script'leri de aktif edebiliyor.

PHISHING KORUMASI VE SMARTSCREEN

Phishing olarak adlandırdığımız güvenlik açığı aslında ağırlıklı olarak sosyal bir saldırısı mekanizması da sayılabilir. Phishing'i teknik olarak tamamen engellemek mümkün değil. Peki, nedir Phishing? Basit bir örnek olarak internet bankacılığı mekanizmalarını ele alalım.

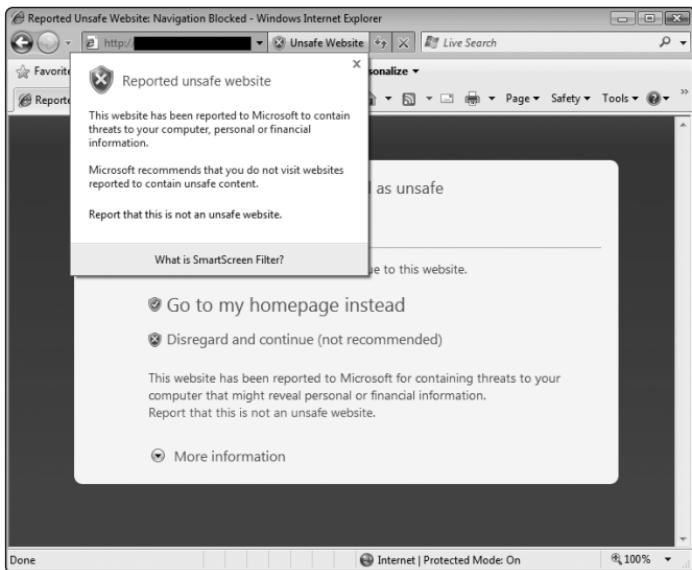
Varsayılmı her gün çalışığınız bankanın web sitesi olan **bankam.com** adresini ziyaret ediyorsunuz. Şans eseri size bir e-posta geldi ve bankanızın web sitesinde bir işlem yapmanız gerektiği söylenenerek "Buraya Tıklayın" tarzında bir link verildi. Tarayıcınız açıldığından **e-bankam.com** adında bir adres görmeniz halinde eğer bu durumdan rahatsız olup adresin gerçekten sizin bankaniza ait olup olmadığını kontrol etmezseniz belki de çoktan düşmanın elliğine düştünüz demektir.

Bu noktada, tabi ki Internet Explorer 8.0'ın hangi alan adının gerçekten bankaniza ait olup olmadığını bilme şansı yok. Fakat en azından kontrol edilebilecek başka noktalar var.



SmartScreen Filter kullanıcının ziyaret ettiği adreste bir gariplik olduğunu algılıyor.

Örneğin, bir alan adı değil de doğrudan bir web sunucusunun IP adresine yönlendirilmiş iseniz büyük ihtimal ile bu dinamik bir IP adresidir ve söz konusu sunucu geçerli bir kurumsal web sunucusu değildir. Bu gibi durumları algılayan **SmartScreen**, Filter Phishing konusunda kullanıcıyı uyararak içerişine düşülebilecek kötü durumları azaltmayı hedefliyor.



SmartScreen Filter “Zararlı” olarak raporlanmış bir siteye erişimi engelliyor.

KOLAY KULLANIM

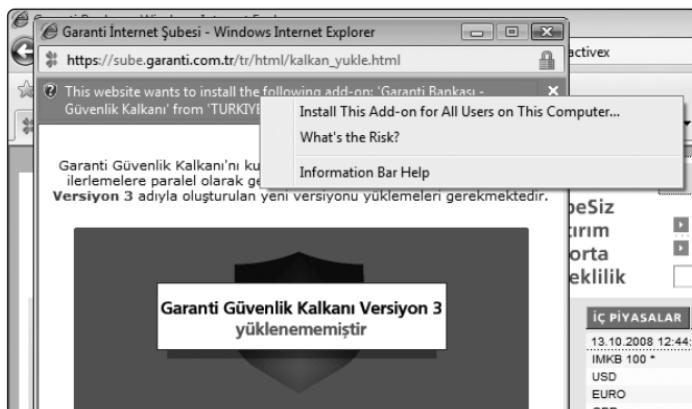
Bilgisayar başında bulunduğuuz süre boyunca en çok kullandığımız yazılımları bir düşünelim. Akliniza gelenler arasında internet tarayıcıınızın ilk beşte olacağindeneminim. Bu kadar yoğun bir şekilde kullandığımız bir yazılımin tabi ki olabildigince işimizi kolaylaştırması ve hızlandırması da gerekiyor. Birazdan IE 8.0 ile beraber gelen ve detaylarda saklı olan ufak ama bir o kadar da değerli özelliklere göz atacağız. Esas sürprizimi bölümün sonuna saklıyorum.

ACTIVE-X YÜKLEMELERİ

ActiveX yüklemeleri için “başımızın belaları” desek herhalde hiç de yanlış olmaz. ActiveX yüklemesi gerektiren bir sayfada karşımıza çıkan güvenlik uyarısı sonrası sayfayı tekrar açmak zorunda kalmak veya yönetici haklarına sahip olmadığımız bir bilgisayarda ActiveX’leri yükleyemediğimiz için çok basit web uygulamalarını bile kullanamamak neredeyse günümüştür sıkıntılardır arasında yer alıyor. Bu sıkıntılar aslında sadece son kullanıcının değil, dolaylı olarak sistem yöneticilerinin ve programcıların da sıkıntıları arasında.

Özellikle Vista ile beraber gelen **UAC** (*User Account Control*) sonrasında bu dertler gergin senaryolara sebep olabiliyor. Ama artık her şey değişti, kullanıcılar yönetici haklarına sahip olmasalar da kendi kullanıcı hesaplarına özel olarak ActiveX uygulamaları yükleyebilecekler. Eğer söz konusu ActiveX uygulaması zararlı bir kod içeriyorsa bu durumun bilgisayara hiçbir şekilde zararı dokunamayacak. En güzel haber ise yazılım geliştiriciler için; var olan ActiveX uygulamaları herhangi bir sorun yaşamadan bu sistem ile çalışabiliyor.

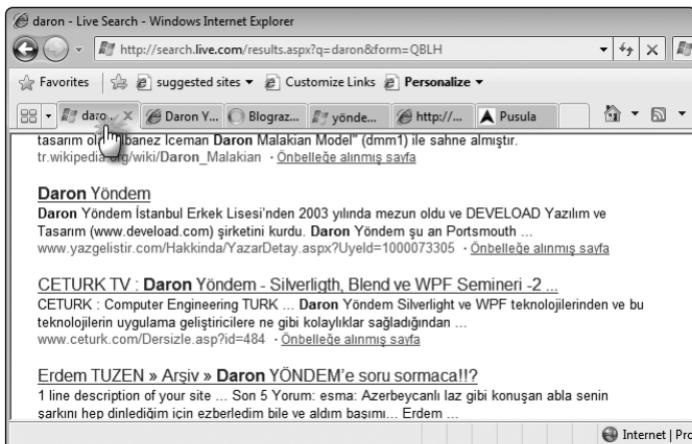
Kullanıcılar herhangi bir ActiveX kontrolü ile karşılaşıklarında kontrolü sadece kendileri için veya tüm makine bazında yüklemek isteyip istemediklerini seçebilecekler. Bu seçim şu anki Internet Explorer içerisinde ActiveX kontrolleri için gelen uyarı mekanizmasına dahil edilmiş durumda.



ActiveX yüklemeleri artık yönetici hakları olmaksızın da çalışabiliyor.

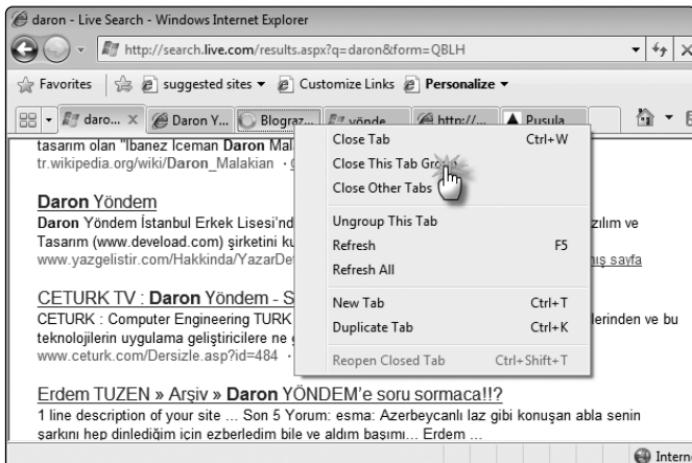
SEKME RENKLENDİRME

Internet Explorer 7 ile beraber internet tarayıcısı içerisinde **sekmeler** (*tabs*) kullanmaya iyice alıştık. Fakat bazen çok sayıda sekme açınca hangi sekmenin hangi konuya ilişkili olduğunu bulmak zor olabiliyor. Varsayılmış ki, iki farklı konuda **live.com** üzerinde birer arama yaptınız ve arama sonuçlarının bulunduğu sayfaları açarak ve yine o sayfalarдан da başka sayfaları yeni sekmlerde açarak araştırmanızı devam ediyorsunuz. Peki, hangi sekmler hangi aramanızla ilişkiliydi?



Farklı kaynaklardan açılan seküler farklı renklerde gösteriliyor.

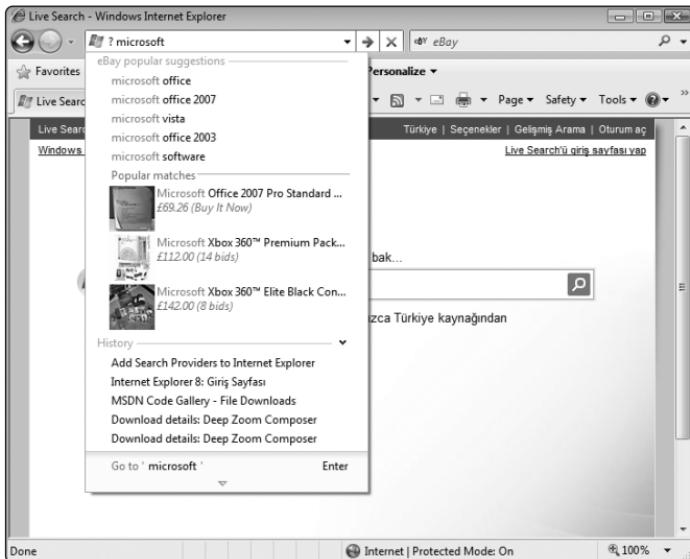
Internet Explorer 8.0 ile beraber artık açığınız her sekme kendi sahibini tanıyor. Bir anlamda tarayıcınız sizin iki veya daha fazla sayıda farklı yoldan interneti gezdiğinizizi algılayarak bu yolları da açık bir şekilde gösterebilmek adına sekülerleri farklı renklerle grupluyor. Hayalet kolaylaştırın ufak ve hoş bir detay değil mi?



Sekme gruplarını sağ tuş ile gelen menüden yönetebilirsiniz.

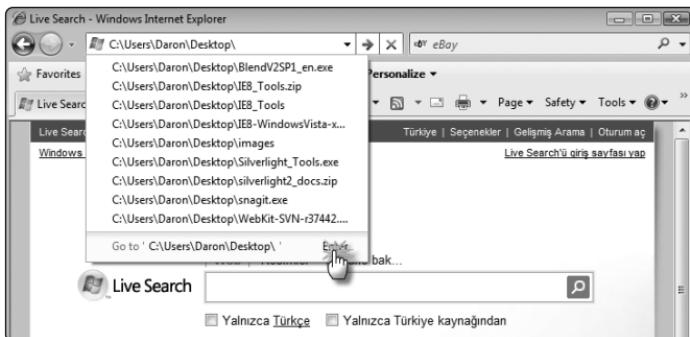
AKILLI ADRES ÇUBUĞU

Tarayıcı içerisindeki adres çubuğu bir tarayıcıda en çok kullanılan yerlerden biridir. Sürekli adres çubuğuna bir adres yazarız, daha önce yazdığımız adresleri ararız ve tüm internet gezintimiz boyunca sürekli adres çubuğunu kullanırız. Internet Explorer ekibi de bunun farkında olduğu için adres çubuğunda da bazı değişiklikler yapılmış. Bundan ilki “Search Suggestion” entegrasyonu. **Search Suggestion** konusunda detaylara ileriki bölümlerde değineceğiz fakat bu aşamada önemli olan bizim kullanıcı olarak istediğimiz bir arama mekanizmasını IE içerisine entegre ettikten sonra adres çubuğundan da bu sistemi kullanabiliyor olmamız.



Adres çubuğunda hem arama sonuçları hem de adres geçmişi beraber.

Adres çubuğunda doğrudan arama yapabilmek için tek yapmanız gereken aratacağınız sözcükleri yazmadan önce bir soru işaretçi (?) koymak. Böylece sistem otomatik olarak varsayılan Search Provider’ın, varsa Search Suggestion mekanizmasını da kullanarak uygun arama sonuçlarını tarayıcı geçmişi ile beraber bir mini rapor olarak sunacaktır.



Adres çubuğuında diskteki klasör ve dosyaları görerek istediğimiz dosyayı IE ile açabiliyoruz.

Belki nadiren de olsa Internet Explorer içerisinde sistemimizdeki dosyaları da açmak isteyebiliriz. Bu dosyalar bazen basit birer JPEG resim dosyası olabilirken belki de doğrudan IE içerisinde açılabilen XPS dosyaları da olabilir. Bu gibi durumlarda adres çubuğundan diskin içeriğini gezerek istediğimiz dosyayı hızlıca bulabilmek büyük avantaj sağlar. İşte karşınızda adres çubuğunda Visual Studio içerisinde Intellisense yapısına benzeyen disk gezgini.

YENİ TRENDLER

Geldik bölümün en heyecanlı kısmına. IE 8.0 ile beraber artık internette sörf sporuna yeni bir akım geliyor diyebiliriz. Bu akımı oluşturan kelimeleri Accelerator, Search Suggestion ve Web Slice şeklinde özetleyebiliriz. Tüm bunlar internette sörf yapma şeklinizi, webi kullanma mantığınızı ve tüm alışkanlıklarınızı değiştirmeye aday yepyeni araçlar. Hemen gelin bu yeni dünyaya dalıp hızlı bir tur atalım ve nasıl olacak da IE alışkanlıklarımızı değiştirecek görelim.

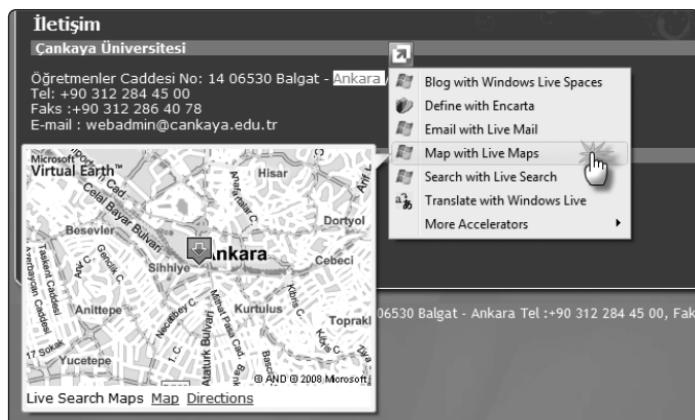
HİZLANDIRICI (ACCELERATOR)

İngilizce kelime anlamı hızlandırıcı olan Accelerator yapısı, aslında farkında olmadan gün içerisinde defalarca kullandığımız web tabanlı servisleri daha rahat ve hızlı kullanabilmemiz için gerekli altyapıyı sunuyor. Gelin beraber bir düşünelim, internette gezinirken sürekli yaptığım ve tekrar ettiğimiz işlemler nelerdir? Sanırım birincisi arama motorlarını kullanmak, kişisel olarak düşündüğümde benim en sık kullandığım servislerden biri de harita servisleri. Bir web sitesinde gördüğüm adresin tam olarak nerede olduğu? Oraya nasıl gidebileceğim?

Başka neler olabilir düşünelim... Sözlük hizmetleri de bu servisler arasında yerini alabilir, bilmediğim bir kelimeyi hızlı bir şekilde Türkçeye çevirerek anlamını öğrenmek istediğimde kütüphaneye uza-nıp bir sözlük bulmaktansa hemen internette online bir sözlük kullan-mayı tercih ederim.

Tüm bu servislere baktığımızda aslında sürekli aynı şeyi tekrar ettiğimizi görebiliriz. İnternette karşılaştığımız bir kelimeyi, adresi veya bilgiyi alır başka bir internet servisine teslim eder ve uygun bir sonu-cu gözle görmek isteriz. Bu sonuç bazen bir harita, bazen bir arama sonu-cu bazen de farklı dilde bir kelimenin açıklaması olabilir. Tüm bu örneklerde aslında hep aynı mekanizma çalışıyor ve bu işlemleri biz defalarca aynı şekilde tekrarlıyoruz. İşte Accelerator'lar bu noktada devreye girip tekrarladığımız bu süreçleri kısaltmayı hedefliyor.

Accelerator'lara ulaşmak için Internet Explorer içerisinde herhangi bir web sitesini gezerken web sitesi üzerinde sağ tuş ile tıklamak ye-terli olacaktır. Karşınıza çıkan menüde her bir Accelerator yapısı için birer komut göreceksiniz. Accelerator'ların bazıları sadece içerisinde bulunanın sayfanın adresi ile ilgilenirken (Örneğin içerisinde bulunduğu sayfayı başka bir dile çevirecek bir Accelerator) bazı-ları ise sadece fare ile söz konusu sayfada bir yazı seçilmişse çalışma-caktır (Örneğin seçili adresi haritada gösteren bir Accelerator).



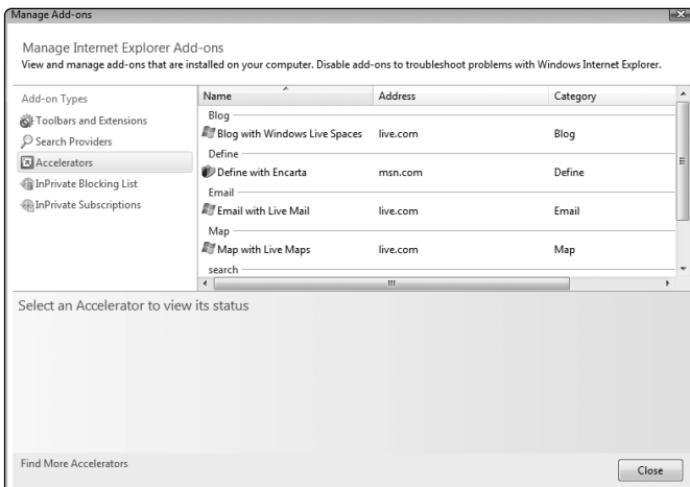
LiveMaps Accelerator'ını kullanarak seçtiğimiz bir adresi doğrudan haritada görebiliyoruz.

