



CLASSIC

AJAX

Asynchronous JavaScript and XML

EDITION: 1.0

**Emre Can
ÖZTAŞ**

Kapak Tasarımı: Emre Can ÖZTAŞ

Classic AJAX

Book Version: 1.0

Yazan: Emre Can ÖZTAŞ

Önsöz

Merhaba. Ben Emre Can ÖZTAŞ. Gazi Üniversitesi – Bilgisayar Mühendisliği bölümü öğrencisiyim. Bu kitapla birlikte yazdığım kitap sayısı 4 etti. Allah (c.c) ömür verdiği sürece yazmaya devam edeceğim. Daha önce herhangi bir kitabım elinize ulaşmamış ise belirtmek isterim. Bu kitaplardan zerre kadar kazancım yok. Öğrendiklerimi ya da bildiklerimi, belki birilerine yardımım dokunur maksadıyla yazmaktayım. Yani tamamen Allah rızası için yapılan bir çalışma. Bazılarının buna kılıf uydurmasına veya iftira atmasına gerek yok. Ki yazdığım kitapları internet üzerinden bedava dağıtmaktayım zaten.

Her kitap için “Önsöz” kısmı benim için önemli olmuştur. Belki bu kitaplardan tek kazancım; Önsöz'lerde gelecekteki “bana” mesaj göndermem. Bilmiyorum, ilerleyen yaşlarımı görürüm veya görmem Allah (c.c) bilir. Eğer görürsem geçmişte neler yapmışım ya da neler yaşamışım bunları anlamama yardımcı olacak bu kitaplar.

Kitaplar tamamen ücretsizdir. İstedığınız gibi kullanabilir, dağıtabilir, çıktısını alabilirsiniz. Sizden tek ricam; kitabın tamamı veya belli bir bölümü kaynak olarak kullanılacaksa bunu belirtmeniz. Başka herhangi bir isteğim yok sizden.

Günümüzde bir çok kitap yazılıyor ve yazılan bu kitapların bir çoğunun e-book versiyonuna ücretsiz olarak erişebilirsiniz. Bu yazarın ve yayın evinin izniyle gerçekleşir. Lakin ülkemizde böyle birşey söz konusu değildir. Ülkemizde birisi bir kitap yazdığı zaman sanki hazine olur. Ondaki başka kitap yazan yok, o en iyisi, en güzeli v.s bu böyle uzayıp gider. Bu tamamen yanlış bir düşünce. İnsan kendini üstün gördüğü kadar, yaptıklarını gözünde büyüttüğü kadar küçülür. Fakat bundan kimsenin haberi yok. Sen böyle olma adaşım. İçindeki iyiliği ve saflığı her zaman koru.

Son olarak Hz. Mevlana'nın bir sözüyle Önsöz kısmını bitirmek istiyorum.

“Nasibinde varsa alırsın karıcadan bile ders.
Nasibinde yoksa bütün cihan önüne serilse sana ters!”

Teşekkürler

Bu kitabın yazımında bana olumlu/olumsuz destek olan, hakkımda iyi veya kötü konuşan herkese teşekkür ederim. Siz olmasaydınız buralara kadar gelemez, gelişimimi tamamlayamazdım. İyi ki varsınız ve umarım hep var olmaya devam edersiniz.

Ayrıca bana yıllarca emek veren anne – babama teşekkürü bir borç bilirim. Allah sizi benim başımdan eksik etmesin. Umarım hep benimle kalırsınız.

İçindekiler

1	AJAX	6
1.1	AJAX Nedir?	6
1.2	AJAX Nasıl Çalışır?	7
1.3	Avantajları ve Dezavantajları	8
1.3.1	Avantajları	8
1.3.2	Dezavantajları	8
1.4	Araçlar	8
2	XMLHttpRequest	10
2.1	Doğru Kullanım	13
3	Request	15
4	Response	19
4.1	responseText	21
4.2	responseXML	22
5	Uygulamalar	23
5.1	responseText Uygulamaları	23
5.2	responseXML Uygulamaları	26
5.3	JSON Uygulamaları	29
6	PHP ile AJAX	34
7	JSP ile AJAX	54
8	AJAX: Çoklu Kullanım	71
	KAYNAKLAR	76

BÖLÜM 1:

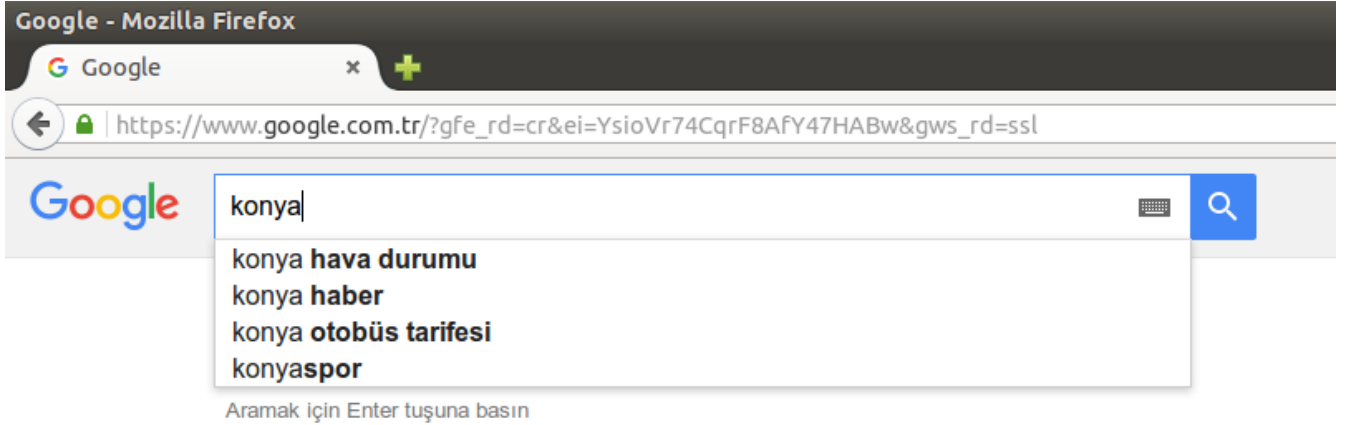
AJAX

Bu bölümde genel olarak AJAX nedir? Çalışma mantığı v.s gibi konular üzerinde duracağız. Bu ilk bölüm olduğu için kullandığımız teknoloji hakkında bilgi sahibi olmamız gerekli. Hazırsanız zaman kaybetmeden Bismillah diyerek hemen başlayalım.

1.1 AJAX Nedir?

AJAX yani Asynchronous JavaScript and XML, Türkçe'ye çevirmek istersek; Eşzamansız JavaScript ve XML, etkileşimli web sayfaları oluşturulmasına olanak sağlayan bir teknolojidir. Peki bu nedir? Örneğin elimizde bir web sayfası olduğunu düşünelim. Bu web sayfası bir form aracılığıyla yeni bir kullanıcı eklesin ya da bir sorgulama yapsın. Bu gibi durumlarda AJAX kullanılmadığında; geliştirilen uygulamanın mimarisine göre ya sayfa refresh (yenile) edilir veyahut yeni bir sayfada gösterim sağlanır. AJAX kullanıldığı durumda ise yeni bir sayfaya gerek kalmadan, web sayfasının bir kısmı veya tamamı yenilenmeye veya ekstra bir sayfalamaya gerek kalmadan değiştirilebilir. Yani bir anlamda sayfa; update (güncelleme) edilebilir.

Dediklerimden hiç birşey anlamadıysanız sorun değil. Bir örnek üzerinden gidelim. En basitinden Google'da herhangi bir dilde arama yaparken otomatik olarak kelime tamamlama veya varolan kelimeyi herhangi bir buton v.s tıklamadan getirmesi işlemi AJAX kullanılarak yapılır.



Yukarıdaki resimde de görüldüğü gibi google, herhangi bir kelime araması yapacağımız zaman bu kelimeleri tamamlama veya önerilerde bulunur. İşte kullanılan bu teknoloji AJAX'tır. AJAX kullanılmadığını düşünürsek; arama yapacağımız zaman, tamamlama özelliği çıkmaz ve aramalarımız diğer bir sayfada gösterilirdi. Buradan çıkaracağımız sonuç: AJAX çok kullanışlı bir teknoloji ve RIA (Rich Internet Application)'un olmazsa olmazıdır.

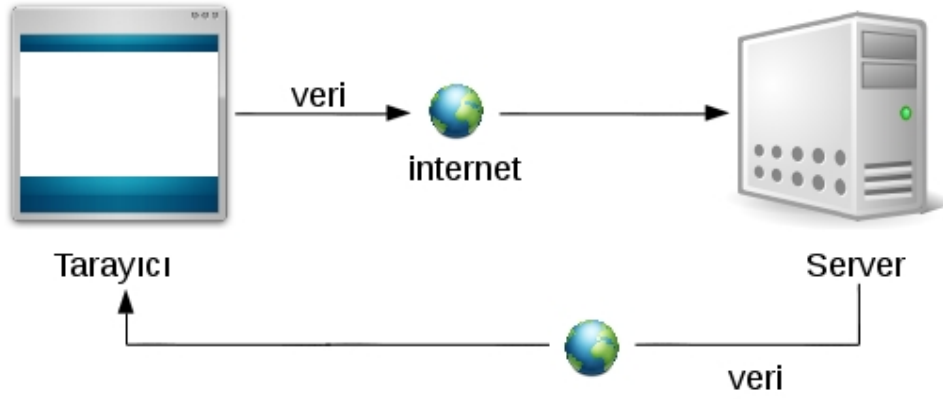
AJAX bir programlama dili değildir. Yeri gelmişken bundan da bahsedelim. AJAX, adından da anlaşılacağı gibi; JavaScript ve XML dillerinden türetilmiş bir yapı. XML dili de XHTML'den türediği için; AJAX, JavaScript ve XHTML kökenlidir diyebiliriz. Yani script bir dil. Eğer hiç HTML veya JavaScript bilginiz yoksa benden size tavsiye; hiç başlamayın. Kitabı kapatın. Çünkü bu iki kavramı bilmeden AJAX'ı öğrenmeniz biraz zor.

AJAX yeni bir teknoloji değil 2005 yılında Google'ın desteklemesi sayesinde adını duyurmuştur. Zaten Google, AJAX kullanımını daha doğrusu AJAX'ın doğru kullanılmasını önermektedir. Google, hemen hemen tüm platformlarında AJAX'ı kullanmaktadır. En basitinden bunlara örnek verecek olursak; ilk akla gelen Translate ve Maps verilebilir.

1.2 AJAX Nasıl Çalışır?

AJAX, Asynchronous Data Transfer (Eşzamansız Veri Transferi) yani HTTP isteklerini kullanarak; web sayfası ve server arasında veri iletimini sağlar. Bu veri iletimi tek yönlü olabileceği gibi çift yönlü de olabilir. Yani hem data gider hemde işlenmiş olan data gelebilir. Tabi AJAX'ı nasıl kullanacağı tamamen yazılımcıya kalmış bir şeydir.