Accelerator'ların bazıları yanında sonucu ek bir pencerede gösterebilirken bazıları sadece tıklandığında başka bir sekme veya pencere açarak sonucu gösterebilirler. Sisteminize yeni Accelerator'ları ancak Accelerator sağlayan web sitelerinde uygun linklere tıklayarak yükleyebilirsiniz. Accelerator'ların programatik olarak nasıl hazırlandığı ile ilgili detaylara bir sonraki bölümümüzde değineceğiz.



Manage Accelerator kısmından sistemde yüklü tüm Accelerator'ları yönetebilirsiniz.

Herhangi bir Accelerator ilk defa sisteme yüklendiğinde doğrudan menülerde yerini almaz. **More Accelerators** menüsüne yerleşen yeni Accelerator'ları kullanıcılar isterlerse ana menüye taşıyabilirler. Bu nın için tarayıcı içerisinde sağ tuş ile gelen menüden **Manage Accelerator** komutunu vererek uygun bölüme geçmeleri gereklidir.



Manage Accelerator bölümünden isterseniz sistemdeki Accelerator'ları silebilirsiniz.

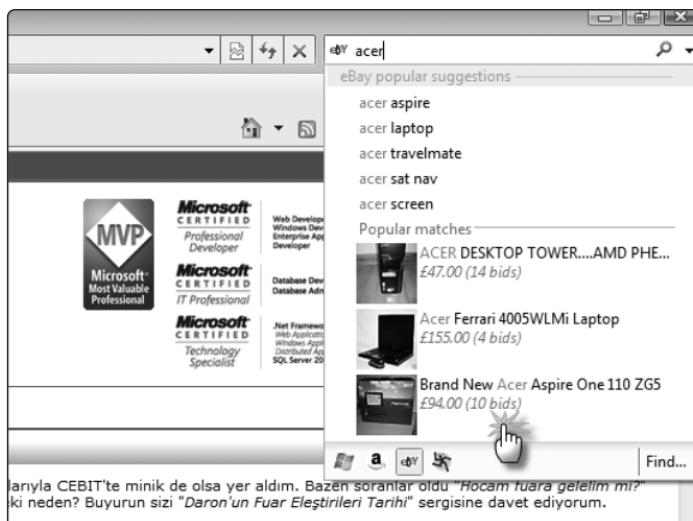
Manage Accelerators bölümünde kullanıcılar hangi Accelerator'ların ana menüde gözükeceğine karar vermek için istedikleri Accelerator'ı seçerek sağ alta gelen düğmelerden **Set as Default** düğmesine tıklayabilirler. Aynı şekilde ana menüde yer alan bir Accelerator'ı alt menüye göndermek için de **Remove as default** düğmesi kullanılabilir. **Manage Accelerators** penceresinde herhangi bir Accelerator seçildiğinde istenirse **Disable** düğmesi ile bu Accelerator pasif hale getirilebilir veya **Remove** düğmesi ile tamamen sistemden kaldırılabilir.

GÖRSEL ARAMA (VISUAL SEARCH)

Internet Explorer 7.0 ile beraber gelen Search Provider yapısı kullanıcılar tarayıcı içerisine kendi tercih ettilerini arama motorlarını entegre etme olanlığı tanıdı. Böylece kullanıcıların tarayıcısının sol üstündeki ufak kutucuğa yazdıkları aramalar otomatik olarak tarayıcıya yüklenmiş Search Provider içerisindeki kurallara göre söz konusu arama motoruna gönderilip tarayıcı içerisinde sonuçlar gösteriliyordu.

Internet Explorer 8.0 ile beraber Search Provider yapısına **Visual Search** adında yeni bir sistem eklendi. Bu sistemin amacı kullanıcılar aramak istedikleri kelimeleri tarayıcı içerisinde yazarken otomatik olarak olası sonuçları yine tarayıcı ile entegre olarak göstermek.

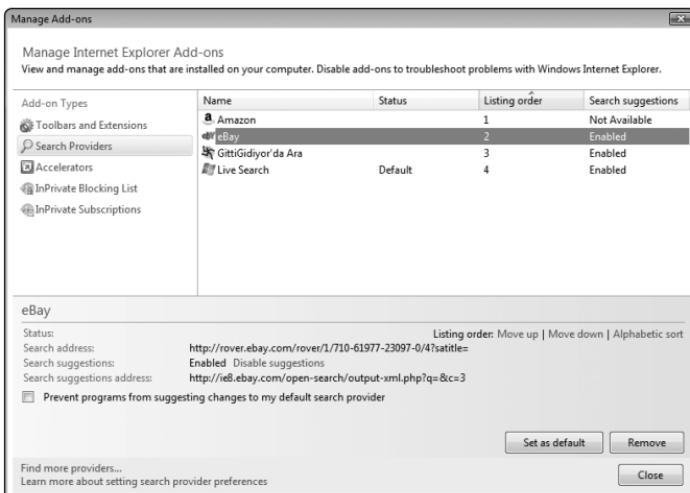
Yazılımcı kimliğiyle konuya bakarsak aslında bir süredir web ortamında alışık olduğumuz **AJAX** ile kurulan **AutoComplete** sistemlerinin bir türevi ile karşı karşıya olduğumuzu söyleyebiliriz. Tabi bu sistemin kullanılabilmesi için Internet Explorer'a arama yapılacak arama motorunun Search Provider'ının yüklenmiş olması ve söz konusu Search Provider'in da Visual Search sistemini desteklemesi gerekiyor. Bu konudaki teknik detayları bir sonraki bölümümüzde derinlemesine inceleyeceğiz.



Görsel Arama sistemi aramalarda otomatik sonuçları gösteriyor.

Sisteminize herhangi bir **Görsel Arama** eklemek için özünde bu desteği sahip bir Search Provider eklemeniz gerekiyor. Search Provider'lara farklı web sitelerinden ulaşabileceğiniz gibi isterseniz doğrudan Internet Explorer'in web tabanlı Search Provider galerisini de **Find More Providers** komutu ile ziyaret edebilirsiniz.

Yüklemiş olduğunuz Search Provider'ları silmek veya varsayılan arama motoru olarak atamak için arama kutucuğunun yanındaki oka basarak **Manage Search Providers** komutunu verebilirsiniz.



Search Provider'ların hepsini bu pencereden yönetebilirsiniz

“Manage Search Providers” altında kayıtlı arama motorlarının tarayıcı arayüzünde gösterilmesi sırasını değiştirebileceğiniz gibi istediğiniz bir Provider yapısını **Remove** düğmesi ile sistemden silebilirsiniz de.

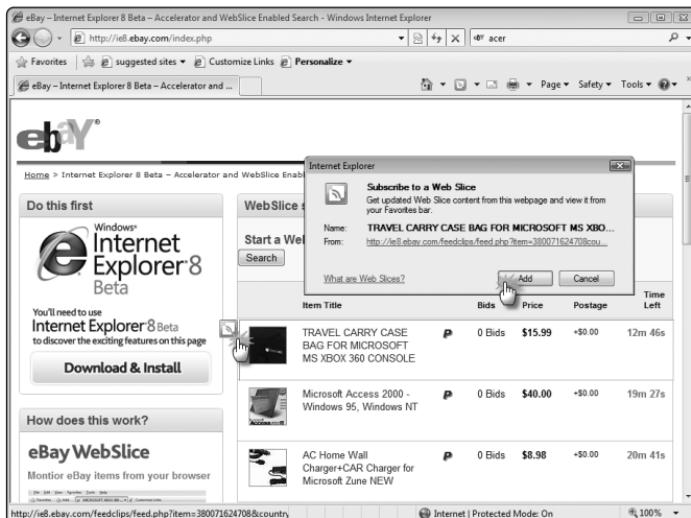
WEB SLICE

Benim en favori özelliğime geldik. **Web Slice**'ların mantığında bir web sitesinin belirli bir bölümünü kesip sürekli o bölümü takibe almak yiyor. Söz konusu kesik bölümde herhangi bir değişiklik olduğunda tarayıcı sizi hemen haberdar ediyor ve ister yine o kesiği ister tüm sayfayı yeniden inceleyebiliyorsunuz.

Tabi sanal ortamda bir şeyleri kesmek pek mümkün değil, o nedenle tam olarak neden bahsettiğimize gerçek bir örnekle göz atalım. Varsayılmak ki en sevdiğiniz haber sitesindeki haberleri takip etmek istiyorsunuz veya bir açık artırma sitesindeki belirli bir ürünün durumunu sürekli kontrol etmeniz gerekiyor. Bu gibi senaryolarda kullanabileceğimiz teknolojilerden biri **RSS**'ler. Eğer karşı taraf size bir RSS kaynağı sağlarsa söz konusu RSS'i **RSS Okuyucu** programınıza ekleyerek takip edebilirsiniz. Bu durum belki bir haber sitesini takip etmek için uygun olsa da maalesef açık artırma sitesindeki örnek için pek kolay bir çözüm olmayacağından emin olabiliriz. Kısacası bir süre için takip etmek istediğiniz ufak verileri tek tek bir RSS okuyucuya eklemenin zahmetini geç-

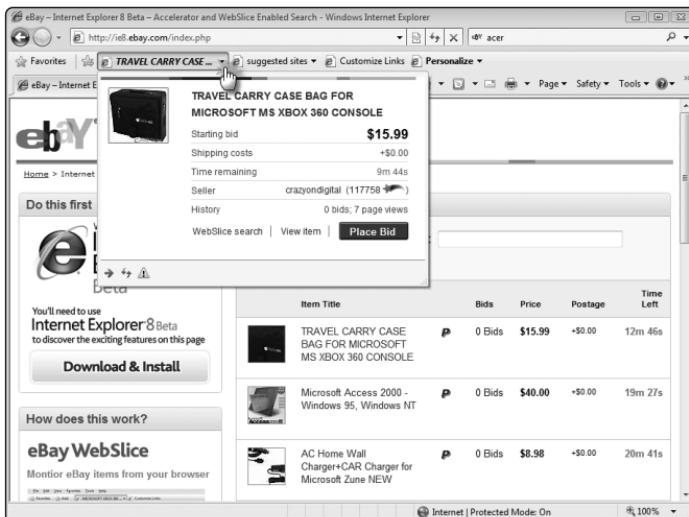
sek bile maalesef araya ikinci bir program veya arayüz eklemiştir olmanın getireceği hantallık da üzerimize çökecektir.

Tüm bu dertlerin içinde boğuşurken kahramanımız Web Slice yardımımıza koşuyor ve web sitelerinde desteklenen bölümleri kesip tarayıcımızın bir parçası haline getirebiliyoruz. Tabi bu işlemin yapılabilmesi için söz konusu web sitesi tarafındaki programcıların ve tasarımcıların yapmaları gereken bazı ufak işler var. Bu işlerin detaylarına bir sonraki bölümde değineceğiz. Şimdilik hali hazırda bu desteği veren bir site ile yola çıkarak **Web Slice**'ların kullanımına göz atalım.



istediğimiz ürünne **Web Slice** olarak alıp tarayıcıma ekliyoruz

Eğer bir web sitesinde takip edebileceğiniz, kesip alabileceğiniz bir Web Slice varsa söz konusu bölümün fare ile üzerine geldiğinizde yeşil bir çerçeve ve Web Slice logosu karşımıza gelecektir. Bu görsellik tabi ki farklı web sitelerinde tasarımcılar tarafından farklı şekillerde değiştirilebilir. Web Slice logosunu gördüğünüz hemen o logoya tıklayarak web sitesinin söz konusu bölümünü kesip alabilirsiniz. Bu esnada Internet Explorer size nereden veri alacağını da gösteren bir uyarı kutucuğu gösterecektir.



Aldığımız Web Slice tarayıcımızın araç çubuğuuna yerleşti.

Kesip alma işlemi tamamlandığında artık Web Slice’ınız Internet Explorer’ın araç çubuğuunda yerini almış olacaktır. İstediğiniz zaman Web Slice’ın başlığına tıklayarak web sitesinin kestiğiniz bölümünü inceleyebilirsiniz. Bazen kestiğiniz yer ile araç çubuğuunda gördükleriniz bire bir birbirlerine benzeyebileceğ gibi tamamen farklı bir görSELLİKE de karşılaşabilirsiniz. Bu noktadaki karar ziyaret ettiğiniz site-nin tasarımcı ve programcıların ait.

Artık Internet Explorer sizin için söz konusu kesiği sürekli kontrol edecek ve herhangi bir değişiklik olduğunda Web Slice’ın başlığı araç çubuğuunda parlayarak kalın bir şekilde yazılmacaktır. Her Web Slice’ın kendi penceresinin sol altında üç adet düğme yer alabilir. Bunlardan sağa ok işaretli şeklinde olan sizi Web Slice’ın kesildiği sayfaya yön-lendirecektir.

Refresh (Yenile) düğmesinin logosuna benzer görsellikte olan düğme Web Slice’ın içeriğinin ana kaynaktan tekrar alınmasını sağlanırken, eğer varsa sarı bir ünlem işaretli ise Web Slice’ınızın ne kadar süre sonra kullanımdan kalkacağına dair bilgiyi sağlayacaktır.



Web Slice'ımıza özel ayarlar.

Bir Web Slice'in araç çubuğundaki başlığını sağ tıkladığınızda gelen menüden **Properties** komutunu verirseniz karşınıza söz konusu Web Slice ile ilgili düzenleyebileceğiniz ayarların bulunduğu özel bir pencere gelecektir. Bu pencere içerisinde gerekiyorsa Web Slice için güvenlik bilgilerini girebilir veya Web Slice içeriğinin ne kadar sürede bir internetteki kaynaktan yenilenmesi gerektiğini ayarlayabilirsiniz.

Herhangi bir Web Slice'ı silmek için yine sağ tuş ile Web Slice'in araç çubuğundaki başlığını tıklayarak gelen menüden **Delete** komutunu verebilirisiniz.

WEB'DE YENİ TRENDLER

Bir Hızlandırıcı (Accelerator)		VB ve ASP.NET ile
Nasıl Hazırlanır?	25	XML Üretmek
Son Adımlar...	30	PHP ile XML Üretmek
Artık Hazırız	32	Görsel Aramalar İçin
Görsel Arama Nasıl Hazırlanır?	32	JSON Veri Kaynağı
Görsel Arama ve		Web Dilimleri Nasıl Hazırlanır
Arama Sağlayıcıları İşbirliği	35	Çalışma Yapısı
Sunucu Tarafında		Display Source Değişikliği
Neler Yapmalı?	36	Farklı Web Dilimleri'ni
Sunucuda Gerçek Zamanlı		Birbirine Bağlamak
XML Üretime	39	Web Dilimi'ne Kaynak Olarak
C# ve ASP.NET ile		RSS Kullanmak
XML Üretmek	39	50

Internet Explorer 8 ile beraber gelen yeniliklere hızlı bir şekilde göz attıktan sonra son kullanıcının ötesinde biz üreticiler, yani yazılımcılar ve web tasarımcılar için neler nasıl işliyor sorusuna cevap vereceğimiz bu bölümde odaklanacağımız ana üç konu var. İlk olarak **Hızlandırıcıların** (*Accelerator*) yapısını inceleyerek hem tasarım hem de yazılım noktasında ne gibi bir altyapı hazırlamamız gereğine bakacağız. Sonrasında hemen Internet Explorer 8'in **Görsel Arama** (*Visual Search Suggestion*) özelliği için gerekli programatik yapının oluşturulmasına değinecek ve **Web Dilimleri**'nin (*Web Slices*) oluşumu ve hazırlanması ile ilgili detaylar ile bölümümüzü sonlandıracagız. Gelin hızlı bir şekilde örnekler ile bu yeni web trendlerini inceleyeceğim.

BİR HIZLANDIRICI (ACCELERATOR) NASIL HAZIRLANIR?

Accelerator'lar özünde birer XML dosyasıdır. Bu XML dosyalarının yazım standarı Microsoft tarafından belirlenmiş olup **OpenService-Description** olarak geçer. Bu standarda uygun yazılmış bir XML kodu ile tarayıcının hangi durumlarda ne gibi işlemler yapacağına dair hızlandırıcılar çerçevesinde kararları biz yazılımcılar olarak alırız. Böylece XML içerisinde tanımlı olan bir ayarlar Internet Explorer 8 içerisinde kullanıcıların karşısına birer **Hızlandırıcı** olarak çıkar.

```
<?xml version="1.0" encoding="utf-8" ?>
<openServiceDescription xmlns="http://www.micro-
```

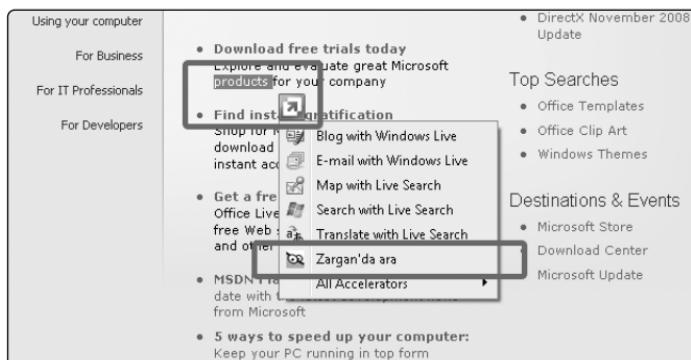
```

soft.com/schemas/openservicedescription/1.0">
<homepageUrl>http://www.zargan.com/</homepageUrl>
<display>
  <name>Zargan'da ara</name>
  <icon>http://www.zargan.com/favicon.ico</icon>
</display>
<activity category="search">
  <activityAction context="selection">
    <execute method="get" action=
    "http://www.zargan.com/sozluk.asp">
      <parameter name="Sozcuk" value="{selection}" />
    </execute>
  </activityAction>
</activity>
</openServiceDescription>

```

Zargan.com Türkçe-İngilizce çeviri sitesinde kullanılan hızlandırıcıya ait kod.