Aşağıdaki şekilde AJAX'ın çalışma sistematığı görülmektedir.



Yukarıdaki resimde de görüldüğü gibi tarayıcı (yani kullanıcı) bir istek yaptığı zaman, AJAX ile arkaplanda bu istek server'a iletilir. Server'da işlenen veri tekrar tarayıcıya iletilir. Yani sayfanın bir bölümü veya tamamı değiştirilir. Bu değiştirilme işlemi sayfanın yenilenmesi değildir. İstenilen alanlar, değiştirilir. Sayfa aynı kalır.

Peki bu işlemi AJAX nasıl gerçekleştiriyor? HTTP istekleri ile server'a data gönderdiğini söylemiştik, daha önce. İşte bu işlemi XMLHttpRequest nesnesi ile gerçekleştirir. Şimdilik üzerinde fazla durmamıza gerek yok. İlerleyen bölümlerde detaylı olarak değineceğiz lakin şimdilik bu nesne ile veri iletimi ve veri alımını gerçekleştirildiğini bilelim.

AJAX tamamen programlama dillerinden bağımsızdır. Örneğin PHP, Java, ASP.NET, Ruby, Python v.s gibi bütün dillerle çalışabilir. O yüzden AJAX şu dil ile çalışır veya bu dil ile daha performanslı çalışır diye bir şey söz konusu değildir.

AJAX için bir programlama dili değildir demiştik. Hala sözümüzün arkasındayız. AJAX aslında bir ekibin parçasıdır diyebiliriz. Nasıl mı?

- Sunum için: XHTML ve CSS kullanılır.
- Dinamik gösterim ve etkileşim için: DOM (Document Object Model) kullanılır.
- Veri taşınması ve değişimi için: XML ve XSL kullanılır.
- Tarayıcı istekleri için XMLHttpRequest nesnesi kullanılır.
- Buraya kadar olan herşeyi de JavaScript yapar.

İşte yukarıdaki sıraladığımız maddeler AJAX'ın dayandığı temelleri göstermektedir.

1.3 Avantajları ve Dezavantajları

AJAX'ın avantajları olduğu gibi bir çok dezavantajı da bulunmaktadır. Lakin AJAX kullanımı geliştiriciye kalmış bir durumdur. Doğru kullanıldığı durumlarda oldukça etkili web sayfaları elde etmek mümkündür. Yani dezavantajları kontrol altına alabilmek önemlidir. Bu da programcının kalitesine kalmış bir durumdur.

Peki AJAX'ın avantajları ve dezavantajları nelerdir şimdi bunlara bakalım.

1.3.1 Avantajlar

- AJAX kullanılması durumunda; sayfa yenilenmesi veya yeni sayfaların çalıştırılması gerekmeceği için sunucu boşuna iş yapmış, yorulmuş olmaz.
- Sayfanın yenilenmemesi veya başka bir sayfanın yüklenmemesi durumunda Bandwith (Bant Genişliği) kullanılmayacağı için yani yeni sayfalama olmayacağından dolayı, adresin trafiği rahatlatılmış olur.
- AJAX nesnelere kolay entegre edilebilir. Bir defa oluşturmuş olan nesnelere birden fazla sayfa için kullanılabilir.
- Sayfalar yeniden yüklenmeyeceği için kullanıcı gereksiz yere bekletilmez.
- Kullanıcının form doldurduğu bir sayfada; sayfa yenilenmesi olmayacağı için kullanıcı boş yere aynı bilgilerle yeniden uğraşmaz zorunda kalmaz.
- Bu maddeden daha önce bahsetmiştik lakin tekrar bahsedelim: AJAX herhangi bir dile bağlı değildir. Bu yüzden istenilen dil ile etkileşimli olarak çalışabilir.

1.3.2 Dezavantajları

- Arama motorları ile uyumsuz çalışır. Yani arama motorları AJAX nesnelere algılayamaz veya algılamakta oldukça zorlanır. Dolayısıyla arama motorlarından alınacak olan trafiğin önemsiz olduğu ortamlarda kullanılması önerilir.
- AJAX'ın en büyük dezavantajı ise: tarayıcı uyumsuzluğu. AJAX bir çok tarayıcı ile çalışabilir lakin eski tarayıcılarda bazen sorun çıkarabilir. O yüzden bu konuda dikkatli olunması lazım veya eski tarayıcı kullanan kullanıcılar gözardı edilmelidir.
- AJAX arkaplanda çalıştığı için kullanıcının web sayfası içerisinde attığı adımların takip edilmesi zordur. Şayet kullanıcının her yaptığını kontrol altına almak istiyorsanız. Ama bununda bir çözümü var. genellikle Google Analytic ile bir web sayfasında kullanıcının attığı her adım takip edilebiliyor.
- Klasik olarak JavaScript ortamında yazılan AJAX, dikkatli kodlanmadığı zaman; karışıklığa sebep olabilir. O yüzden ya iyi bir tasarım yapmalısınız veya bir tasarım şablonu kullanmalısınız veyahutta JavaScript Framework'lerinden birisini tercih etmelisiniz. Şuan günümüzde kullanılan bir çok JavaScript Framework'ü ile AJAX kodlaması basit bir şekilde halledilebiliyor. Bunun en önemli örneği; JQuery. Lakin biz bu kitap boyunca, klasik JavaScript ile AJAX kullanımından bahsedeceğiz. Buradaki amacımız en azından bir temel oluşturabilmek ya da işin mutfağında bulunmaktadır.