Yukarıda görmüş olduğunuz XML kodu şu anda **Zargan.com** İngilizce-Türkçe sözlük tarafından kullanılan Hızlandırıcı'nın XML kodudur. Özünde basit bir hızlandırıcının hazırlanması yolunda eğer hızlandırıcı **Ön İzleme** (Preview) ekranı içermiyorsa sunucu taraflı olarak ek bir kod veya işlem yapılması, sayfa hazırlanması gerekmekz. Bir önceki bölümde de incelediğimiz üzere hızlandırıcıların amacı bir siteden herhangi bir veriyi alıp baka bir siteye taşımaktır. Yukarıdaki örnekte herhangi bir sitedeki kelime doğrudan Zargan.com'daki çeviri sayfasına taşınarak olaşı çeviri işlemlerinin sonuçlarının yanında gösterilmesini sağlıyor.



Zargan.com hızlandırıcısını Microsoft sitesinde kullanırken.

Bir hızlandırıcı yazarken karar verilmesi gereken noktalardan biri hızlandırıcının hangi sekillerde kullanıcıya sonuç göstereceği. Var olan iki alternatiften biri kullanıcının tarayıcı içerisinde sağ tuşa tıklayınca erişebileceğи bir menüden tekrar bir tıklama ile hedef siteye yönlendirilmesi, bir diğer ise doğrudan söz konusu menü üzerinde kullanıcı gezerken yanında bir **Ön İzleme** penceresi gösterilmesi. Bu iki alternatif metodun her biri hızlandırıcının XML'i içerisinde birer action olarak tanımlanmak durumunda.

```
<activityAction context="selection" >
    <preview action="http://daron.yondem.com/aramasonuc.
    aspx" method=" get" >
        <parameter name="kelime" value="{selection}" />
    </preview>
    <execute action=" http://daron.yondem.com/arama-
    sonuc_ozel.aspx" method=" post" >
        <parameter name="kelime" value="{selection}" />
    </execute>
</activityAction>
```

Hemen yukarıdaki kod örneğinde kalın olarak yazılı noktalara dikkat edersek aslında farklı durumlarda hızlandırıcının nasıl işlemler yapacağını ayrı ayrı belirttiğimizi görebiliriz. Standart olarak **execute action** tüm hızlandırıcılarla bulunması zorunlu elementlerden biri olsa da her hızlandırıcı preview action ile bir ön izleme ekranı sağlamak zorunda değil. Hızlandırıcılardaki ön izleme ekranları aslında basit birer **IFRAME** olarak çalışırlar ve içlerinde yine hızlandırıcı içerisinde tanımlı web sayfası uygun parametreler ile çalıştırılır. Bir hızlandırıcının ön izleme ekranı kendi içinde Post veya Get işlemleri de yapabilir böylece belki de gösterilecek sonuçlar sayfalama yapılarak gösterilebilir. Maalesef ki gelen ön izleme ekranının boyutunu değiştirme şansınız olmayacaktır, bu denenle ön izleme ekranında sonuç gösterimi yapacak olan sayfaları özel olarak bu duruma uygun şekilde tasarlamak şart.

```
<activityAction context="selection" >
    .....
    .....
</activityAction>
```

Kodun en üstünde ve en altında **activityAction** adında bir XML tag'ı daha göreceksiniz. Bu tag içerisindeki **context** parametresi bir hızlandırıcıının çalışacağı durumu tanımıyor. Örneğin eğer **context** değeri **selection** olarak ayarlanmış ise söz konusu hızlandırıcı sadece tarayıcı içerisinde bir metin seçildiğinde devreye girecek ve hızlandırıcılar menüsünde yerini alacaktır. Oysa aynı **context** özelliğine **link** değerini verirseniz bu sefer de hızlandırıcınız tarayıcı içerisindeki herhangi bir linke sağ tıklandığında gelen menüde yer alacaktır. Tabi ki bu her iki durum için de ayrı ayrı **execute action** ve **preview action**'lar tanımlamak mümkün. Tek yapmanız gereken bu tanımlamaları uygun **activityAction** tag'ları arasına almak.

```
<execute action=" http://daron.yondem.com/arama-
sonuc_ozel.aspx" method=" post" >
<parameter name="kelime" value="{selection}" />
</execute>
```

Şimdi gelin detaylı olarak yukarıda kod ile tarayıcıya neler anlatmaya çalıştığımızı inceleyelim. İlk olarak iki önceki paragrafta da bahsettiğimiz üzere **execute action** tag'ını açarak hızlandırıcıya hızlandırıcılar menüsünde tıklandığında yapılacak işlemleri tanımlıyoruz. Sonuç itibarı ile ayrı bir siteye veri gönderme amacı ile yola çıktığımız için yapılacak işlem bir **POST** veya **GET** işlemi olabilir. Bunu hemen **method** özelliğinde belirtiyoruz. Eğer **POST** seçeneğini seçerseniz tüm parametreler **form** olarak gönderilecektir, eğer **GET** metodunu seçerseniz tüm parametreler adresre **query** olarak eklenecek gönderilecektir. Tabi ki tüm bu parametrelerin gönderileceği bir de adres bulunmak zorunda, işte o adresi de **action** özelliğine atıyoruz. Böylece artık hızlandırıcımız parametreleri nereye nasıl göndereceğini biliyor. Peki parametreler nelerdir?

```
<parameter name="kelime" value="{selection}" />
```

Yukarıdaki şekilde herhangi bir **action** için istediğiniz sayıda parametre tanımlayabilirsiniz. Her parametrenin bir adı ve bir de değeri bulunmak zorunda. Bu değeri **value** özelliği olarak tanımlarken isterkeniz elle sabit değerler girebileceğiniz gibi tarayıcı tarafında çalışma zamanında belirlenecek dinamik değerleri de karşı sunucuya gönderebilirsiniz.

Aslında bizim esas amacımız da kullanıcının sayfada seçtiği metin veya linkin bilgileri gibi dinamik bilgileri anında kendi sunucumuzdaki uygulama gönderebilme. Bu çerçevede örnekte olduğu üzere `{selection}` gibi değişken parametreler kullanarak kendi parametrelerimizin değerlerini Internet Explorer 8'in belirlemesini sağlayabiliyoruz. Kullanabilecek farklı parametreleri ve alacağı değerlerin anlamını aşağıdaki tabloda bulabilirsiniz.

Adı	Açıklaması
<code>documentDomain</code>	İçerisinde bulunan sayfanın alan adı.
<code>documentTitle</code>	İçerisinde bulunan sayfanın HTML başlık (<code>title</code>) bilgisi.
<code>documentUrl</code>	İçerisinde bulunan sayfanın tam adresi.
<code>link</code>	Eğer kullanıcı bir linkte sağ tıklaşmış ise bu linkin adresi.
<code>linkDomain</code>	Eğer kullanıcı bir linkte sağ tıklaşmış ise bu linkin hedeflediği ana alan adı.
<code>linkRel</code>	Eğer kullanıcı bir linkte sağ tıklaşmış ise bu linkin rel microformat değeri.
<code>linkText</code>	Eğer kullanıcı bir linkte sağ tıklaşmış ise bu linkin içerisindeki yazı.
<code>selection</code>	Kullanıcının tarayıcı sayfasında seçmiş olduğu metin.

Tüm bu işlemleri tamamladıktan sonra aynı parametreleri kullanarak bir ön izleme ekranı da hazırlayabilirsiniz.

```
<preview action="http://daron.yondem.com/aramasonuc.
aspx" method=" get" >
  <parameter name="kelime" value="{selection}" />
</preview>
```

Üstte görebileceğiniz kod içerisindeki tek farklılık ana tag'ın `execute action` yerine `preview action` olarak tanımlanmış olası. Ayrıca tabii ki bu sefer hedef adres olarak verdığımız yerdeki web sayfamızın ta-

şarımı da normalden farklı olarak ön izleme penceresi içerisinde siğabılacak şekilde tasarlanmış olmalı.



SUNUCU TARAFINDA NELER YAPMALI?

Hızlandırıcıların sunucu taraflı programlama teknolojileri veya işletim sistemlerinden tamamen bağımsız olarak istemcide çalışırlar. Önemli olan tek şey hızlandırıcıya ait tanımların yapıldığı XML dosyasının bir şekilde web sunucusu tarafından site ile aynı port üzerinden yayınlanabiliyor olmalıdır.

Bir hızlandırıcı tarafından siteye gönderilen parametreler programının tercihine göre standart POST veya GET işlemleri ile gönderilir ve bu verileri alabilmek için sunucu tarafında yine standart prosedürler uygulanır.

[ASP.NET C#]

```
//GET ile gelen veriyi almak için.  
Request.QueryString["sozcuk"];  
//POST ile gelen veriyi almak için.  
Request.Form["sozcuk");
```

[ASP.NET VB]

```
'GET ile gelen veriyi almak için.  
Request.QueryString("sozcuk")  
'POST ile gelen veriyi almak için.  
Request.Form("sozcuk")
```

[PHP]

```
//GET ile gelen veriyi almak için.  
$_GET['sozcuk'];  
//POST ile gelen veriyi almak için.  
$_POST['sozcuk'];
```

SON ADIMLAR...

Hızlandırıcımızla ilgili tüm işlevselliklere karar verdığimize göre son olarak hızlandırıcımızın tanımları ve kimliği ile ilgili ayarları da yaparsak yayına hazır hale getirebiliriz.

```
<?xml version="1.0" encoding="utf-8" ?>  
<openServiceDescription xmlns="http://www.microsoft.  
com/schemas/  
openservicedescription/1.0">
```

```

<homepageUrl>http://daron.yondem.com/</homepageUrl>
<display>
    <name>Deneme ile çevir</name>
    <icon>http://daron.yondem.com/favicon.ico</icon>
</display>
<activity category="translate">
    <activityAction context="selection" >
        <preview action="http://daron.yondem.com/cevir.
asp">
            <parameter name="Sozcuk" value="{selection}" />
        </preview>
        <execute action="http://daron.yondem.com/cevir.
asp">
            <parameter name=" Sozcuk " value="{selection}" />
        </execute>
    </activityAction>
</activity>
</openServiceDescription>

```

İlk olarak tanımladığımız tüm `activityAction`'ları ayrı bir `activity` tag'ı içerisinde alarak bu tag'a da `category` değeri vermemiz gerekiyor. Bu noktadaki `category` elementine vereceğiniz değer için önceden hazırlanmış bir liste yok. `category` değeri için Microsoft tarafından belirlenen kural söz konusu değerin İngilizce bir fil içermesi gerektiği ve kısmen emir kipinde bulunması şeklinde tanımlanabilir. Bu değerler tarayıcı tarafından sadece gruplama amacıyla kullanılıyor ve aynı değerlere sahip hızlandırıcıları hızlandırıcı menüsünde beraber gösteriliyor.

```

<homepageUrl>http://daron.yondem.com/</homepageUrl>
<display>
    <name>Deneme ile çevir</name>
    <icon>http://daron.yondem.com/favicon.ico</icon>
</display>

```

Son olarak `openServiceDescription` dosyasının başında hızlandırıcıımızla ilgili bilgilerini de giriyoruz. Hemen ufak bir uyarıda bulunmak istiyorum `homePageUrl` olsun `icon` adresi olsun tüm bu ad-

reslerin `action` adresleriniz ile aynı alan adını hedeflemesi şart. Aksı halde hızlandırıcınızın çalışması ile ilgili sorunlar yaşayabilirsiniz.

ARTIK HAZIRIZ

Hızlandırıcımızı yayına almaya hazırız. Eğer hızlandırıcının kullanıcıları yönlendireceği sayfalarla ilgili tasarımsal veya programatik değişiklikler yaptıysanız ilk olarak söz konusu kısımları sunucunuza yüklemenizde fayda var. Sonrasında hızlandırıcınızı sunucuya yüklemek için sadece `openServiceDescription` XML dosyasının web sunucusu üzerinden ulaşılabilir bir konu koyma yeterli olacaktır. Örneğin web sitenizin adresinin `deneme.com` olduğunu düşünürsek `deneme.com/hizlandirici.xml` şeklinde hızlandırıcınızın XML dosyasına ulaşabiliyor olması yükleme işlemini tamamlamış olmak için yeterli.

Peki kullanıcıların bu hızlandıryı nasıl bulacaklar? Nereye tıklayarak bilgisayarlarına yükleyecekler? İşte bu noktada karar sizin. Aşağıdaki JScript komutunu çağırarak istediğiniz bir hızlandırıcıyı istemci taraflına yüklenmek üzere gönderebilirsiniz.

```
<div>
    <input id="Button1" type="button"
        value="IE 8 Hızlandırıcımızı Yükleyin"
        onclick="window.external.AddService('hizlandirici.
xml');"
    </div>
```

`AddService` metodu parametre olarak hızlandırıcı tanımlarınızın bulunduğu XML dosyasının yerini alıyor ve istemci tarafındaki yükleme işlemlerine dair onay mekanizmasının çalışmasını sağlıyor. Artık bu noktadan sonra tüm kontrol kullanıcıda, hızlandırıcınız ziyaretçiye ulaştı.

GÖRSEL ARAMA NASIL HAZIRLANIR?

Görsel Arama sistemleri hazırlamadan önce hali hazırda bir **Arama Sağlayıcı** dosyanızın bulunması gerekiyor. Görsel Arama sistemi eski Arama Sağlayıcı sisteminin üzerine inşa edilmiş durumda olduğu için önce bir Arama Sağlayıcı nasıl oluşturulacağını inceleyelim.

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSearchDescription xmlns="http://a9.com/-/spec/
opensearch/1.1/">
```

```
<ShortName>Deneme Arama Motoru</ShortName>
<Url type="text/html" template="http://daron.yon-
dem.com/?aranan=
{searchTerms}" />
<Image height="16" width="16" type="image/icon">
    http://daron.yondem.com/favicon.ico</Image>
</OpenSearchDescription>
```

OpenSearchDescription standartı çerçevesinde hazırlanan arama sağlayıcılar çok sayıda tarayıcı tarafından destekleniyor. Yukarıdaki şekilde tanımlanan bir arama sağlayıcı tarayıcının arama kutucuğunda yazılan herhangi bir verinin nereye nasıl gönderileceğini tanımlıyor. İlk aşamada XML dosyası içerisinde arama sağlayıcının adı **ShortName** tag'ları arasında yazıldıktan sonra arama işleminin nereye yönlendirileceği ise bir **Url** tagi içerisinde belirtiliyor.

```
<Url type="text/html" template="http://daron.yondem.
com/?aranan=
{searchTerms}" />
```

Yukarıda gördüğünüz kod içerisinde yönlendirilen yerdeki veri kaynağı tipinin **text/html** olduğu belirtildikten sonra yönlendirilecek olan adres de **template** değerine aktarılıyor. Örneğimizdeki adres şablonunu incelerseniz en sonundaki **{searchTerms}** kısmı dikkatinizi çekecektir. İşte tam da bu noktada, adres içerisinde **{searchTerms}** 'in yazılı olduğu yere kullanıcının tarayıcı içerisinde yazmış olduğu aranacak kelimeler yerleştirilecek ve söz konusu adres bu şekilde çalıştırılacaktır.

Örneğin kullanıcı eğer tarayıcısında aranacak kelime olarak “asp.net” yazmış ise çalıştırılacak olan sayfa <http://daron.yondem.com/?aranan=asp.net> şeklinde olacaktır. Böylece sunucu tarafında uygulama ile adresin alınabilecek bu parametre kullanılarak gerekli arama işlemleri yapılabilir ve sonuç doğrudan kullanıcuya gösterilebilir.

Son olarak arama sağlayıcısına ait tarayıcı içerisinde gösterilecek bir ikon dosyası da aşağıdaki şekilde openSearchDescription dosyasına eklenir.

```
<Image height="16" width="16" type="image/icon">
    http://daron.yondem.com/favicon.ico</Image>
```

Tüm bu işlemleri yaparak kendi siteniz için bir arama sağlayıcı geliştirdikten sonra tabi ki bu sağlayıcının kullanıcılar tarafından bulunabilmesi ve yüklenebilmesi için gerekli değişiklikleri de yapmamız gerekiyor. Örneğin hemen bir yükleme düğmesi hazırlayarak sitemize yerleştirmek istersek arama sağlayıcısına ait yükleme ekranlarını açabilmek için aşağıdaki JScript komutunu kullanabiliriz.

```
<a href="#" onclick="window.external.AddSearchProvider
('http://
    daron.yondem.com/aramasaglayici.xml')"> Arama Sağ-
layıcımızı Yükleyin</a>
```

Hazırladığımız arama sağlayıcıya ait XML kodlarını sunucuya **aramasaglayici.xml** dosyası olarak yükledikten sonra ana sayfamiza yukarıdaki gibi bir link ekleyerek AddSearchProvider metodunu çağırabiliriz. Söz konusu metod parametre olarak arama sağlayıcıya ait ayarların bulunduğu XML dosyasının yolunu alıyor ve sonrasında yükleme işlemlerini istemci tarafında başlatıyor.

Bir diğer alternatif ise tarayıcıların kendi kendilerin sayfa içerisindeki arama sağlayıcıları bulabilmesi için yapılabilecek linkleme ayarıdır. Böylece kullanıcıların sayfadaki yükleme düğmelerini bulaması veya sayfada bir yükleme düğmesi olmasa da tarayıcı söz konusu sayfada bir arama sağlayıcı olduğunu anlayarak kendi arayüzünde gerekli mesajları kullanıcıya gösterecektir.