1.4 Araçlar

AJAX kullanımı için herhangi bir araca ihtiyaç yoktur. Neden? Çünkü AJAX, JavaScript tabanlıdır. Web sayfaları da şu 3 şeyden anlarlar; HTML, CSS ve JavaScript. Dolayısıyla herhangi bir kurulum veya araç gerekmez. Burada bana göre dikkat edilmesi gereken en önemli nokta; düzgün bir kod editörü kullanmaktır. Şuan benim gördüğüm kadarıyla etkili bir JavaScript kod editörü bulunmamaktadır. Ama yine de siz bir kaç tavsiye de bulunmak isterim.

Eğer herhangi bir ücret karşılığında bir editör kullanmak isterseniz; WebStorm çok kalitelidir. Fiyatı şuan

itibari ile 129\$. Verdiğiniz paranın hakkını sonuna kadar alacağınıza eminim. WebStorm bir JetBrains ürünüdür. Şayet öğrenci iseniz; öğrenci e-posta adresinizle 1 yıllık ücretsiz sahip olma hakkınız var. Bence kullanın. Bunun dışında eğer ödemeyi gerçekleştirip WebStorm'a sahip olursanız; her güncelleme geldiğinde ekstra olarak ücret talep edilmesi söz konusudur. Bunu da dikkate almanızı öneririm.

WebStorm:

<https://www.jetbrains.com/webstorm/>

Ben genelde Open Source (Açık Kaynak) araçlar kullanılmasından yanayım. Crack'li kullanımı kesinlikle tavsiye etmem. En azından bu olaya benim bakışım: Kul Hakkı. Boşuna karşınızdakilerin haklarına girmeye kendinizi heba etmeyin. Ahmet Kaya'nın bir şarkı sözünde söylediği gibi: "Kısa çöp uzun çöpten hakkını alır elbette". Yani o girdiğiniz hakkın bir gün acısını çıkarırlar. O yüzden açık kaynak yazılımları kullanın. Çok kaliteli açık kaynak yazılımlar var. Saymakla bitmez. Ben size bir kaç tanesini önermek istiyorum.

Geany:

<http://www.geany.org/>

SublimeText:

<http://www.sublimetext.com/>

Atom Editor:

<https://atom.io/>

Yukarıdaki editörlerden herhangi birisini seçebilirsiniz. Ben genelde web projelerinde NetBeans tercih ederim. Çünkü çok kaliteli. Hatta şunu da söyleyebilirim: piyasada bir çok ücretli yazılımdan daha iyi. Örneğin ben daha önce PHP konusunda, NetBeans kadar iyi bir araç görmedim. Java konusunda da çok iyi. NetBeans'ı da tercih edebilirsiniz.

Buraya kadar editörlerden bahsettik ve bir kaç editör tavsiyesinde bulduk lakin bunları seçmek tamamen size kalmış. Eğer herhangi bir ortamda geliştirme yapıyorsanız o ortamda geliştirme yapmaya devam edin. Ekstra herhangi bir araç edinmenize gerek yoktur. Bu kitap boyunca Eclipse kullanacağım. Bunu da belirteyim.

BÖLÜM 2:

XMLHttpRequest

Bu bölümle birlikte artık AJAX kodlamaya başlayabiliriz. Bir önceki bölümde AJAX'ın HTTP istekleri kullanarak `server`'la iletişim kurduğundan bahsetmiştik. Bu istekleri de `XMLHttpRequest` nesnesi ile sağladığını belirtmiştik. Peki bir `XMLHttpRequest` nasıl oluşturulur ilk olarak bununla başlayalım.

Aşağıdaki şekilde bir `XMLHttpRequest` oluşturulabilir.

```
var ajaxObject = new XMLHttpRequest();
```

Yukarıdaki şekilde bir `XMLHttpRequest` oluşturduk. Yukarıdaki şekilde oluşturduğumuz nesne bir çok tarayıcı tarafından tanınmaktadır. Bu tarayıcılar aşağıdaki gibidir.

- Opera (7.6 ve üstü)
- Mozilla Firefox (1.0 ve üstü)
- Chrome (veya Chromium)
- Safari (1.2 ve üstü)
- IE (7 ve üstü)

Eğer kullanıcı yukarıdaki tarayıcılardan herhangi birisini kullanır ise (Listede olmayan başka tarayıcılar da var) herhangi bir sıkıntı olmadan oluşturmuş olduğumuz nesnemizi tanıyacaktır. Peki kullanıcı eski bir IE tarayıcısı kullanıyorsa? O zaman `XMLHttpRequest` yerine farklı bir sınıftan nesne oluşturmamız gerekecektir. Aşağıdaki örnek kodumuza bakalım.

```
var ajaxObject= new ActiveXObject("Microsoft.XMLHTTP");
```

Yukarıdaki kodumuz IE5 ve IE6 içindi. Eğer bunlar dışında herhangi bir IE sürümü kullanıyorsa aşağıdaki kodu kullanmamız gerekir.

```
var ajaxObject = new ActiveXObject("Msxml2.XMLHTTP");
```

Buraya kadar herhangi bir sıkıntı yok. Peki kullanıcının hangi tarayıcıyı kullandığını nerden bileceğiz de ona göre destek vereceğiz? Bu işleme göre de kodlarımızı düzenleyeceğiz. Burada iki farklı metotla kodlarımızı yazabiliriz. Kullanıcımızın tarayıcısı hangisi ise AJAX nesnemizi ona göre oluşturacağız.