```
<link title="Daron Yöndem Arama" rel="search"
type="application/
opensearchdescription+xml" href="http://daron.yon-
de.com/ aramasaglayici.xml" />
```

Yukarıdaki linki HTML sayfasının Header bölümüne yerleştirdiğinizde artık tarayıcıların sizin sitenizde arama sağlayıcından haberدار olacaklar. Link kodu içerisinde title değeri arama sağlayıcınızın ismini tanımlarken rel ve type değerleri tam olarak önekteki şekli ile değiştirilmeden yazılmalıdır. Tarayıcı ancak bu şekilde söz konusu linkin bir arama sağlayıcıya ait olduğunu anlayabilir. Son olarak href özelliğine ise arama sağlayıcıya ait XML dosyasının tam yolu verilir.

GÖRSEL ARAMA VE ARAMA SAĞLAYICILARIN İŞBİRLİĞİ

Artık elimizde hazır bir arama sağlayıcı bulunduğuna göre bu arama sağlayıcıya görsel arama işlevselligi eklemenin zamanı geldi. Aşağıda nedirtv.com sitesinde kullanılan ve görsel arama işlevselligi de sunan arama sağlayıcının XML **OpenSearchDescription** dosyasını inceleyebilirsiniz.

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSearchDescription xmlns="http://a9.com/-/
/spec/opensearch/1.1/">
  <ShortName>NedirTV?com</ShortName>
  <Url type="text/html" template="http://www.ne-
dirtv.com/Arama.aspx?A={searchTerms}" />
  <Url type="application/x-suggestions+xml" template=
"http://www.nedirtv.com/RssVideo.ashx?SS={search-
Terms}" />
  <Image height="16" width="16" type="image/
icon">http://www.nedirtv.com/favicon.ico</Image>
</OpenSearchDescription>
```

Bir önceki bölümde hazırladığımız arama sağlayıcılar ile yukarıdaki kod arasında farkı incelerken aslında sadece yeni bir Url tagının açıldığını ve bu sefer farklı olarak **type** özelliğin **text/html** değil de **app-
lication/x-suggestions+xml** olduğunu görebilirsiniz. Söz konusu ta-
nimlama ile artık arama sağlayıcımız kendi içerisinde bir de görsel
arama veri kaynağı tanımlıyor. Böylece adresi verilmiş olan veri kay-
nağından gerekli sonuçlar tarayıcı tarafından otomatik olarak çekile-
rek kullanıcıya gösterilebilecek. Tüm bu işlevsellik içerisinde bizim
tek yapmamız gereken ek olarak gerekli veri kaynağını doğru format-
ta sağlayacak olan sunucu tarafı mekanizmayı hazırlamak.

```
<Url type="application/x-suggestions+xml"
template="http://www.nedirtv.com/RssVideo.ashx?SS={se-
archTerms}" />
```

Kodumuzun görsel arama ile ilgili kısmını incelersek **Url** tag'ının ti-
pinde veri kaynağı olarak XML formatının belirtildiğini görebilirsi-
niz. Veri kaynağı adresi tanımlanırken de yine **{searchTerms}** para-

metresi ile kullanıcının yazdığı arama hedef adresе eklенerek sunucuya gönderilebilecek. Böylece sunucu tarafından bu parametre alınabilir ve uygun XML veri kaynağı oluşturulabilir.

```
<Url type="application/x-suggestions+json" template="http://www.nedirtv.com/RssVideo.ashx?SS={searchTerms}
```

Eğer isterseniz veri kaynağı olarak **JSON** formatını da kullanabilirsiniz. **JSON** formatında maalesef ki **Görsel Arama** sistemindeki arama sonuçlarında resim gösterimi gibi özellikler kullanılamıyor. O nedenle çoğunlukla Görsel Arama altyapılarında **XML** formatı tercih ediliyor.

SUNUCU TARAFINDA NELER YAPMALI?

Görsel arama sistemi ile ilgili yapılması gereken çalışmanın büyük kısmı sunucu tarafında yapılıyor. Bunun nedeni aslında çok basit. Open-SearchDescription dosyamızda tanımlamış olduğumuz veri kaynağını tarayıcının anlayabileceğim bir formatta sağlamamız şart. Aksi halde kullanıcıya Görsel Arama sonuçları göstermemiz mümkün olmayacaktır. Peki Internet Explorer 8 bizden nasıl bir formatta XML istiyor?

```
<SearchSuggestion>
    <Query>aranan</Query>
    <Section title="İlk Bölüm">
        <Item>
            <Text>Sonuç 1</Text>
            <Description>Açıklama</Description>
            <Url>http://daron.yondem.com?id=3</Url>
        </Item>
        <Separator title="Diğerleri"/>
        <Item>
            <Text>Sonuç 2</Text>
            <Description>Açıklama</Description>
            <Url>http://daron.yondem.com?id=6</Url>
        </Item>
    </Section>
</SearchSuggestion>
```

Yukarıdaki kod içerisinde basit bir arama sonucunda Internet Explorer'a döndürmemiz gereken XML veri kaynağının yapısını inceleyebilirsiniz. Satır satır XML kodlarının incelemeden önce bu veri kayna-

ğının sadece iki sonuç döndüren örnek bir veri kaynağı olduğunu belirtmekte fayda var. Normal şartlarda on adet sonuç gönderebilirsiniz.

```
<Query>aranan</Query>
```

Görsel arama için istemciye gönderdiğimiz XML kodunun ilk satırında **Query** tagı bulunuyor. Bu tag içerisinde sunucu tarafına aranmak üzere yollanan kelimeleri yazdırıyoruz. Böylece Internet Explorer kendisine gelen veri ile talep ettiği arama verisinin aynı olup olmadığını kontrol edebiliyor.

```
<Section title="İlk Bölüm">
```

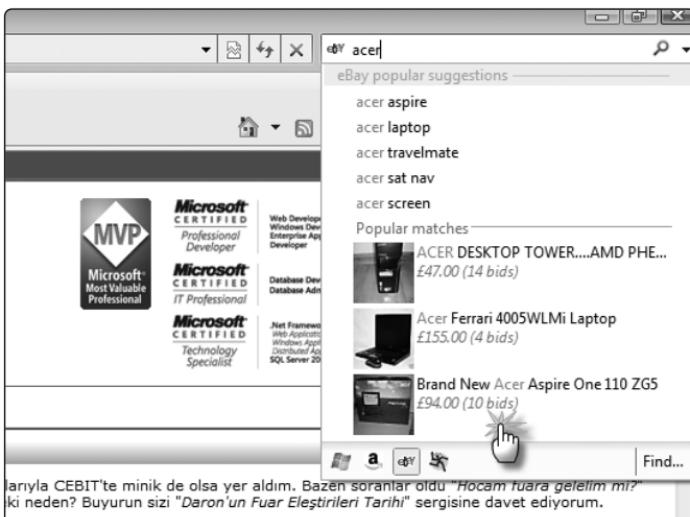
Section tag'ları arama sonuçları gösterilirken sonuçları gruplamak amacıyla kullanılabilir. Her section'un (bölüm) bir de **title** bilgisi, yani başlığı bulunuyor. Örnek bir kullanım olarak herhangi bir arama'nın sonucunda farklı kategorilerinde ürünlerinin getirildiği bir e-ticaret sitesi düşünülebilir.

```
<Item>
    <Text>Sonuç 1</Text>
    <Description>Açıklama</Description>
    <Url>http://daron.yondem.com?id=3</Url>
</Item>
```

İşte geldik en can alıcı noktaya. Görsel arama işlemi esnasında gösterilecek olan her bir sonucu tanımlamak için **Item** tag'ları ile gerekli verileri XML içerisinde yazdırımız şart. Her bir **Item** tagının **Text**, **Description** ve **Url** elementlerini içermesi gerekiyor. **Text** ve **Description** sonuç listesinde gösterilirken **Url** ise söz konusu sonuca tıklandığında kullanıcının yönlendirileceği adresi tanımlıyor. Eğer arama sonuçlarında her bir öğe ile ilgili bir de ufak resim göstermek istiyorsanız **Item** tagları arasına **Image** elementi de yerleştirmelisiniz.

```
<Item>
    <Text>Sonuç 1</Text>
    <Description>Açıklama</Description>
    <Url>http://daron.yondem.com?id=3</Url>
    <Image Source="http://daron.yondem.com/
resim.jpg
        alt="Bu bir resimdir" width="70"></Image>
</Item>
```

Yukarıdaki kod örneğinde `Item` elementi içerisinde bir de `Image` elementi bulunuyor. `Image` elementi doğrudan `Source` olarak bir resim dosyasının tam yolunu içeriyor. Böylece Internet Explorer eğer söz konusu resme erişebilirse doğrudan görsel arama sonuçlarını gösterirken resme de sol tarafta yer verecektir.



Görsel Arama sonuçları ve resimli gösterim şekli.

Eğer `section` tagları, yani bölümler içerisinde ayrıca sonuçları birbirinden ayırmak isterseniz. `Separator` elementini kullanabilirsiniz. `Section` elementindeki gibi `Separator`'lara da `title` (`başlık`) değeri verilebiliyor.

```
<Separator title="Diğerleri"/>
```

Bahsettiğimiz standartlara uygun şekilde gerekli veri tabanı sorgularını yapan ve uyumlu XML çıktısı veren sunucu taraflı kodları yazdıktan sonra arama sağlayıcı dosyanızı eskisi gibi aynı JScript kodları ile sitenizin ziyaretçilerine sunabilirsiniz. Eğer hedef tarayıcıların görsel arama sistemini destekliyorsa otomatik olarak sonuçlar gösterilecektir, aksi halde arama sağlayıcı sistemi eski tarayıcılarında eski hali ile çalışmaya devam eder.

SUNUCUDA GERÇEK ZAMANLI XML ÜRETİMİ

Görsel arama sistemlerini kullanabilmek için en önemli nokta sunucu tarafında uygun XML çıktısını oluşturabilmek. Bu çerçevede kitabımızın bu bölümünde sunucu tarafında veri kaynağı olarak basit bir **Array** (*Dizi*) kullanarak gerekli XML kodunu nasıl yaratabileceğimiizi inceleyeceğiz.

C# VE ASP.NET İLE XML ÜRETMEK

ASP.NET tarafında özellikle .NET Framework 3.5 ile beraber LINQ'in de yardımımıza yetişmesi XML üretimini ciddi anlamda kolaylaştırıyor. Biz de kodlarımızda LINQ'ten faydalananarak hızlı bir şekilde elimizde bulunan Generic bir List içerisinde veriyi XML'e dönüştüreceğiz.

XML kodunu istemciye doğrudan göndereceğimiz için HTML yaratıcı bir Web Form yerine bize daha esnek bir yapı sunan Generic Handler'ları kullanmadıkça fayda var. İlk olarak örneğimizde kullanacağımız veri kaynağını yaratalım.

```
public class Sonuc
{
    public string Text { get; set; }
    public string Description { get; set; }
    public string Url { get; set; }
    public string Photo { get; set; }
}
```

Veri kaynağımızda her bir sonucu temsil edecek olan sınıfımızı yukarıdaki şekilde tanımladıktan sonra **Generic Handler** çalıştırıldığında hem örnek amaçlı olarak bir veri yaratacağız hem de o veri üzerinden sonrasında da XML üreteceğiz.

```
System.Collections.Generic.List<Sonuc> Veri =
new System.Collections.Generic.List<Sonuc>();
for (int i = 0; i < 10; i++)
{
    Veri.Add(new Sonuc())
}
```

```

        Text="Sonuç" + i.ToString(),
        Description="Açıklama" +
        i.ToString(),
        Photo="Foto" + i.ToString() +
        ".jpg",
        Url="adres.aspx?ID=" + i.ToSt-
ring()
    });
}

```

Veri kaynağımızı yaratma işlemini tamamladığımızda göre artık elimizde yer alan Veri adındaki Generic List üzerinden yola çıkarak uygun XML kodunu yaratabiliriz. Bunun için olarak sayfanın çıktısına ait **Content Type**'ı **Text/XML** olarak değiştirelim.

```
context.Response.ContentType = "text/xml";
```

Bir sonraki adımda XML dokümanımızın ana elementlerini yaratalım ve içerisinde veri, yani **Item** nesneleri eklenebilecek hale getirelim.

```

XDocument XMLFeed = new XDocument(new
XDeclaration("1.0", "utf-8", "yes"));
XMLFeed.Add(new XElement("SearchSuggestion",
    new XElement("Query", context.Request.
QueryString["aranacak"].ToString()),
    new XElement("Section",
        new XElement("Separator",
            new XAttribute("title", "Bölüm
1"))));

```

Şimdi sıra geldi elimizdeki Generic List içerisinde gezerek yukarıda yaratmış olduğumuz doküman içerisinde **SearchSuggestion** elemanı içindeki **Section** elementinin içine uygun **Item** elementlerini yerleştirmeye.

```

for (int i = 0; i < Veri.Count; i++)
{
    XMLFeed.Element("SearchSuggestion").
Element("Section").Add(
    new XElement("Item",

```

```

        new XElement("Text", Veri[i].Text),
        new XElement("Image",
            new XAttribute("source", Veri[i].Url),
            new XAttribute("alt", "Bir resim"),
            new XAttribute("width", "75"),
            new XAttribute("height", "50")),
        new XElement("Description", Veri[i].
Description),
        new XElement("Url", Veri[i].Url)));
    }
}

```

Gördüğünüz gibi yarattığımız yeni **XElement** ve içerisinde **XAttribute** nesnelerine veri kaynağı olarak For döngüsü içerisinde sürekli Veri adındaki **Generic List**'i kullanıyoruz. Böylece istediğimiz XML DOM'u yaratabiliyoruz. Son olarak tüm bu XML'i istemciye göndereilmek üzere sayfaya yazdırımız gerekiyor.

```
context.Response.Write(XMLFeed.ToString());
```

Artık **Generic Handler** dosyamız kendisine gelecek olan **aranacak querystring** parametresine göre farklı XML'ler döndürmek için hazır durumda. Tek yapmanız gereken örneğimizdeki Veri adındaki **List**'i farklı veri kaynakları ile değiştirmek.

VB VE ASP.NET İLE XML ÜRETMEK

Bir önceki bölümde C# ile ASP.NET tarafından nasıl XML çıktısı üretebileceğimizi incelemiştik. C# ile VB arasında çok büyük farklar bulunmasa da XML konusunda VB 9.0 ile beraber gelen XML Literals yapısını kullanırsak XML yaratma işlemi çok daha farklı bir çehreye bürünebiliyor. O nedenle gelin C# ile yaptığımız örneği bir de VB ile yaparak nasıl ilerleyebileceğimizi inceleyelim.

```

Public Class Sonuc

    Private PText As String
    Public Property Text() As String
        Get
            Return PText
        End Get
        Set(ByVal value As String)
            PText = value
        End Set
    End Property
End Class

```

```
        End Set
    End Property

    Private PDescription As String
    Public Property Description() As String
        Get
            Return PDescription
        End Get
        Set(ByVal value As String)
            PDescription = value
        End Set
    End Property

    Private PUUrl As String
    Public Property Url() As String
        Get
            Return PUUrl
        End Get
        Set(ByVal value As String)
            PUUrl = value
        End Set
    End Property

    Private PPhoto As String
    Public Property Photo() As String
        Get
            Return PPhoto
        End Get
        Set(ByVal value As String)
            PPhoto = value
        End Set
    End Property

End Class
```

İlk aşamada yukarıdaki şekli ile tüm veri kaynağımızı üretirken kullanacağımız Sonuc adındaki sınıfımızı tanımlıyoruz.

Bir sonraki adımda ise deneme amaçlı olarak kullanacağımız veri kaynağımızı üreteceğiz.

```

    Dim Veri As New System.Collections.Generic.List
    (Of Sonuc)
        For index As Integer = 1 To 10
            Veri.Add(New Sonuc With {.Text = "Sonuç"
    & index, _
                                .Description =
    "Açıklama" & index, _
                                .Photo = "Photo"
    & index & ".jpg", _
                                .Url = "adres.
    aspx?ID=" & index})
        Next

```

Yukarıdaki kodumuz ile deneme amaçlı olarak kullanacağımız veri kaynağımızı da yarattığımıza göre artık sıra geldi bu veri kaynağı üzerinden gerekli XML'i yaratmaya. İlk olarak sayfanın **Content Type**'ını doğru şekilde ayarlayalım.

```
Context.Response.ContentType = "text/xml"
```

Son olarak XML yaratma işlemlerine başlayabiliriz. Daha önce de bahsettiğim üzere VB artık dil ile entegre XML yazımını destekliyor. O nedenle C#'da olduğu gibi XElement vs nesneler ile uğraşmaya gerek kalmadan doğrudan istediğimiz XML formatını dile dahil edebiliyoruz.

```

Dim XMLFeed = <?xml version="1.0" encoding="utf-8"?>
    <SearchSuggestion xmlns="http://opensearch.org/
searchsuggest2" version="1">
        <Query><%= Context.Request.QueryString("ara-
nacak") %></Query>
        <Section title="Bölüm 1">
            <%= From inc In Veri Select <Item>
                <Text><%= inc.Text %></Text>
                <Description><%= inc.Description
%></Description>
                <Url><%= inc.Url %></Url>
                <Image source=<%= inc.Photo %>
alt="Bilgi" width="75" height="75"/>
            </Item> %

```

```
</Section>
</SearchSuggestion>
```

Kod içerisinde de görebileceğiniz üzere doğrudan XML kodları bir değişkene eşitlenmiş durumda. Bu kodlar .NET içerisinde VB derleyicisi tarafından derleme zamanında C#’da olduğu gibi XElement nesnelerine çevrilecektir. O nedenle herhangi bir performans farklılığı olmayaçaktır. Biz XML kodumuzu değişkenimize eşitlerken bir de LINQ sorusu kullandık. Böylece Veri adındaki **Generic List** içerisinde kaç tane kayıt varsa uygun şekilde Item XML nesnelerini de yaratmış olduk.

Artık XML kodumuzu sayfaya yazdırabiliriz.

```
Context.Response.Write(XMLFeed.ToString())
```

PHP İLE XML ÜRETMEK

Windows Server 2008 ve IIS 7 ile beraber gelen **FastCGI** ile artık Windows sunucu üzerinde PHP kullanımı hatta ASP.NET ile entegre ortak **Forms Authentication** kullanımları ve Session paylaşımı gibi özelliklerin hayatımıza girdiği bu günlerde sunucu tarafında PHP kullanmanız halinde de tabii ki GörSEL Arama için uygun XML kodlarını üretebilirsiniz. Bunun için örneğimizde herkesin sunucusunda **PHP DOM**’un kurulu olup olmadığından emin olmadığımız için PHP DOM kullanmak yerine standart String işlemleri ile devam edeceğiz.

```
header('Content-type: text/xml');
```

İlk olarak yukarıdaki kodumuz ile sayfamızın Content Type ayarını yapalım. Böylece tarayıcı kendisine bir XML gönderildiğini rahatlıkla anlayabilecektir.

```
$items = array();
$item[] = array('text' => 'Sonuç 1',
               'image' => array
('source' => 'http://x.com/resim1.jpg',
             'alt' => 'bir resim',
             'width' => 75,
             'height' => 60),
               'description' =>
'Description 1!',
               'url' =>
'http://daron.yondem.com');
```

```

$items[] = array('text' => 'Sonuç 2',
                 'image' => array
('source' => 'http://x.com/resim2.jpg',
                 'alt' => 'baska resim',
                 'width' => 75,
                 'height' => 60),
                 'description' =>
'Description 2!',
                 'url' =>
'http://daron.yondem.com');

$items[] = array('text' => 'Sonuç 3',
                 'image' => array
('source' => 'http://x.com/resim3.jpg',
                 'alt' => 'baska bir resim daha',
                 'width' => 75,
                 'height' => 60),
                 'description' =>
'Description 3!',
                 'url' =>
'http://daron.yondem.com');

```

Sonrasında örneğimizde kullanmak üzere yukarıdaki gibi bir Array tanımlayalım. Böylece bir sonraki adımda bu dizi içerisindeki verileri kullanarak uygun XML çıktısını yaratacağız.

```

$output = "<SearchSuggestion>\n";
$output .= "<Section>\n";
$output .= "<Query>asp</Query>\n";
$output .= "\t<Separator title=\"Bölüm 1\" />\n";

foreach( $items AS $item ) {

    $output .= "\t\t<Item>\n";
    $output .= "\t\t\t<Text>". $item['text'] . "</Text>\n";
    $output .= "\t\t\t<Image source=\"". $item['image']
['source'] ."\"\n";
    $output .= "\t\t\t\t alt = \"". $item['image']['alt']
."\"\n";
    $output .= "\t\t\t\t width = \"". $item['image']
['width'] ."\"\n";

```

```

    $output .= "\t\t\t height = '\"'. $item['image']
['height'] ."\n";
    $output .= "\t\t\t /> \n";
    $output .= "\t\t<Description>". $item["description"]
."</Description> \n";
    $output .= "\t\t<Url>". $item["url"] ."</Url>\n";
    $output .= "\t\t</Item>\n";

}

$output .= "</Section>\n";
$output .= "</SearchSuggestion>\n";

echo $output;

```

Yukarıdaki kod içerisinde de görebileceğiniz üzere standart **string** birlleştirme işlemleri ile XML kodumuzu yaratıyoruz. XML’imizin başlangıç ve sonunu belirledikten sonra orta kısmında da bir **For** döngüsü ile elimizdeki diziyi gezerek uygun **Item** XML elementlerini yaratıyoruz. Bu şekilde yaratılan bir XML kodu da **Görsel Arama** alt-yapıları tarafından rahatlıkla kullanılabilir.

GÖRSEL ARAMALAR İÇİN JSON VERİ KAYNAĞI

Görsel aramalar için bu bölüme kadar veri kaynağı olarak XML kullandık. Bunun nedeni bir alternatif olarak **JSON** formatı gelse de JSON içerisinde hem **Image** nesneleri hem de **Separator** kullanamıyor olmamız. Eğer tüm bu dezavantajlarına rağmen JSON kullanmayı düşünüyorsanız oluşturmanız gereken JSON veri kaynağının bir örneğini aşağıda inceleyebilirsiniz.

```
[ "aranan",
[ "sonuc1", "sonuc2", "sonuc3"],
[ "aciklama 1", "aciklama 2", "aciklama3"],
[ "http://daron.yondem.com/1", " http://daron.yondem.
com/2", " http://daron.yondem.com/3" ]]
```

Gördüğünüz gibi JSON içerisinde toplam üç sonuç döndüren kodumuzun başında yine Internet Explorer tarafından parametre olarak gönderilen arama kelimeleri bulunuyor.

WEB DİLİMLERİ NASIL HAZIRLANIR

Web dilimleri kullanıcıların sitelerimizi takip edebilmeleri açısından ilginç işlevsellikler sağlayabiliyor ve bu işlemi çok kolaylaştırıyorlar. Bu çerçevede web dilimlerinin oluşturulması ile ilgili teknik açıdan birkaç farklı yol var. Bu bölümümüzde her farklı yolu avantaj ve dezavantajlarını da değerlendirerek inceleyeceğiz.

```
<div class="hslice" id="Urun1">
  <h1 class="entry-title">
    Bu bir ürün!
  </h1>
  <div class="entry-content">
    <p>
      Ürün bilgisi burada.
    </p>
  </div>
</div>
```

Yukarıda gördüğünüz basit HTML kodu herhangi bir web sayfasında Web Slice'ı tanımlıyor. Web sayfaları içerisinde Web Slice'lar tama-men microformatlar ile tanımlanır, normal şartlarda teknik olarak bir işlevselliği olsun veya olmasın kodların tarayıcı için farklı anlamları vardır. Örneğin `class` özelliğine `hslice` alan bir HTML elementi tarayıcı için bir Web Slice tanımı anlamına gelir. Normalde bizler `class` değerlerine CSS sınıflarını atarız ve bu özelliği tasarımsal ayarlamalar için kullanırız, oysa Internet Explorer 8 için `hslice` adındaki `class` değerinin çok daha farklı bir önemi var. Tabi diğer yandan eğer isterse web tasarımcı veya programcısı `hslice` adında bir CSS sınıfı yaratarak bu durumdan görsel anlamda da faydalabilir.

Internet Explorer 8 için bir Web Slice tanımlarken ilk olarak web diliminin ana gövdesine `hslice` ismi ile bir `class` değeri vermelisiniz. Sonrasında bu Web Dilimine ait gövdeyi teşkil edecek olan HTML elementinin kesinlikle sayfa içerisinde tekil bir `ID` değeri olmalıdır. Buradaki ID değerinin önemi çok büyük çünkü Internet Explorer sayfa içerisindeki Web Dilimleri'ni bu ID üzerinden tanıracaktır. İleriki adımlarda da bahsedeceğimiz üzere kullanıcıların bu web dilimini kesip tarayıcılarına aldıktan sonra eğer söz konusu web diliminin ID bilgisini değiştirirseniz daha önce kesip alınan dilimler ile bu dilim arasındaki ilişki kopacaktır ve kullanıcılar artık kesmiş oldukları dilimle ilgili güncellemeleri takip edemeyecektir.

Son olarak basit bir Web Slice içerisinde bir de `entry-content` ve `entry-title` bulunması gereklidir. Yine farklı HTML elementlerine verilen class değerleri ile tanımlanana bu özelliklerden `entry-title` bir web diliminin kullanıcıların tarafından tarayıcıya entegre edilmesi halinde araç çubuğunda gösterilecek başlığını tanımlarken `entry-content` ise web diliminin ana içeriği taşır. Her iki değeri de class olarak almış olan HTML elementleri içerisinde farklı HTML kodları yerleştirebilir.

ÇALIŞMA YAPISI

Internet Explorer açtığı her sayfada `hslice` ile işaretlenmiş HTML elementlerini arar ve iç yapısı uygun olanları otomatik olarak birer web dilimi olarak kullanıcıya gösterir. Kullanıcı bu web dilimini tarayıcısına ekledikten sonra kullanıcı veya programcı tarafından tanımlanmış aralıklar ile Internet Explorer web dilimini güncelleyecektir.

Bir önceki bölümde kendi içerisinde `entry-content` içeren bir Web Dilimi örneği gördük. Eğer herhangi bir web dilimi kendi içeriğini doğrudan HTML kodları olarak kendi ana elementi içerisinde `entry-content` içinde tanımlarsa Internet Explorer söz konusu web diliminin bulunduğu sayfayı belirli aralıklarla istemciye indirir.

Sonrasında gerekli parsing işlemlerini de yapan Internet Explorer web dilimini ID'si üzerinden HTML kodunda bulup içerisindeki `entry-content`'te değişiklik var ise kullanıcıyı uyarır. Bu yapı programlama açısından en kolay hazırlanabilen yapıdır. Tek yapmanız gereken sayfanızda dilimlenmesini istediğiniz bölümleri tespit edip gerekli yerbelerde `hslice`, `entry-content` ve `entry-title` CSS sınıflarını atamak.

Düzen yandan farklı senaryolarda daha esnek yapılar oluşturabilmek adına **Web Dilimler**'inin içeriklerinin farklı kaynaklardan çekilebilmesi de mümkün. Seçeneklerden ilki web diliminin içeriğini ayrı bir web sayfasından almak.

DISPLAY SOURCE DEĞİŞİKLİĞİ

Yarattığınız bir web diliminin tarayıcının üzerinde açılan kısmını hariç olarak tanımlamak isteyebilirsiniz. Böyle bir talep aslında çok basit bir nedeni olabilir. Web diliminizin kesildiği yerdeki hali ile tarayıcıda kesilmiş halinin farklı tasarımlar ile gözükmesini isteyebilirsiniz.

niz. Bu gibi durumlarda herhangi bir web dilimi tanımlarken web dilimi kesildikten sonra tarayıcının hangi veri kaynağından içerik takibi yapabileceğini belirleyebilirsiniz.

```
<div class="hslice" id="Urun1">
    <h1 class="entry-title">
        Yeni Ürün</h1>
    <div class="entry-content" href="http://daron.yondem.com/baska.aspx?ID=2">
        </div>
</div>
```

Yukarıdaki kod içerisinde her zamanki gibi yine **entry-content** class değerine sahip bir elementimiz var fakat bu sefer söz konusu elementin bir de **href** değeri bulunuyor. Bir **DIV** elementinin **HREF** özelliğine sahip olması normal şartlarda hiçbir şey ifade etmez. Oysa Internet Explorer 8 için bu çok anlamlı bir işaretleme. **Entry-content**'e verilen **href** değerinde web diliminin içeriğini saklayan başka bir sayfanın adresi yer alıyor. Bu web dilimi şu anda bulunduğu sayfada farklı şekillerde gözükebilse de kullanıcı tarafından tarayıcıya eklendiğinde tarayıcı değişiklikler için **href** değerinde verilmiş sayfayı kontrol edecek ve söz konusu sayfadaki içeriği kullanıcıya gösterecektir. Böylece ana sayfada yer alan web dilimi ve görsellik bozulmadan tarayıcının araç çubuğuunda gözüken web dilimi tasarnı tamamen özelleştirilebilir.

FARKLI WEB DİLİMLERİ'NI BİR BİRİNE BAĞLAMAK

Bazı durumlarda farklı web dilimleri için ortak bir kaynak kullanmak veya sitenizin farklı yerlerinden kesilebilir şekilde gözüksede aslında aynı web dilimine yönlendirme yapmak isteyebilirsiniz. Bu gibi durumlarda yarattığınız yeni bir web dilimine **feedurl** verebilirsiniz.

```
<div class="hslice" id="Urun1">
    <div class="entry-title">
        Urun Adı
    </div>
    <a rel="feedurl" href="http://daron.yondem.com/baska.aspx#Urun2"></a>
</div>
```

Yukarıdaki kod ile tanımlanan bir web dilimi aslında **baska.aspx** sayfasındaki ID'si **Urun2** olan web dilimini hedef göstermektedir. GörüTÜNÜZ a tag'ı ile koyulan link içerisinde herhangi bir yazı yazılmadığı için kullanıcılar tarafından söz konusu link algılanmayacak fakat Internet Explorer linkin rel ve href özelliklerinden yola çıkarak gerekli işlemleri yapabilecektir.

WEB DİLİMI'NE KAYNAK OLARAK RSS KULLANMAK

Bir web diliminin veri kaynağını özelleştirme yolunda gidilebilecek son nokta olarak RSS XML kaynaklarından bahsedilebilir. Normal şartlarda bir RSS içerisinde birden çok item tag'ı bulunur. Oysa bir web dilimi için tek bir item yeterlidir. Internet Explorer RSS kaynağı içerisinde her zaman ilk item'ı alarak içerisindeki HTML'i araç çubuğuında kesilmiş bir web dilimi içerisinde gösterecektir.

```
<div class="hslice" id="Urun1">
    <div class="entry-title">
        Ürün Adı
    </div>
    <a rel="feedurl" href="/rssfeed.xml"></a>
</div>
```

Yukarıdaki gördüğünüz örnek Web Dilimi kendi kaynağını rssfeed.xml adında bir dosya olarak tanımlıyor. Dikkat ederseniz bu web diliminin herhangi bir entry-content'i yok. Rel özelliği feedurl olarak atanmış a tagının href özelliğine hedef veri kaynağının adresi yerleştirilmiş durumda. Gelin bir de hızlıca RSS'in yapısına göz atalım.

```
<?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0">
    <channel>
        <title>Web Dilimi RSS</title>
        <ttl>120</ttl>
        <item>
            <title>Ürün Adı</title>
            <description>HTML &lt;b&gt;kodları&lt;/b&gt; kullanılır</description>
        </item>
```

```
</channel>
</rss>
```

RSS kaynağı içerisinde sadece bir ITEM bulunuyor. Söz konusu Item'ın **title** değeri Internet Explorer içerisinde araç çubuğunda gözükeceken **description** değeri ise Web Dilimi'nin içeriğini belirleyecektir. Son olarak **ttl** tagları arasındaki değer ise bu web diliminin ne kadar sürede bir istemci tarafından kontrol edilmesi gerektiğini belirler. TTL (*Time To Live*) değeri olarak verilebilecek en düşük değer 15'dir (dakika).

```
<div class="hslice" id="Urun2">
    <h1 class="entry-title">
        Urun Adı </h1>
    <div class="entry-content">
        <p>
            Ürün açıklaması.</p>
        <div class="ttl" style="display: none;">
            15</div>
    </div>
</div>
```

Eğer web diliminiz harici RSS kaynağı kullanmayacaksanız TTL değerini HTML kodu içerisinde de yukarıdaki gibi tanımlayabilirsiniz. **class** özelliğine **ttl** değeri verilen DIV elementi içerisinde değerin ekranda gözükmemesi için elementin CSS stil özelliklerinden **display** özelliği **none** olarak değiştirilebilir.



WEB DİLİMLERİNE DİNAMİK RSS

Görsel Arama bölümümüzde incelediğimiz üzere hem ASP.NET hem de PHP ile dinamik XML üretimi kolayca yapılabiliyor. Bu çerçevede harici RSS kaynaklarına bağlı web dilimlerinin RSS kaynakları da özünde birer XML dosyası olduğu için bu kaynaklar da rahatlıkla sunucu tarafında dinamik olarak oluşturulabilir. Böylece kullanıcılar çok daha bir şekilde web sitenizi takip edebilirler.

YAZILIM GELİŞTİRİCİLER İNÇİN INTERNET EXPLORER 8

JavaScript Yenilikleri	53	JavaScript Debuging	67
tostaticHTML Metodu	53	Profiler ile Optimizasyon	68
Dahili JSON Sınıfları...	55	Image Optimizasyonu	69
AJAX Navigasyon	57	Pratik Araçlar	70
Online ve Offline Çalışma Yapısı	58	Önce Güvenlik	71
Cross Domain Request Nesnesi	60	Data Execution Prevention	71
DOM Veri Ambarı	64	Kullanıcıya Özel ActiveX	72
Session Storage	64	Siteye Özel ActiveX	73
Local Storage	65	XSS Saldırıları	74
Yazılımcı Araçları	65	MIME TYPE Kararları	75
CSS ve HTML DOM		O Uygulamayı Sitemde Çalıştırma!	75
Gerçek Zamanlı Düzenleme	65	DOM Elementlerini Sorgulayın	75

Kitabımızın bu bölümünde Internet Explorer 8.0 ile beraber gelen ve özünde sadece biz yazılımcıları ilgilendiren özelliklere göz atacağız. Yeri gelecek çok basit fakat hayatımıza kolaylaştırın JScript yeniliklerine değinecek, yeri gelecek uygulamalarımızın tüm yapısını değiştirecek altyapılardan bahsedeceğiz. Gelin konuyu daha fazla uzatmadan örneklerimiz ile Internet Explorer 8.0 içerisinde bizleri gizli saklı neler bekliyor göz atalım.

JAVASCRIPT YENİLİKLERİ

Özellikle istemci taraflı programlama söz konusu olduğunda çoğunkula aklımıza JavaScript gelir ve yazdığımız JavaScript kodlarının kontrol edilmesinden tutun performansına ve güvenliğine kadar çoğu noktada farklı testler yapmamız gereklidir. Bu bölümde Internet Explorer 8.0 ile beraber gelen bazı JavaScript yeniliklerinden bahsedeceğiz.

TOSTATICHTML METODU

Kullanıcılarından veri girişi alan ekranlarda aldığınız veri girişini doğrudan sayfa içerisinde kullandığınızda Script Injection dediğimiz güvenlik açığı ile karşı karşıya kalabilirsiniz. Aslında kodumuz kullanıcından bir metin değeri beklerken söz konusu metin içerisinde bulunan ve kullanıcının kasıtlı olarak yazmış olduğu JScript kodlarını farkında olmadan sayfada farklı yerlere yerleştirmiş olabilirdir. Bu durumda web sayfamızda çalışacak kodları doğrudan dışarıdan almışız demektir. İşte bu gibi bir güvenlik açığından kaçınabilmek adına

kullanıcıların giriş yaptığı tüm ekranlardaki verileri kullanmadan önce gerekli kontrolleri de yapmamız gereklidir.

Internet Explorer 8.0 ile beraber gelen `tostaticHTML` metodu parametre olarak kendisine bir HTML verisi alır ve bu veriyi bir metne çevirecek şekilde gerekli filtreleme ve temizleme işlemlerini yapar.

```
<%@ Page Language="VB" AutoEventWireup="false" Code-
File="Default.aspx.vb" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>

    <script type="text/javascript">
        function Goster() {
            alert(toStaticHTML(document.getElementById('icerik').innerHTML));
        }
    </script>

</head>
<body>
    <form id="form1" runat="server">
        <div>
            <input id="Text1" type="text" />
            <input id="Button1" onclick="Goster()" type="button" value="button" />
            <div id="icerik">
                DENEME
                <script type="text/javascript">
                    function Deneme() {
                        document.getElementById("icerik").innerHTML = document.getElementById("Text1").value;
                    }
                </script>
            </div>
        </div>
    </form>
</body>
```

```

</div>
</form>
</body>
</html>

```

Yukarıdaki örnekte içinde JavaScript kodu bulunan bir DIV elemen- tini alarak içeriğini kullanıcıya göstermek istiyoruz. Tam da bu işlemi yaparken arada toStaticHTML metodunu kullanıyoruz. Böylece tüm script'ler kodumuzdan temizleniyor ve ortaya sadece “DENEME” metni çıkıyor. Bu sistemi özellikle farklı alanlarında alınan verile- rin kontrolü için de rahatlıkla kullanılabilir.

DAHİLİ JSON SINIFLARI...

AJAX ile programlama yaparken en sık kullandığımız veri taşıma for- matlarından biri **JSON** (*JavaScript Object Notation*). Detaylarına bak- tığımızda ASP.NET AJAX içerisinde de tüm PageMethod'lar sunu- cundan istemciye JSON formatından veri gönderiyor. Bunun tabi ki anlamlı nedenleri var; JSON XML'e kıyasla çok daha düşük boyutta çok daha fazla veri taşıyabiliyor. Ayrıca JSON'un JavaScript tarafın- da kullanımı XML'e göre daha kolay. JSON'un artık neredeyse tüm AJAX kütüphanelerinde kullanıldığı düşünürsek Microsoft tarafın- da da IE 8.0'a JSON ile ilgili özelliklerin eklenmesi gereki̇ği kararı alınarak Beta 2 ile beraber doğrudan JSON'a özel yeni JavaScript özellikleri sunuluyor. Gelin neler varmış beraber inceleyelim.