İlk kullanım stilimizle başlayalım.

```
var ajaxObject;  
try {  
    ajaxObject = new XMLHttpRequest();  
} catch (err) {  
    try {  
        ajaxObject = new ActiveXObject("Msxml2.XMLHTTP");  
    } catch (err) {  
        try {  
            ajaxObject = new ActiveXObject("Microsoft.XMLHTTP");  
        }  
        catch (err) {  
            document.write(err.message);  
        }  
    }  
}
```

```
}  
}  
}
```

Yukarıdaki örnek kodlarımızda; kullanıcının tarayıcısı hangi nesneyi destekliyorsa, try - catch ile kontrolünü yaptık. Kodlarımızı çalıştırdığımız anda hangi tarayıcı çalıştığını artık öğrenebiliriz ve ona göre işlem yapabiliriz.

Herhangi .html veya .htm uzantılı bir dosya oluşturarak yukarıdaki kodlarımızı <script></script> etiketleri arasında yazıp deneyeceğiz.

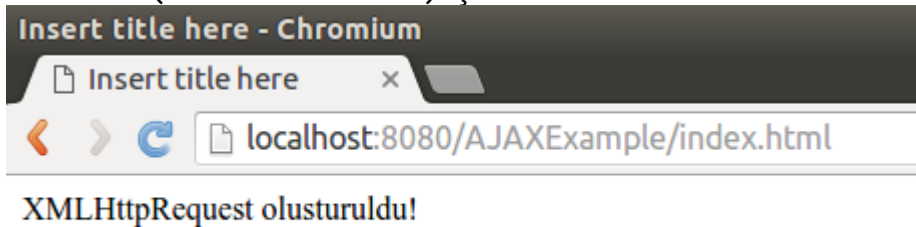
Şimdi yukarıdaki kodlarımızın arasına document.write() satırını da ekleyelim ve tarayıcımızın hangi nesneyi desteklediğine bakalım. Son olarak örnek kodlarımız aşağıdaki gibi olacaktır.

```
var ajaxObject;  
try {  
    ajaxObject = new XMLHttpRequest();  
    document.write("XMLHttpRequest oluşturuldu!");  
} catch (err) {  
    try {  
        ajaxObject = new ActiveXObject("Msxml2.XMLHTTP");  
        document.write("Msxml2.XMLHTTP oluşturuldu!");  
    } catch (err) {  
        try {  
            ajaxObject = new ActiveXObject("Microsoft.XMLHTTP");  
            document.write("Microsoft.XMLHTTP oluşturuldu!");  
        }  
        catch (err) {  
            document.write(err.message);  
        }  
    }  
}
```

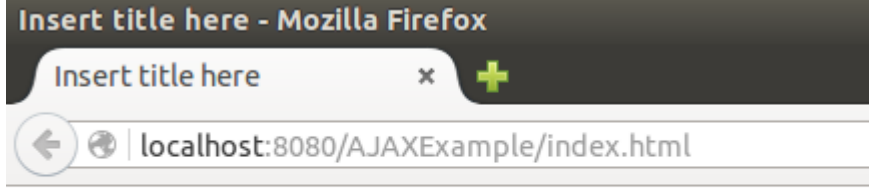
Yazmış olduğumuz örneğimizin ekran çıktısına bir bakalım. Belki de tarayıcımız desteklemiyordur, belli olmaz. Ben GNU / Linux dağıtımlarından Ubuntu kullandığım için sistemimde; Chromium (Chrome'in GNU / Linux sistemler için uyarlaması), Midori, Mozilla Firefox ve Arora tarayıcıları var. 4 farklı tarayıcı ile test yapayım ve sonuçları görelim. Siz de isterseniz; özellikle Windows kullanıcıları IE ile test yapıp sonuçları görebilirsiniz.

Ekran çıktılarımız aşağıdaki gibi olacaktır.

Chromium (V: 47.0.2526.106) için:

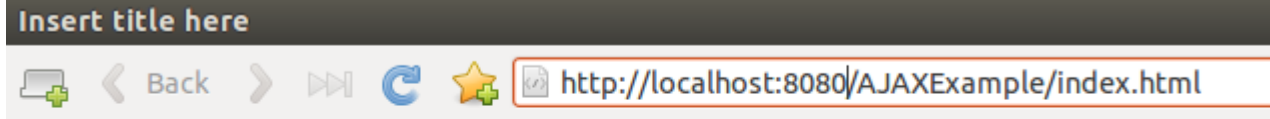


Mozilla Firefox (V: 43.0.4) için:



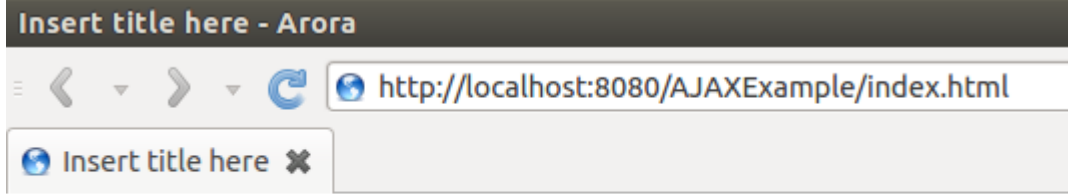
XMLHttpRequest oluşturuldu!

Midori (V: 0.4.3) için:



XMLHttpRequest oluşturuldu!

Arora (V: 0.11.0) için:



XMLHttpRequest oluşturuldu!

Görüldüğü gibi 4 tarayıcı da XMLHttpRequest nesnesini tanıdı. Daha öncede dediğim gibi farklı tarayıcılar üzerinde bu testi yaparak AJAX nesnesinin oluşturulup oluşturulmadığını, oluşturulursa hangi nesnenin oluşturulduğunu öğrenebilirsiniz.



Tarayıcı çubuğundaki adres sizi yanıltmasın. Ben Eclipse üzerinde çalıştığım için bir HTML5 projesi oluşturdum. Apache Tomcat kullandığım için 8080 no'lu portta çalışıyorum. Herhangi bir .html uzantılı bir dosyaya yukarıdaki kodlarımızı yazıp deneyebilirsiniz.



Chromium, Chrome'in GNU / Linux için uyarlamasıdır. Yani Chrome üzerine kurulu bir yapı fakat açık kaynaklı olarak geliştiriliyor. Hatta Chrome'den daha performanslı çalıştığını söyleyebilirim. Chrome'ın Web Store mağazasına bağlanabilir ve Chrome için olan extension (eklenti)'leri kullanabilirsiniz.

Şimdi gelelim diğer bir yöntememize. Diğer yöntemimizde if'le kontrolü içermektedir. Dilerseniz switch - case ile yapabilirsiniz lakin pek tavsiye edilmez.

```
var ajaxObject;  
if (window.XMLHttpRequest) {  
    ajaxObject = new XMLHttpRequest();  
} else if(window.ActiveXObject){  
    ajaxObject = new ActiveXObject("Microsoft.XMLHTTP");  
} else {  
    document.write("Tarayiciniz AJAX'i desteklemiyor!");  
}
```

Yukarıdaki örneğimizde de görüldüğü gibi window nesnesi ile tarayıcımızın hangi nesneyi oluşturup oluşturmadığını kontrol ettik. Lakin bu kodlar arasında eksik olan bir kod var: Msxml2.XMLHTTP. Microsoft.XMLHTTP ve Msxml2.XMLHTTP, ikisi de window.ActiveXObject üzerinde işlem görmekte olduğu için ikisinin kontrolü if ile sağlanamamaktadır. O yüzden try - catch yapısını kullanmak daha mantıklı. Şayet daha eski tarayıcılara da destek verilecekse.

2.1 Doğru Kullanım

Bu başlık altında size başka hiç bir yerde göremeyeceğiniz bir kullanım stilinden bahsedeceğim. Giriş bölümümüzde bir AJAX nesnesi nasıl oluşturulur, detaylıca bahsettik. Her seferinde bu kodları yazmak zorundayız. Peki bunu bir kere yazıp kullansak? Daha mantıklı değil mi? Bence çok mantıklı. Bu kitabı yazarken çeşitli kaynaklara baktım fakat hiç kimse benim kullandığım stilde kullanmamış. Ee o da benim bir programcılık yeteneğim olsa gerek. Herneyse fazla kasılmaya gerek yok. Gelin bu yazmış olduğumuz kodları bir fonksiyona bağlayalım. Bu fonksiyonu da ayrı bir .js uzantılı olarak kaydedelim. Sonra ihtiyacımız olduğunda çağırıp kullanalım.

Bahsetmiş olduğum yöntemi kullanmak; hem ekstra kod karmaşıklığını önleyecek hemde yazmış olduğumuz kodları her seferinde tekrar yazmayacağız. Ayrıca bu kodları ezberlemeye de gerek yok. Programcılık hayatımda hep ezberden uzak durdum. Ezber yapılmasına karşı oldum. Lakin bu zamana kadar okulda bana gösterilen; belli yöntemlerin ezberletilmesi idi. Sistemi eleştirmiyorum. Sistemi eleştirmekle hiç bir yere varılmaz. Herkes yine kafasına bildiği gibi hareket eder. O yüzden insan kendi bildiğini uygulamalı.

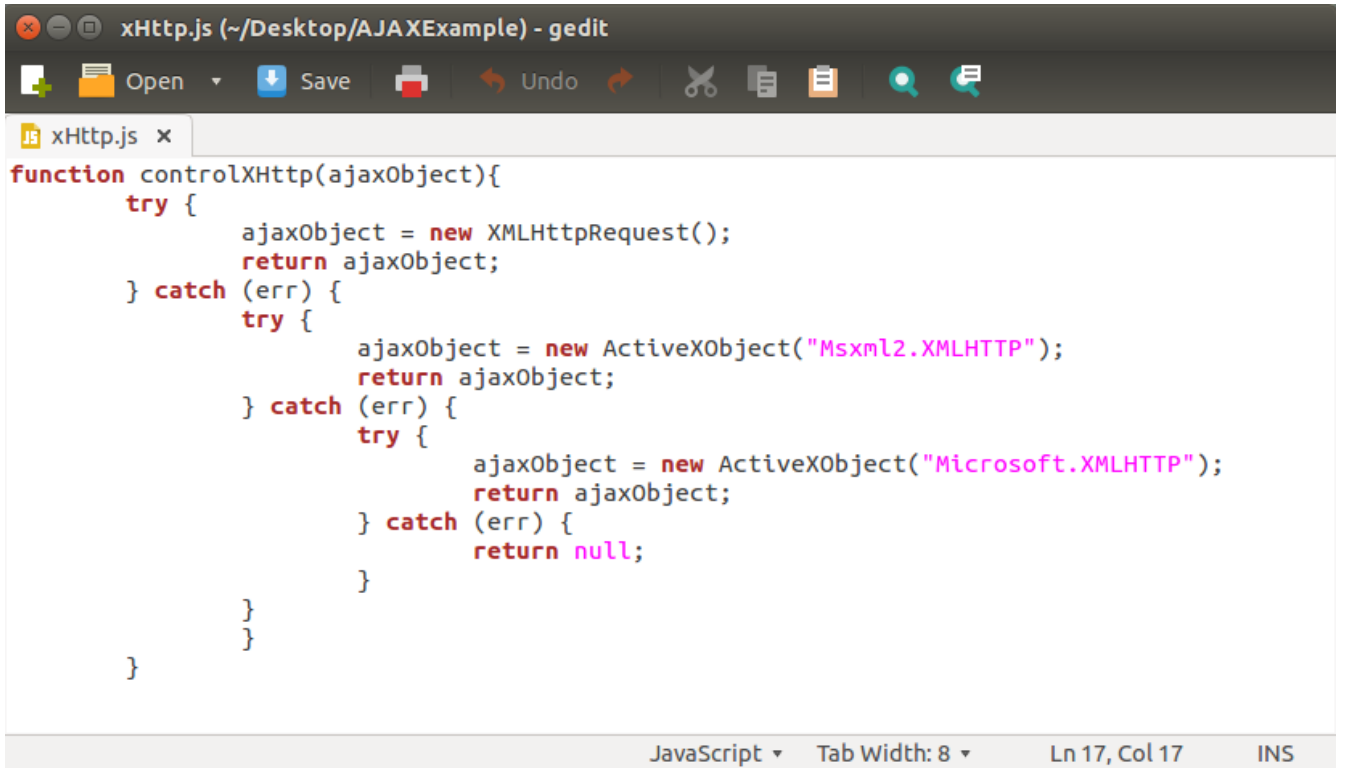
Lafı çok fazla uzattık. Bir .js uzantılı dosya açalım. Adını herhangi bir kelime veya kelime grubu olarak verebilirsiniz. Ben kısaca: xHttp.js dedim. Bu dosyamızın içerisinde aşağıdaki kodlarımızı yazalım ve kaydedelim.

```
function controlXHttp(ajaxObject) {
    try {
        ajaxObject = new XMLHttpRequest();
        return ajaxObject;
    } catch (err) {
        try {
            ajaxObject = new ActiveXObject("Msxml2.XMLHTTP");
            return ajaxObject;
        } catch (err) {
            try {
                ajaxObject = new
                ActiveXObject("Microsoft.XMLHTTP");
                return ajaxObject;
            } catch (err) {
                return null;
            }
        }
    }
}
```