AJAX kullanılan sitelerdeki en büyük güvenlik açıklarının arkasına baktığımızda belki de çoğu zaman **Eval** metodu karşımıza çıkacaktır. Peki **Eval** metodu ne yapar? **Eval** kendisine **String** olarak verilen bir değişkenin içindekileri bir kodmuş gibi çalıştırır. Örneğin aşağıdakı kod çalıştırıldığında ekrana normalde **alert** JavaScript metodу çalıştırılarak getirilen bir mesaj kutusu gelecektir.

```

var Degisken = "alert('DENEME');";
eval(Degisken);

```

Gördüğünüz gibi aslında çalıştırılan kod doğrudan bir metin olarak başka bir değişkenin içerisinde saklanıyor. Güvenlik açığı nerede? **JSON** kullandığınızda sunucudan gelen bir metni doğrudan **Eval** ile çalıştırarak JSON'un bir JavaScript dizisi olmasını sağlayız. Bu nok- tada eğer veri harici bir kaynaktan geliyorsa veya bu veriyi daha öncे kullanıcı girmişse aslında kullanıcının doğrudan sizinizde çalıştırı-

lacak bir JavaScript kodunu girmesini sağlamış olursunuz. Bunu en-gellemenin tabi ki yolları var, harici kütüphanelerde yine JavaScript ile eval'e verilen metinleri Parse ederek inceleyen metodlar mevcut, hatta bazıları bunun için Regular Expression bile kullanabiliyor. Fa-kat bunların hepsi unutmayalım ki yine JavaScript metodları kullanı-larak yapılan işlemler. Oysa bizim doğrudan tarayıcı motorunda bu iş-lemleri yapan bir sistemimiz olsa daha hızlı ve güvenli olmaz mı?

IE 8.0 ile beraber doğrudan JSON'u parse edebilme özelliği geliyor. **Serialization** ve **Deserialization** işlemlerini doğrudan IE içerisinde **Native** kodla yaptığı için tabi ki ortaya çok daha yüksek performanslı bir çözüm çıkıyor.

```
var KitaplarMetin = "{\"kitaplar\":{\"kitap\":[" +
[\"ASP.NET AJAX\", \"ASP.NET 3.5 AJAX\"]]}";
var Kitaplar = JSON.parse(KitaplarMetin);
alert(Kitaplar.kitaplar.kitap[0]);
```

Yukarıdaki kod içerisinde **KitaplarMetin** değişkenine bir JSON met-ni aktarıyoruz. Bu metnin bir şekilde JavaScript nesnelerine çevrilme-si gerekiyor. Bunun için klasik eval metodunu kullanmaktansa doğru-dan IE 8.0 ile beraber gelen JSON nesnesinin **parse** özelliğini kullanı-miyoruz. Son satırda ise artık **Kitaplar** değişkenimize aktarılan JSON nesnelerinin içerisinde ilk kitabin adını alıyoruz.

Peki ya eldeki JavaScript nesnelerini JSON'dan standart metne çevir-mek istersek ne yapacağız? Bu durumda da yardımımıza yine JSON nesnesinin **stringify** metodu yetişiyor.

```
var Kitaplar = {
    kitaplar:{
        kitap:[
            'ASP.NET AJAX',
            'ASP.NET 3.5 AJAX'
        ]
    }
};

alert(JSON.stringify(Kitaplar));
```

Yukarıdaki örnekte yarattığımız **Kitaplar** adındaki nesneyi **stringify** metodu ile bir **JSON** metnine çeviriyoruz. Ayrıca isteyenler haricen **. toJSON** metodunu da kullanabilirler. IE 8.0 ile beraber standart olarak **Boolean**, **String**, **Date**, **Number** sınıflarının prototiplerine **toJSON** metodu eklenmiş durumda.

Bugüne kadar harici kütüphanelerle yukarıdaki işlemleri yaptığımuz için tabi ki bu kütüphanelerin **JSON** nesnesi tanımladıklarını unutmadık. Böyle bir durumda aşağıdaki gibi ufak bir kontrol ile tarayıcının **JSON** nesnesini tanımlayıp tanımlamadığını kontrol edip eğer tanımlı değilse harici scriptler üzerinden ilerlemeyebilirsiniz.

```
if(!this.JSON)
{
    JSON = ....;
}
```

AJAX NAVIGASYON

Web sitelerinde AJAX kullanımı gün geçtikçe artıyor ve neredeyse artık AJAX kullanılmayan sitelere demode gözüyle bakmaya başladık. Bu çerçevede AJAX'lı sitelerde yaşadığımız en büyük sorunlardan biri tarayıcı geri ve ileri düğmelerinin çalışmaması. Konu ile ilgili farklı tarayıcılar için farklı taktiksel çözümler geliştirilmiş olsa da artık IE 8 ile beraber standart bir çözüm de geliyor.

AJAX ile sayfaları programatik olarak yenilediğimizde sayfa adresinin değişmiyor olması aslında sorunumuzun esas nedeni. Eğer sayfanın adresini değiştirebilseydik tarayıcı da bu adresi kendi tarayıcı geçmişine ekleyebilecekti. Tam da bu noktada web adreslerinin sonuna ekleyebildiğimiz çapalar akılımıza geliyor. Örneğin default.aspx#1 ile default.aspx#2 arasında aynı sayfa olsalar da ayrı adresler şeklinde tanımlanıyor. Bu adresin çapa kısmı, yani # işaretinden sonraki kısmı değişse de tarayıcı sayfayı baştan yüklemeyecektir çünkü konsept olarak aynı sayfa içerisinde farklı bir konum arayacaktır. Çapaların bu özellikleinden faydalananarak normal kullanımının yanı sıra sadece adres çubuğuundaki adresi değiştirebilmek için de kullanabiliriz. Peki tüm bunları nasıl yapacağız?

window.location.hash

Yukarıda gördüğünüz kod AJAX kullandığımız web sitelerinde hayatımızi kurtaracak olan kodun sadece ufak bir parçası. Doğrudan **win-**

dow.location.hash değerine aktardığınız değerler artık tarayıcının adres çubuğu # işaretinden sonrasına eklenecek ve hash değeri her değiştirildiğinde bir önceki adres de tarayıcı geçmişine kaydedilecek. Böylece hızlı bir şekilde AJAX sitemizde ileri ve geri düğmelerinin aktif hale gelmesini sağlayabiliyoruz. Fakat bir sonraki adımda kullanıcının ileri veya geri düğmelerine bastığını da algılamamız gerekiyor ki duruma göre gerekli AJAX isteklerini tekrar yaparak sayfayı eski haline getirebilelim. Unutmayın sayfa hiçbir şekilde tekrar yüklenmeyecek o nedenle sayfayı eski haline getirmek de tamamen bize kalıyor. Hash değerine daha önce söz konusu sayfaları tanımlayacak bir değer verdiyseniz kullanıcı tarayıcıda gezerken gidilmek isteyen sayfanın hash değeri size verildiğinde sayfanın eski halini üretебilmeniz hiç de zor olmayacağındır.

```
<body onhashchange="HashChanged();">
```

Yine IE 8 ile beraber gelen AJAX navigasyon sisteminin bir parçası da kullanıcının tarayıcı içerisinde ileri veya geri düğmelerini kullandığında bizi haberدار edecek olan event-listener. Standart şekli ile herhangi bir HTML sayfada yer alan Body tagının **onhashchange** event'ına özel bir JavaScript listener fonksiyonu ekleyebilirsiniz. Böylece artık ileri veya geri navigasyonlarında bizim **HashChanged** metodumuz çalıştırılacak. Peki ne yapacağımız **HashChanged** içerisinde? Hash değiştiğinde göre yeniden **window.location.hash** üzerinden yeni hash değerini alıp ona uygun AJAX isteklerini gerçekleştirerek siteyi eski haline getirmemiz gerekiyor.

Tüm bu manzara içerisinde önemli olan noktalardan biri de AJAX sitelerinde yaşadığımız bir diğer sorunu daha gidermiş olmamız. Bildiğiniz üzere AJAX sitelerinde sitenin farklı durumlarına ait farklı URL adresleri de bulunmaz. Oysa bizim # implementasyonu ile beraber artık sitemizin **hash** bilgisini değiştirdiğimiz her noktada yeni bir sayfa ve URL de yaratmış oluyoruz.

ONLINE VE OFFLINE ÇALIŞMA YAPISI

Özellikle RIA uygulamalarındaki en büyük sorunlardan biri anlık internet bağlantısı kesintilerinde sayfanın bir daha geri ulaşlamayacak şekilde ekrandan kaybolması veya farklı hataların ortaya çıkarak geri dönüşü imkansız hale getirmesidir. Bu hoş olmayan durumu artık Internet Explorer 8.0 ile beraber çözebiliyoruz. IE 8.0 içerisinde **navig-**

tor.offline nesnesi ile istemci tarafında internet bağlantısının o an için olup olmadığı kontrol edebildiğimiz gibi internet bağlantısı ile ilgili değişiklikleri algılayacak event-handler tanımlamaları da yapabiliyoruz. Bu kolaylıklar ile artık **AJAX** veya **Silverlight** uygulamalarının istemci tarafındaki kesintileri algılayarak uygun bir şekilde kullanıcıyı uyarmaları mümkün. Hatta belki de internet bağlantısının koptuğu- nu algılayan uygulama veriyi DOMStorage'a saklayıp bir sonraki çalıştığında sunucuya gönderebilir. Minik bir örnek ile bu işlemleri nasıl yapabileceğimize göz atalım.

```
<body ononline="VarMi();" onoffline="VarMi();">
    <form id="form1">
        <div id="icerik">
        </div>
        <input onclick="VarMi();" id="Button1"
type="button" value="button" />
    </form>
</body>
```

Yukarıdaki HTML kodu içerisinde özellikle koyu yazılı noktalara dikkat etmemiz gerekiyor. Birazdan yazacağımız **VarMi** adındaki JavaScript fonksiyonumuz istemcide internet bağlantısı olup olmadığını kontrol edecek. Fakat hangi durumlarda bu kontrolü yapacağız? İlk olarak söz konusu fonksiyonumuza sayfamızda bir düğmeye bağladık. Böylece istediğimiz zaman tıklayarak internet bağlantısı olup olmadığını öğrenebiliriz. Diğer yandan istemcide internet bağlantısı koptuğunda veya internet bağlantısı geldiğinde de **VarMi** fonksiyonumuzun çalışarak gerekli değişiklikleri yapmasını istiyoruz. O nedenle body'nin **ononline** ve **onoffline** özelliklerine de söz konusu fonksiyonumuzun adını yazıyoruz. Böylece **online** durumunda yani internet bağlantısı geldiğinde veya **onoffline** durumunda yani internet bağlantısı kesildiğinde de anında **VarMi** fonksiyonumuz çalıştırılacak. Gelelim şimdi de **VarMi** fonksiyonumuza yazmaya.

```
function VarMi() {
    if (window.navigator.onLine) {
        $get("icerik").innerHTML = "internet var";
    }
    else {
```

```

        $get("icerik").innerHTML = "Internet YOK";
    };
}

```

Aslında kod çok basit. **navigator.onLine** metodu bize geriye bir **Boolean** değeri döndürüyor. Eğer tarayıcıda o an internet bağlantısı varsa sayfadaki bir DIV içeresine uygun uyarı mesajını yazıyoruz. Aynı şekilde eğer bağlantı yoksa bu sefer de farklı bir uyarı mesajı yazıyoruz. Siz örneklerinizde bu durumlara göre farklı işlemler yapabilirsiniz. Böylece hazırladığınız web sitesi yeri geldiğinde belki bazı işlemleri bir süreliğine sunucudan bağımsız olarak da yapabilir ve nasıl çalışacağına internet bağlantısının durumuna bakarak kendisi karar verebilirim.

CROSS DOMAIN REQUEST NESNESİ

Bir alan adından yola çıkararak başka bir alan adından veri almak istediniz mi? Veya tam tersi, belki de veri göndermek istediniz. Normal şartlarda bu gibi alan adları arası veri trafiği güvenlik nedeni ile tarayıcılar kapalıdır. Oysa gönül isterdi ki en azından her iki alan adının da sahibiysek veya her iki alan adına da admin olarak erişimiz var ise karşı taraftan veriye erişmek isteyenlere izin verme hakkımız olsun. İşte Internet Explorer 8 ile beraber gelen **XDomainRequest** nesnesi eski **XmlHttpRequestler**'e benzer kullanımı ile tam da bu işlevselliliği sağlıyor. Normal bir **XmlHttpRequest**'ten farklı olarak **XDomainRequest** gerektiğinde karşı alan adı ile ilgili kontrolleri de yaparak birden çok alan adı arasındaki iletişimini rahatlıkla sağlayabiliyor.

Internet Explorer 8 üzerinden bir XDomainRequest nesnesi yaratarak sunucudan talepte bulunduğu anda talebiniz ile beraber otomatik olarak **XDomainRequest: 1 http header** bilgisi gönderilecektir. Eğer bu header bilgisine karşılık olarak uzak noktadan da **XDomainRequestAllowed: 1** header bilgisi gelirse artık her iki taraf da birbirleri ile iletişimle geçebilir demektir.

Bahsettiğimiz şekilde kendi sunucunuzda bulunan bazı veri kaynaklarına bu şekilde farklı alan adlarından da erişilebilmesini istiyorsanız sunucu tarafından dinamik olarak gerekli olan http Header bilgilerini eklemeniz yeterli olacaktır.

[ASP.NET / C#]

```
Response.AppendHeader("XDomainRequestAllowed", "1");
```

Unutmayın ki **Cross Domain Request** yaratırken talebin başladığı adresle ait protokol ile hedef adresin protokolü aynı olmalıdır. Örneğin `http://` ile başlayan bir adresten `https://` ile başlayan bir adrese talep hiçbir şekilde gönderemezsiniz.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript">
        var xdr;
        function Getir() {
            xdr = new XDomainRequest();
            xdr.onload = geldi;
            xdr.open("GET", "http://twitter.com/
statuses/public_timeline.xml");
            xdr.send();
        }
        function geldi() {
            alert(xdr.responseText);
        }
    </script>
</head>

<body>
<input id="Button1" onclick="Getir();" type="button"
value="button" />
</body>
</html>
```

Yukarıdaki örnek kod içerisinde basit bir şekilde XDomainRequest nesnesinin yaratılma şeklini görebilirsiniz. XDR'ın onload durumu veri geldiğinde çalıştırılacak olan event-listener'i tanımlar. Bizim örneğimizde **geldi** adındaki JavaScript fonksiyonu veri geldiğinde çalıştırılacaktır. **Geldi** fonksiyonu çalıştırıldığında anda tekrar elimizdeki XDR üzerinden `responseText` alarak gelen veriye ulaşabiliyoruz.

Daha detaylı bir örnek için aşağıdaki kodu inceleyebilirsiniz. Satır aralarındaki yorumlara dikkat etmenizde fayda var.

```
<html>
<script type="text/javascript">
    var xhr;

    function readdata() {
        var dRes = document.getElementById('dResponse');
        dRes.innerText = xhr.responseText;
        // Gelen verinin tipini gösterir
        alert("Content-type: " + xhr.contentType);
        // Gelen verinin toplam boyutunu gösterir.
        alert("Boyut: " + xhr.responseText.length);
    }

    function err() {
        alert("XDR'da hata oldu");
    }
    function timeo() {
        alert("XDR zaman aşımına uğradı");
    }
    function loadd() {
        alert("XDR'a veri geldi");
        alert("Geldi: " + xhr.responseText);
    }
    function progres() {
        alert("XDR çalışıyor");
        alert("Geldi: " + xhr.responseText);
    }

    function stopdata() {
        xhr.abort();
    }

    function mytest() {
        var url = document.getElementById('tbURL');
        var timeout = document.getElementById('tbTO');
        if (window.XDomainRequest) {
            xhr = new XDomainRequest();
```

```
if (xdr) {
    xdr.onerror = err;
    xdr.ontimeout = timeo;
    xdr.onprogress = progres;
    xdr.onload = loadd;

    xdr.timeout = tbTO.value;
    xdr.open("get", tbURL.value);
    xdr.send();
}

else {
    alert('Yaratılamadı');
}

else {
    alert('XDR nesnesi yok, IE 8 yok!');
}

}
</script>
<body>
    <h2>XDomainRequest</h2>
    <input type="text" id="tbURL" va-
lue="http://www.contoso.com/xdr.txt"
style="width:300px"><br>
    <input type="text" id="tbTO" value="10000"><br>
    <input type="button" onclick="mytest()" value=
"Getir">&nbsp;&nbsp;&nbsp;
    <input type="button" onclick="stopdata()" value=
"Durdur">&nbsp;&nbsp;&nbsp;
    <input type="button" onclick="readdata()" value=
"Oku">
    <br>
    <div id="dResponse"></div>
</body>
</html>
```

Örneğimizde sayfada yer alan metin kutularına XDR’ın nereden veri çekeceğini ve bu işlemin en fazla ne kadar sürebileğini belirtebilirsiniz. Böylece gereğinden uzun süren işlemlerin durdurulmasını veya XDR ile ilgili farklı durumların yakalanmasını da sağlayabilirsiniz.

DOM VERİ AMBARI

DOM Storage olarak adlandırılan, benim ise veri ambarı dediğim mekanizmayı hali hazırda kullandığınız Cookie'lere benzetebilirsiniz. Fakat benzettme noktası durup aslında aradaki farkları incelemek gerek. İlk olarak Cookie'ler ile **DOM Storage** arasındaki en büyük fark birinin 4KB alan sağlarken diğerinin standart olarak 10MB alan sağlayabiliyor olması. Tahmin edeceğiniz gibi aslında basit değerler saklayacağımız bir ortamdan bahsetmiyoruz belki de artık web tabanlı yazılımların dokümanlarını sağlayabilecekleri kendilerine özgü kullanıcı verilerini tutacakları bir ortamdan bahsedebiliriz.

DOM Storage'u özellikle Internet Explorer 8'in online ve offline çalışma mantığı ile de birleştirirseniz aslında ortaya çok ilginç bir manzara çıkabiliyor. Tarayıcı içerisinde rahatlıkla kullanıcının verilerini sunabilen ve kullanıcının internet bağlantısı duruma göre verileri geçici olarak DOM Storage üzerindeki ortama kaydedebilen uygulamalar çok yakında internet ortamında bizleri şaşırtacaktır.

Hemen akla gelen sorulardan biri de eminim ki DOM Storage'in da Cookie'ler gibi sunucuya gönderilen her isteğe eklenip eklenmediği olacaktır. Kesinlikle hayır, DOM Storage içerisindeki veriler sunucuya gönderilmez hatta Cookie'ler gibi son kullanma tarihleri de yoktur.

SESSION STORAGE

DOM Storage içerisinde kullanabileceğiniz yapılardan ilk tamamen session bazlı olarak verilerin tutulması olabilir. Böyle bir durumda verilen Session'ın kaybedilmesi ile beraber kaybolacaktır.