Yukarıdaki kodlarımızı yazıp kaydettikten sonra artık istediğimiz herhangi bir sayfada bu sayfamızı çağırıp kullanabiliriz. Sizden ricam bu oluşturmuş olduğumuz dosyamızı saklamanız. Çünkü ilerde de işinize yarayacak. O yüzden Code Library (Kod Kütüphanesi)'inize koyun, bence. Kod Kütüphanesi de nedir demeyin. Her deneyimli programcının bir kod kütüphanesi olur ve oraya işine yarayacak olan kodları koyar. Daha sonra da oradan alıp kullanır. Eğer bir kod kütüphaneniz yoksa benden size tavsiye: hemen bugün bir tane oluşturun.

xHttp.js dosyamızın ekran görüntüsünü de eklemek istiyorum. Daha önce de belirttiğim gibi bu dosyamızı

kaybetmeyin. Örneklerimizi bu dosya üzerinden anlatmaya çalışacağız.



```
function controlXHttp(ajaxObject){
  try {
    ajaxObject = new XMLHttpRequest();
    return ajaxObject;
  } catch (err) {
    try {
      ajaxObject = new ActiveXObject("Msxml2.XMLHTTP");
      return ajaxObject;
    } catch (err) {
      try {
        ajaxObject = new ActiveXObject("Microsoft.XMLHTTP");
        return ajaxObject;
      } catch (err) {
        return null;
      }
    }
  }
}
```

Son olarak bir HTML sayfasında, harici bir JavaScript kodu nasıl çalıştırılır ona değinelim. Bir HTML sayfasında; <head></head> etiketleri arasında:

```
<script type= "text/javascript" src="JSDosyaYolu"></script>
```

Şeklinde çağrılabilir.

BÖLÜM 3:

Request

Önceki bölümde bir XMLHttpRequest nesnesi nasıl oluşturulur ve tarayıcı desteği nasıl sağlanır bundan bahsetmiştik. Bu bölümde oluşturmuş olduğumuz XMLHttpRequest nesnesini kullanarak Request (İstek) yapacağız. Bir istek nasıl yapılır detaylı olarak bundan bahsedeceğiz. Bir sonraki bölümde Response (Cevap) alıp işlemlerimizi gerçekleştireceğiz.

Peki bu isteği kime yapacağız? Tabiki Server'a. Şimdilik Local (Lokal)'de çalıştığımız için daha doğrusu herhangi bir programlama dili çalışmadığımız için, Server'a istek yapmamıza gerek yok. İlerleyen bölümlerde bir server'a da istek şeklinden bahsedeceğiz. Lakin mantık hep aynı. Yani yazacağımız kodlarımızda herhangi bir değişiklik olmayacak.

Adım adım giderek işin mantığını anlamaya çalışalım. Çünkü AJAX'ı öğrenmek ilk başlarda biraz karışık gelebiliyor. O yüzden adım adım gidersek daha iyi anlayacağımıza eminim. Bu yüzden bir önceki bölümde oluşturduğumuz xhttp.js dosyamızı unutalım ve kodlarımızı yeniden yazalım.