```
<input type="checkbox" onchange="sessionStorage.  
kargo = checked">  
Bu siparişte kargo isteniyor.
```

Yukarıdaki şekli ise basit olarak session Store içerisinde veri kaydedebilirsiniz. Sonrasında da tahmin edeceğiniz üzere aynı şekilde söz konusu veriyi geri alabilirsiniz.

Eğer tanımlamakta olduğunuz session Store öğesi o ana kadar yaratılmış ise otomatik olarak yaratılacaktır.

LOCAL STORAGE

Session Store'dan farklı olarak Local Store içerisinde saklanan veriler alan adı başına saklanır ve tüm oturumlar tarafından rahatlıkla kullanılabilir.

```
<p>
  Bu sayfayı
  <span id="count">0</span>
  kez gördünüz.
</p>
<script>
  var storage = window.localStorage;
  if (!storage.pageLoadCount) storage.pageLoadCount = 0;
  storage.pageLoadCount = parseInt(storage.pageLoad-
Count, 10) + 1;
  document.getElementById('count').innerHTML = storage.
pageLoadCount;
</script>
```

Yukarıdaki örnek içerisinde basit bir şekilde yaratılan **pageLoadCount** değişkeni içerisinde sayfanın kaç defa istemci tarafına yükleniği takip ediliyor. Başlangıçta sıfır olan değer her sayfa açıldığında bir tane arttırılarak ayrı bir element içeresine yazdırılıyor.

YAZILIMCI ARAÇLARI

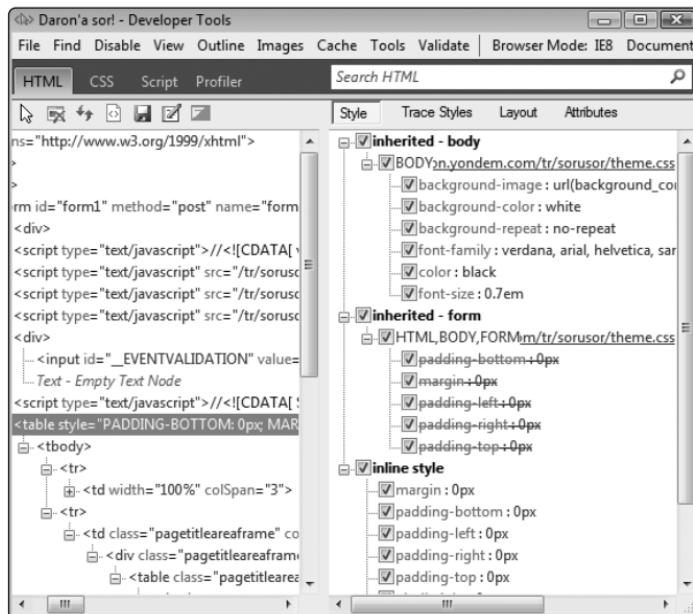
Internet Explorer içerisinde en büyük eksikliklerden biri de biz yazılım geliştiriciler için gerekli profilleme, hata bulma araçlarına sahip olmamasıydı. Bunun için bazı durumlarda harici eklentiler kullanırken bazen alternatif tarayıcılara da yönelik zorunda kalmıştık. Geçin beraberce bu sorunları çözme amacıyla Internet Explorer 8.0'a eklenmiş olan "Developer Tools" uygulamasını inceleyelim.

CSS VE HTML DOM

GERÇEK ZAMANLI DÜZENLEME

Internet Explorer 8.0 içerisinde **Tools** menüsünden ulaşabileceğiniz "Developer Tools" ayrı bir pencerede ayrı bir program gibi açılسا da tabii ki tarayıcı ile entegre çalışıyor. Açılan "Developer Tools" penceresinin sağ üst köşesinde "Pin" düğmesine basarsanız bu pencere kendini IE içeresine yerleştirecektir. Açıığınız herhangi bir sitenin

HTML DOM’unu incelemenin yanı sıra CSS konusunda raporlar da alabiliyorsunuz. Hangi CSS sınıfının nereden geldiği, ve sayfada geçerli olup olmadığını görebiliyorsunuz. Örneğin harici bir CSS ayarı local satır içi bir stil ayarı ile devre dışı bırakılmış olabilir. Tüm bunları rahatlıkla bir liste olarak görmek mümkün.



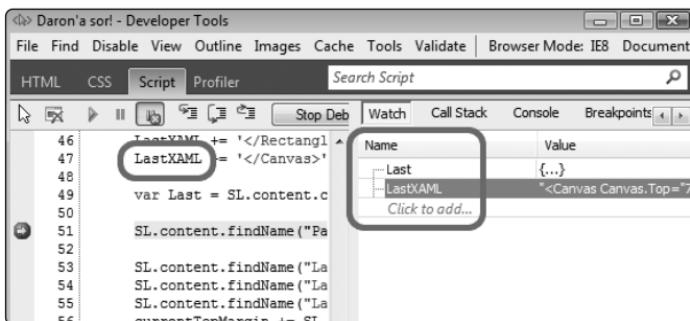
CSS ve DOM kontrol altında!

Yukarıda gördüğünüz ekranda sol tarafta sayfanın DOM ağacını inceleyebilirsiniz. DOM içerisinde herhangi bir element özel olarak seçildiğinde o elementi etkileyen tüm CSS sınıfları, özellikleri ve bu özelliklerin nereden geldikleri sağ tarafta görülebiliyor. Daha da güzeli sağ taraftaki herhangi bir özelliğe tıklarsanız istediğiniz bir değeri değiştirerek IE içerisinde gerçek zamanlı olarak sonucu görebiliyorsunuz.

Isterseniz “Developer Tools” içerisinde HTML tabından CSS tabına geçerek doğrudan sayfadaki tüm CSS özelliklerini yakalayabilir ve gerçek zamanlı olarak değişiklikler de yapabilirsiniz. Tüm bu değişiklikleri tamamladıktan sonra araç çubuğundaki “Kaydet” düğmesine basmanız CSS dosyanızın o anki hali ile kaydedilmesi için yeterli.

JAVASCRIPT DEBUGING

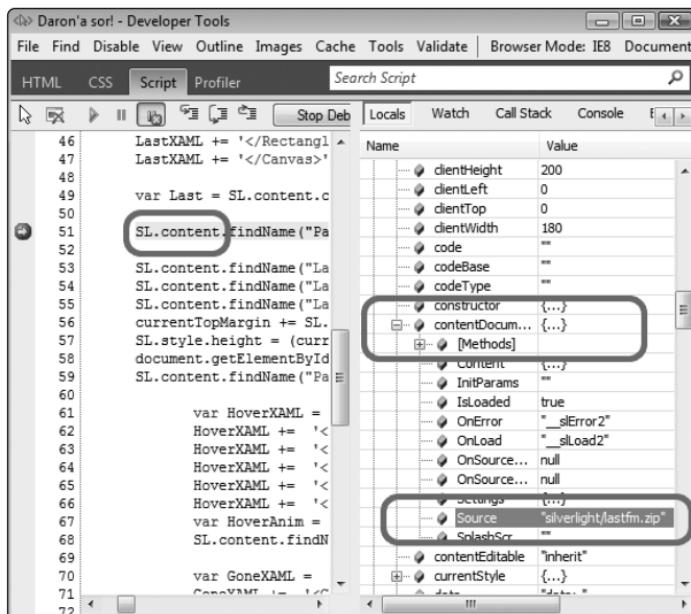
Visual Studio içerisinde JavaScript Debuging'e kıyasla çok daha başarılı bulduğum **IE 8.0 Developer Tools** içerisinde JavaScript araçlarının en büyük avantajı doğrudan IE ile beraber çalışıkları için tarayıcı içerisindeki tüm aktiviteyi takip edebiliyor olmak. İsterseniz herhangi bir JavaScript değişkenine aynı Visual Studio içerisinde VB veya C# kodlarına yaptığımız gibi Watch'lar ekleyin veya istediğiniz bir adıma BreakPoint yerleştirin. Hatta F10 ve F5 gibi Visual Studio kısayolları bile aynı.



JavaScript tarafında **Watch** koyarak durumu takip edin.

Özellikle **Silverlight 1.0** tarafında yazılan JavaScript kodlarının veya AJAX tarafında yazılan veri ulaşım kodlarının incelenmesi ve hataların bulunması epey kolaylaşmış durumda. Aşağıdaki görsel içerisinde JavaScript ile tanımlanmış bir **Silverlight** nesnesinin **Source** özelliğine verilen değeri doğrudan "Locals" tabı üzerinden giderek sayfada tanımlanmış tüm JavaScript nesnelerini listeleyip bulabiliyoruz.

Tüm bunları yaparken istediğiniz anda herhangi bir değişkenin değerini Developer Tools içerisinde değiştirebilirsiniz, sonucu gerçek zamanlı olarak IE içerisinde göreceksiniz.

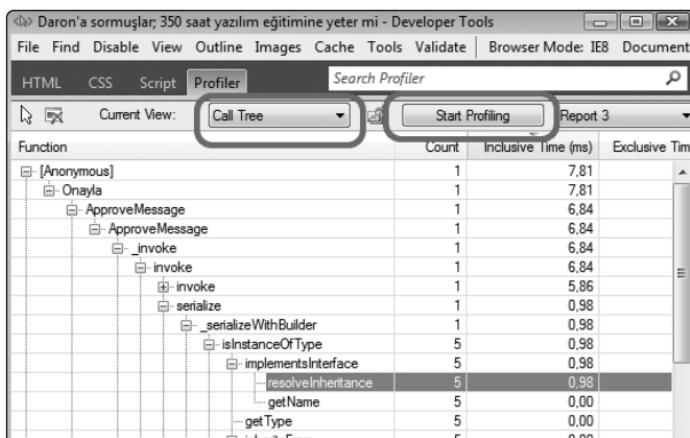


JavaScript tarafındaki sayfada bulunan tüm nesneler ve değişkenler düzenlenebiliyor.

PROFILER İLE OPTİMİZASYON

Yazdığımız kodun ne kadar optimize olduğunu anlamak çok önemli. Bunun için tam olarak hangi kodun daha çok zaman aldığı bilmeliyiz. Özellikle SQL tarafında alışık olduğumuz **Profiler** sistemine benzeyen bir yapı ile artık IE üzerinde de **Developer Tools** içerisinde bir Profiler bulunuyor. “Start Profiler” düğmesine bastıktan sonra IE penceresine geçip site üzerinde istediğiniz işlemleri yapabiliyorsunuz.

Sonra Developer Tools'a dönerek “Stop Profiler” dediğinizde geçen süre içerisinde yaptığı tüm işlemlerin bir listesi karşınıza çıkıyor. Bu listeyi ister bir “Function” listesi olarak alın ister bir ağaç görüntüstünde hangi function’ın hangisini çağrıdıguna bakarak inceleyin. Önemli olan artık hangi işlemin ne kadar süredğini görebiliyor olmamız.



Kod optimizasyonu için Profiler.

IMAGE OPTİMİZASYONU

Bazı durumlarda bir web sitesine koyduğumuz resmin hem en ve boy boyutu hem de dosya boyutuna bakabilmek için doğrudan dosyanın kendisini bulmamız gerekebilir.

Developer Tools içerisinde **Image** menüsü böyle bir durumda yardımımıza koşuyor ve doğrudan gerçek zamanlı olarak gezdiğiniz tüm sitelerdeki resimlerin boyutlarını resimlerin üzerine yazıyor. Gerçekten hoş bir özellik.

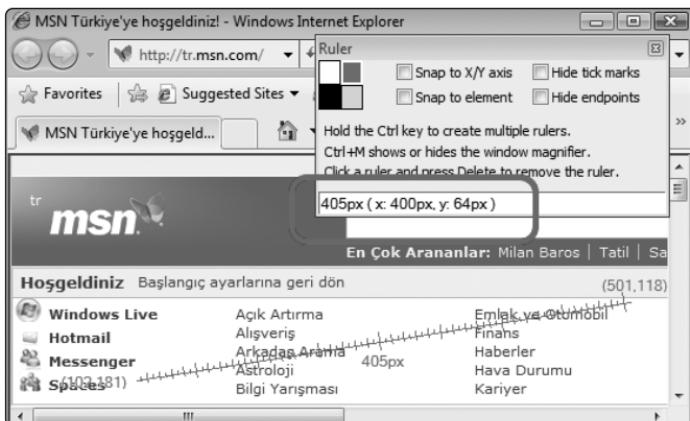


Gerçek zamanlı olarak sitedeki resimlerle ilgili detayları görebilirsiniz.

PRATİK ARAÇLAR

Bir sayfadaki tüm DIV'leri görmek mi istiyorsunuz, tek yapmanız gereken “Outline” menüsünden DIV seçeneğini işaretlemek. Böylece gerçek zamanlı olarak sayfa içerisinde tüm DIV'ler ayrıca birer çerçeve içine alınacaktır. Sadece DIV elementleri değil istediğiniz bir elementi kendiniz de belirterek aramasını sağlayabilirsiniz.

Tools menüsünden “Show Ruler” özelliği ile fare ile ekranda tıkladığınız iki nokta arasında mesafeyi piksel olarak ölçebilir bu mesafeler arasında X ve Y koordinatlari farkları görebilirsiniz.



Pratik araçlardan biri: **Cetvel!**

Aynı menüdeki “Show Color Picker” ile gezdiğiniz sitedeki herhangi bir rengi seçebilir **RGB** ve **HEX** renk kodlarını alabilirsiniz. “Resize” menüsünden ekran çözünürlükleri seçerek tarayıcının otomatik olarak farklı ekran çözünürlüğündeymiş gibi boyutlandırılmasını sağlayabilirsiniz.

Burada daha bahsedemediğimiz bir çok özellik “Developer Tools” ile beraber Internet Explorer 8 içerisinde geliyor. Bir yazılımcı olarak bu gelişme beni çok mutlu etti diyebilirim.

ÖNCE GÜVENLİK

Neredeyse her yazılımin yeni sürümde artık “Daha Güvenli” sözlerini duymaktan bıktık. Peki nedir daha güvenli olan? diye sorduğumuzda çoğu zaman tatmin edici cevaplar alamıyoruz. O nedenle ben bu bölümde sizlerle Internet Explorer 8 içerisindeki güvenliğe bakış tekniğin bir açıdan inceleyeceğim.

DATA EXECUTION PREVENTION

Kısa adı **DEP** olan sistemin aslında doğrudan IE ile bir ilişkisi yoktu. Windows XP ve 2003 ile beraber gelen altyapı sisteme belirli bellek alanlarının korunmasını ve bu alanlardan kod çalıştırılmasını engelleme bilmesini sağlıyor. Böylece **Buffer Overrun** saldırılara ait boşlukların bulunması çok daha zor bir hal alıyor. Tabi “Managed Code” yazarları olarak VB.NET ve C#.NET programcılara bu yapı yabançı gelecektir. Maalesef şimdilik çok detaylarına girme şansımız yok.

Şu ana kadar bu altyapı Windows'da olmasına karşın maalesenf **IE 7.0** ile beraber varsayılan ayarlarda açık gelmiyor. Bunun mantıklı bir nedeni var; DEP ile uyumsuz programların bugüne kadar çalışması gerekiyordu, özellikle IE 7.0 için yazılmış çoğu **Plug-In** maalesenf bu durumdaydı. **ATL 7.1** ve öncesindeki uygulamaların DEP ile karşı karşıya gelmesi durumunda uygulamanın kendisine izin verilmeyen bir hafıza alanına yazmaya çalışması sonucu doğrudan uygulamanın sonlandırılması söz konusu. Tabi ki var olan Plug-In'leri ve uygulamaları uygun şekillere düzelterek (**IMAGE_SCN_MEM_EXECUTE** şeklinde işaretlemeler ile) sorunu gidermek mümkün.

Internet Explorer 8.0 tarafında artık DEP **Vista SP1** ve **Server 2008** içerisinde otomatik olarak açık gelecek. DEP ile beraber bir de Vista'da gelen **ASLR** (*Address Space Layout Randomization*)'yi de birleştirdiğimizde ortaya güvenlik anlamında hoş bir manzara çıkıyor. ASLR'nin yaptığı ise sistem her açıldığında Kernel32 gibi belleğe yüklenen sistem öğelerinin her seferinde farklı bellek noktalarına yüklenmesini sağlamak. Böylece kötü niyetli bir kodun saldırma öncesinde doğru hedefi bulması daha zor oluyor.

Vista içerisinde hangi uygulamaların DEP tarafından korunduğunu görmek isterseniz doğrudan “Görev Yöneticisi”/“Task Manager” içerisinde “View>Select Columns” altından “Data Execution Prevention” kolonunu seçerek ilerleyebilirsiniz.

IE 7.0 içerisinde DEP'yi aktif hale getirmek için “Tools/Internet Options/Advanced” sekmesine giderek uygun seçeneği işaretleyebilirsiniz. Unutmayın bunu yapabilmeniz için IE'yi Admin hakları ile açmış olmanız gerekecektir.

KULLANICIYA ÖZEL ACTIVEX

Özellikle Vista ile beraber gelen UAC (*User Account Control*) sonrasında gelen en büyük şikayetlerden biri ActiveX kontrolleri yüklerken admin haklarının gerekliliğiydi. Kişisel olarak kullandığınız bilgisayarınızda bu bir sorun teşkil etmese de şirket içi domainlere kayıtlı ve farklı güvenlik sınırlamalarını olan bilgisayarlarda bu durum sıkıcı sonuçlar doğuruyordu.

Artık her şey değişti, kullanıcıların Admin haklarına sahip olmasalar da kendi kullanıcı hesaplarına özel olarak ActiveX uygulamaları yükleyebilecekler. Eğer söz konusu ActiveX uygulaması zararlı bir kod

iceriyorsa bu durumun bilgisayar hiçbir şekilde zarar görmeyecektir. Var olan ActiveX uygulamaları herhangi bir sorun yaşamadan bu sistem ile çalışacak.

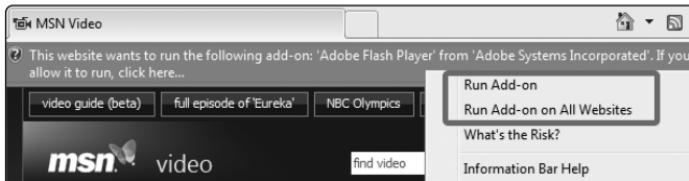
Kullanıcıların herhangi bir **ActiveX** kontrolü ile karşılaşıklarında kontrolü sadece kendileri için veya tüm makine bazında yüklemek isteyip istemediklerini seçebilecekler. Bu seçim şu anki Internet Explorer içerisinde ActiveX kontrolleri için gelen uyarı mekanizmasına dahil edilmiş durumda.

SİTEYE ÖZEL ACTIVEX

Hazırladığınız ActiveX kontrollerinin sadece belirli bir sitede çalışmasını isteyebilirsiniz. Özellikle yüksek güvenlik amacıyla bankacılık uygulamalarında kullanılan ActiveX kontrolleri buna bir örnek olarak gösterilebilir. Bu kısıtlamanın yapılabilmesi için ActiveX uygulaması geliştirilirken “SiteLock Active Template Library”nin kullanılması gerekiyor. Arka planda çalışan mantık aslında çok basit; Internet Explorer içerisinde çalışabilecek ActiveX kontrollerinin “Safe” şeklinde işaretlenmesi gereklidir. Eğer bir ActiveX kontrolü kendi istediği alan adları haricinde çalıştırıldığında kendisini “UnSafe” olarak tanımlarsak IE doğrudan söz konusu ActiveX’i pasif hale getiriyor.

Standart ATL şablonunun yerine oturan **SiteLock** şablonu **IObjectSafety** ve **IObjectSafetySiteLockImpl** üzerinden türüyerek Build esnasından tanımlanan siteler dışında çalışmıyor. **SiteLock 1.14** şablonunu aşağıdaki adresten indirebilirsiniz.

<http://www.microsoft.com/downloads/details.aspx?FamilyID=43cd7e1e-5719-45c0-88d9-ec9ea7fefbcb&displaylang=en>



ActiveX’ler artık birer **Add-On** ve siteye özel yüklenebiliyorlar.

Kullanıcılar ActiveX uygulamalarını IE 8.0 içerisinde birer Add-On olarak göreceleri için istedikleri ayarı “Manage Add-ons” penceresinden yaparak belirli ActiveX’lerin sadece istedikleri sitelerde çalış-

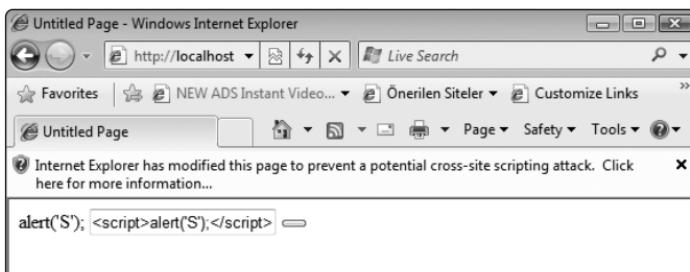
masını sağlayabilirler. Bu ayar “Group Policy” üzerinden de artık yapılabiliyor.

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\Current-
Version\Ext\Stats\{ID}\iexplore\AllowedDomains\
{Domain}
```

Yukarıda adres üzerinden Registry içerisinde gerekli ayarlar bulunabilir. {ID} yerine kontrol edilmesi hedeflenen ActiveX kontrolünün Class ID’si, {Domain} yerine de izin verilen domainler atanabilir.

XSS SALDIRILARI

Son dönemde XSS (*Cross Site Scripting*) saldırıları belki de en sık karşılaştığımız güvenlik açıklarından. Bu teknik ile rahatlıkla bir web sitesi ile kullanıcıyı arasına girerek kullanıcının bastığı tuşlara kadar her tür bilgi alınabiliyor. Bu konuda tam bir koruma sunmanın kullanıcı deneyimini ciddi şekilde kötü durumlara sürüklereceği için minimal koruma mekanizmaları devreye sokulmuş.



XSS Koruması.

Yazılım geliştiriciler isterler bu korumaları sunucu tarafından kapatıyorlar. Tek yapmalı gereken **X-XSS-Protection: 0** şeklinde gerekli **HTTP Header** bilgisinin sayfalarına eklemek.

Bu noktada özellikle bir uyarida bulunmak istiyorum. İstemci tarafında bir tarayıcı olarak IE 8.0’ın XSS koruması her çeşit XSS saldırısını korumamakla beraber kesinlikle bir yazılım geliştiricisinin bu sisteme “güvenerek” hareket etmesine neden olabilecek bir yapı değildir. Unutmamak gereklidir ki herkese IE kullanmayabilir.

MIME TYPE KARARLARI

MIME Type ayarları normal şartlarda **HTTP Header** bilgisi içerisinde saklansa da bugüne kadar Internet Explorer kendi kendine kararlar vererek daha kolay bir kullanım sağlamak için MIME Type değişiklikleri yapabiliyordu. Örneğin text/plain olarak ayarlanmış bir dosya içerisinde HTML kodu varsa bu dosya açıldığında IE içerisinde bir HTML sayfa olarak render edilir. Oysa dosya bir text dosyasıdır ve o şekilde gösterilmelidir.

IE 8.0 ile beraber eğer sunucudan **authoritative=true** HTTP header bilgisini gönderirseniz IE sizin sunucu tarafında istediğiniz MIME Type ayarlarını saygı göstererek herhangi bir değişiklik yapmayacaktır.

□ UYGULAMAYI SİTEMDE ÇALIŞTIRMA!

Bir diğer güvenlik açığı da sitelerde tıklanarak istemci tarafına indirilen herhangi bir uygulamanın veya farklı kodun doğrudan sitenin üzerinde çalıştırılabilir olmasındı. Örneğin bir PDF dosyasına tıkladığında dosyayı indirerek doğrudan IE içerisinde açabilirsınız. Bu işlemi yapabilmeniz için karşınızda uygun seçenekler IE tarafından getirilir.

Eğer kullanıcıya sunulacak dosyayı kesinlikle kullanıcı tarafından disk kaydedilmesini ve doğrudan site üzerinden açılamamasını istiyorsanız **X-Download-Options: noopen** HTTP Header bilgisini vererek bu işlemin tamamlanmasını sağlayabilirsiniz.

DOM ELEMENTLERİNİ SORGULAYIN

DOM içerisinde istediğimiz bir nesneyi bulmak bazen çok zor olabiliyor. Aslında biz kısmen bu işlemi CSS tanımlamalarında yapıyoruz. Örneğin bir sayfadaki tüm a elementlerini seçebiliyor veya belirli bir elementin içindeki farklı elementleri de buldurabiliyoruz. İşte tam da bu noktada **QuerySelector** yapıları devreye giriyor ve aynı CSS komutları gibi komutlarla DOM'u sorgulayarak sonucunda elimize HTML elementlerinin ulaşmasını sağlıyor.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title></title>
```

```

<meta http-equiv="X-UA-Compatible" content="IE=8">

<script type="text/javascript">
    function Goster() {
        var oDiv = document.getElementById
("icerik");
        var oSpan = oDiv.querySelectorAll
("div.siyah span");
        alert(oSpan[0].innerHTML);
    }
</script>

</head>
<body>
    <input onclick="Goster();" id="Button1"
type="button" value="button" />
    <div id="icerik">
        <div class="kirmizi">
            <span>Deneme1</span></div>
        <div class="siyah">
            <span>Deneme2</span></div>
        <div class="kirmizi">
            <span>Deneme3</span></div>
    </div>
</body>
</html>

```

Yukarıdaki örnekte gördüğünüz üzere sayfa içerisinde **icerik** adında bir DIV var ve bu DIV içerisinde de birden çok DIV elementleri buluyor. Bizim hedefimiz **icerik** DIV'i içindeki DIV'lerden **class** özelliği siyah olanın içindeki SPAN'in içindeki değeri almak. Bunu yapabilmek için basit bir şekilde **div.siyah** diyerek sorgumuzda CSS sınıfı **Siyah** olan DIV'Leri buluyoruz sonrasında da bir boşluk bırakarak SPAN elementlerini alıyoruz. **querySelectorAll** komutuna verdığımız sorgumuz sonrasında elimize soruya uygun elementler ulaşıyor. Bu noktadan sonra eldeki elementler ile farklı işlemler yapılabilir. **Query Selector** yapıları W3C tarafından standart olarak geliştiriliyor. Standart ile ilgili tüm detaylara aşağıdaki adresten ulaşabilirsiniz.

<http://www.w3.org/TR/selectors-api/>

TASARIMCILAR İÇİN INTERNET EXPLORER 8

Compatibility View	77
Sitemizi Nasıl Ayarlarız?	79
Özel CSS Filtreleri	80

Her yeni tarayıcı web tasarımcılar için yeni bir dert anlamına gelir. Maalesef bu durum tarayıcının standartlara uyması veya uymaması ile ilişkili değil. Bugün Internet Explorer 8.0 standartlara uygun yeni tara-ma altyapısı ile gelse de bu sefer de eski standartlara uygun olmayan sitelerimizle ilgili sorun yaşayacağımız kesin. O neden bizler için Internet Explorer 8.0’ın yeni özelliklerinden faydalananmanın önemi kadar hali hazırda var olan sitelerimizi de hemen Internet Explorer 8 ile rahatlıkla çalışır hale getirebilmek de çok önemli.

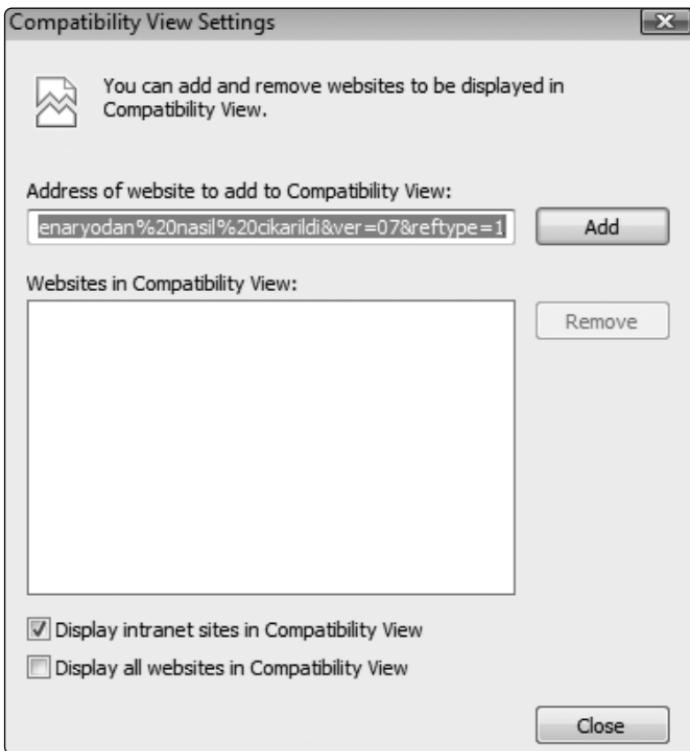
COMPATIBILITY VIEW

Internet Explorer 8.0 ile beraberaslında eski tarayıcı motorları da halen gelmekte. Bunun nedeni eski sitelerin eski tarayıcı motoru ile gösterilmesinin şart olması. Bir anda tüm sitelerin yeni motora uygun şekilde hazırlanması mümkün değil. Bu sistem **Compatibility View** adı altında farklı bir işlevsellik ile karşımıza çıkması. IE8.0 eski sistem için hazırlanmış siteleri algılayıp kullanıcıyı uyarabiliyor ve farklı tarayıcı motorları arasında geçiş tarayıcının kapatılıp tekrar açılmasını gerektirmiyor.



Compatibility View düğmesi.

“Compatibility View” düğmesine basarak eski standartlara göre açığınız bir web sitesiyle ilgili ayarı IE 8 saklayarak bir dahaki sefere aynı siteyi ziyaret ettiğinizde otomatik olarak “Compatibility View” seçeneği aktif hale getiriyor. İnternet üzerinden açılan tüm siteler normal modda çalışırken intranet üzerinden açılan tüm siteler ise otomatik olarak “Compatibility View” modunda açılıyor. Burada önemli olan bir diğer nokta da **User Agent** bilgisi. Özellikle istatistik sistemleri için ASP.NET tarafından tarayıcının sürüm bilgisinin alındığı durumlarda unutmamak gereklidir ki IE 8.0 “Compatibility View” içerisinde sunucuya **User Agent** olarak IE 7 bilgisi gönderecektir. Hangi sitelerin nasıl gösterileceğine ayrıca **Tools** menüsünden “Compatibility View Settings” kısmından ulaşabilirsiniz. Buradan ister tüm sitelerin IE 7 gibi gösterilmesini veya sadece adresini istediğiniz sitelerin IE 7 olarak açılmasını sağlayabilirsiniz.



Hangi sitelerinde “Compatibility View” modunda açılacağını belirleyebiliyorsunuz.

SİTEMİZİ NASIL AYARLARIZ?

Tavsiye edilen tabi ki tüm sitenizi gözden geçirerek IE 8'e uygun şekilde gerekli düzenlemeleri yapmanız. Fakat bu süreçte hızlı bir adaptasyon sağlamak için isterseniz sitenizin IE 8 içerisinde otomatik IE 7 motoru ile, yani "Compatibility View" içerisinde açılmasını da sağlayabilirsiniz. Bunun için iki seçenekiniz var;

Eğer tüm site bazında bu işlemi yapmak istiyorsanız **HTTP Header** olarak aşağıdaki kodu kullanabilirsiniz;

X-UA-Compatible: IE=EmulateIE7

Sitenizdeki sadece bir sayfanın bu modda çalıştırılmasını istiyorsanız, bu sefer de **Meta Tag'lar**ından faydalana bilirisiniz;

```
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />
```

IIS 7.0 üzerinde bir site için genel HTTP Header tanımlamak için aşağıdaki kodu Web.Config dosyasına koymanız yeterli olacaktır.

```
<system.webServer>
    <httpProtocol>
        <customHeaders>
            <add name="X-UA-Compatible" value=
                "IE=EmulateIE7" />
        </customHeaders>
    </httpProtocol>
</system.webServer>
```

Peki sitenizi ilk başta IE7 moduna göre çalışacak şekilde ayarladınız ve "Compatibility View" içerisinde gösterildi. Tüm istemcilerde artık siteniz bu modda gösterilecek şekilde kaydedildiği için IE8.0 ile ilgili uyumluluk sorunlarınızı halletseniz ve HTTP header bilgisi ile meta tagları kaldırırsanız da herkes hala IE 7.0 modunda sitenizi ziyaret etmeye devam edecektir. Bunu aşmanın yolu ise tüm istemcileri IE 8.0 modunda çalışmaya zorlamaktan geçiyor.

X-UA-Compatible: IE=EmulateIE8

Yukarıdaki gördüğünüz şekilde ister **HTTP Header** ayarlayın ister meta taglar kullanır hiç fark etmez. Artık siteniz kesinlikle IE8.0 modunda

gösterilecek ve istemcilerde kullanıcılar isteseler de “Compatibility View” modunu aktif hale getiremeyecekler. Eğer siteniz daha öncesinde istemicde “Compatibility View” ile gösterilecek siteler listesinde yer alıyorduysa IE 8.0 tarafından otomatik olarak o listeden de silinecektir.

ÖZEL CSS FILTRELERİ

CSS kurallarına aykırı olsa da bugüne kadar sadece Internet Explorer içerisinde çalışan filtrelemeler kullandığınızda artık Internet Explorer 8.0 ile söz konusu filtrelemelerin yazınızı değiştirmeniz gerekecektir. Gelin ilk olarak nasıl bir filtrelemeden bahsettiğimize bakalım.

```
#seffafDiv {
    filter: progid:DXImageTransform.Microsoft.Alpha(Opacity=50);
}
```

Yukarıdaki CSS sınıfındaki gibi çok sayıda farklı filtre sadece Internet Explorer içerisinde kullanılabilir şekilde tanımlanabiliyordu. Bu filtreleme sisteminin yazımındaki sorun tek satırda birden çok üst üste iki noktanın bulunuyor olması. CSS kuralları gereği bu şekilde bir yazım mümkün değil. IE 8.0 ile beraber bu yazım şekli aşağıdaki şekilde değiştiriliyor.

```
#seffafDiv {
    -ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=50)";
}
```

Böylece artık herhangi bir sorun yaşamadan rahatlıkla bu filtreleri CSS sınıflarınız arasında kullanabilirsiniz. Diğer yandan şu an için hem eski hem yeni sürüm IE tarayıcılarında kodlarınız çalışması için her iki yazımı da beraber kullanmak durumundasınız.

```
#seffafDiv {
    -ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=50)";
    filter: progid:DXImageTransform.Microsoft.Alpha(Opacity=50);
    opacity: .5;
}
```



Windows®

Internet Explorer® 8

.Son kullanıcıların gözünden IE 8 yenilikleri

.Web Tasarımcılar için notlar

.Web Dilimleri

.Görsel Arama Altyapıları

.Hızlandırıcılar

.Yazılım Geliştiricilere Araçlar

Internet Explorer kendi geçmişine perde çeken farklılıklarla dolu yepyeni bir sürüm ile bu sefer hem son kullanıcıların, hem tasarımcıların hem de yazılım geliştiricilerin hayatlarını değiştirecek güçte yenilikler ile karşımıza çıkıyor. HTML 5 gibi yeni standartların uygulanmasından başlayarak Web Dilimleri, Hızlandırıcılar gibi ekleni altyapılarına kadar bir çok parlak özelliğin bulunduğu Internet Explorer 8 tüm gücü ve detayları ile kitap içerisinde sizi bekliyor.



Daron YÖNDEM

DEVELOAD Yazılım kurucusu Daron Yöndem Microsoft tarafından 2008 ve 2009 yıllarında ASP.NET alanında Most Valuable Professional olarak seçilmiştir. Uluslararası bir konuşmacı olarak Daron Yöndem aynı zamanda Microsoft Regional Director unvanına sahiptir. Türkiye'de iki kitabı olan Daron Yöndem, International .NET Association'da Türkiye Başkanlığı ve Ortadoğu Afrika bölgesi konuşmacı ofisi başkanlığı yapmaktadır. Kendisine <http://daron.yondem.com/tr/> adresinden blogundan ulaşabilirsiniz.

Microsoft®
Regional Director
PROGRAM