İlk olarak nesnemizi oluşturalım ve tarayıcı desteğini sağlayalım.

```
var ajaxObject;
try {
    ajaxObject = new XMLHttpRequest();
} catch (err) {
    try {
        ajaxObject = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (err) {
        try {
            ajaxObject = new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch (err) {
            document.write(err.message);
        }
    }
}
```

Şimdi AJAX ile bir istek nasıl yapılır buna değinelim. Aşağıdaki şekilde bir AJAX isteği yapılır.

```
ajaxObject.open("GetVeyaPost","dosyaVeyaAdresYolu", trueYadaFalse);
```

Yukarıdaki şekilde bir istek gönderilir. Peki yukarıdaki kavramlar nelerdir? Bunlardan da bahsedelim. İlk olarak oluşturmuş olduğumuz ajaxObject nesnesinin open() fonksiyonunu kullanmalıyız. Bunu bir dosyanın veya yolunun açılması gibi düşünebilirsiniz. Yani istek yapacağımız dosyayı veya adresi open() fonksiyonu ile açacağız. open() fonksiyonu 3 parametre alır. Bunlardan ilki HTTP üzerinden hangi method'la bağlanacağımıza karar vermek. Örneğin GET veya POST methodlarından herhangi birisi ile bağlantı sağlanabilir. Bunlar dışında diğer HTTP method'ları kullanılabilir lakin lakin biz tamamen bu iki kavram üzerinde duracağız. Bu iki kavram arasında oldukça büyük farklar var. Bu farklardan kısaca bahsedelim.

GET: HTTP üzerinde default (varsayılan) olarak kullanılan bir method'tur. Neden? Çünkü GET metodu çok hızlı çalışır ve basit bir yapısı vardır. Bilgi gönderimi açık bir şekilde yapılır yani bu bilgiyi herkes görebilir ve okuyabilir. Bu bakımdan oldukça güvensizdir. GET'i basit işlemlerde kullanmamız

gerekir. Örneğin sisteme bir kullanıcı kaydedileceği zaman veya bir kullanıcının giriş kontrolü yapılacağı gibi hayati durumlarda GET kullanmak sistemi üçüncü şahısların eline vermekle aynı şeydir.

POST: HTTP üzerinde çalışan en güvenli method'tur. GET'e göre daha yavaş çalışır. Gereksiz güvenlik açıkları oluşturmaz. Gönderilecek olan verinin büyüklük bakımından sınırı yoktur. O yüzden büyük verilerin iletiminde POST metodu kullanılmalıdır.

İkinci parametrede ise dosya veya adresi belirtmemiz lazım. Yani şu adrese bağlan dememiz gerekiyor. O yüzden bu alanı adres vermek için kullanacağız. Birazdan değineceğimiz gibi; şayet GET ile çalışıyor isek adres yolunda parametreleri de belirtmemiz gerekir. Şimdilik buna fazla takılmayalım.

Üçüncü parametre ise 2 değer almaktadır. True veya False. True, bir sonraki bölümde göreceğimiz onreadystatechange özelliği aktif olunca veri erişimini sağlar. Yani adrese bağlandığımız anda çalışmaya başlayacaktır. False ise server aktif olana kadar çalışmaz. Eğer server meşgulse sıraya ekler ve beklemeye başlar. Server ne zaman hazır olursa o zaman çalışmaya başlar. Bu dediklerimiz şimdilik çok farazi biliyorum lakin bölümün sonlarına doğru daha iyi anlayacağınızdan eminim. Son olarak; True veya False ifadelerinden tercihimiz her zaman True'dan yana olacak. False kullanılması pek önerilmez. Lakin bazı durumlarda kullanımı önerilir. Örneğin yoğun trafik olan bir server'da false kullanmak daha doğru olacaktır.

Buraya kadar herhangi bir problem yok. Şimdi kodlarımızı birleştirelim.

```
var ajaxObject;
try {
    ajaxObject = new XMLHttpRequest();
} catch (err) {
    try {
        ajaxObject = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (err) {
        try {
            ajaxObject = new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch (err) {
            document.write(err.message);
        }
    }
}

ajaxObject.open("GetVeyaPost","dosyaVeyaAdresYolu", trueYadaFalse);
```

Peki bundan sonra ne olacak? Geriye sadece isteğimizi göndermek kaldı. İstedikimizi de aşağıdaki şekilde gerçekleştiriyoruz.

```
ajaxObject.send( );
```

Yukarıdaki kullanım sadece GET methodu ile çalışırken kullanılır. Aslında sadece GET metodu demek yanlış olur. Ya da şöyle bir örnekle durumu açıklamaya çalışayım. Örneğin daha önce bahsettiğimiz gibi herhangi bir kullanıcının kayıt işleminde veya kullanıcı sorgulama işleminin POST metoduyla sağlanması gerekir. İşte burada POST ile işlem yaparken kullanılacak olan parametreler send () metodunda verilir. Sanırım aşağıdaki örneğimizi incelediğinizde ne demek istediğimi daha iyi anlayacaksınız.

```
ajaxObject.open("GET", "ksorgula.php?kadi=emre&ksifre=123", true);
ajaxObject.send();
```

Yukarıdaki satırımızda; ksorgula.php dosyasına kadi, emre ve ksifre, 123 olan parametrelerini

GET metodu ile gönderdik. Eğer parametre gönderimini hakkında en ufak bir bilginiz yok ise sorun değil. Biraz bahsedelim. GET ile parametre iletimi aşağıdaki şekilde yapılır.

```
WebAdres?parametre1=parametre1Deger& parametre2=parametre2Deger
```

Yukarıdaki örneğimizde; WebAdres alanı sayfanın adresidir. Sayfa sonuna ? İşareti konulup gönderilecek olan parametre yazılmalı ve = konulup bu parametrenin değeri yazılmalıdır. Eğer gönderilecek olan parametre birden fazla ise her parametre arasına & işareti konulmalıdır. Bu yöntem ile parametre gönderimi gerçekleştirilir. Bu yöntem sadece GET metodunda böyledir. POST metodun parametre iletimi yukarıdaki şekilde yapılmaz. Çünkü gönderilecek olan parametreler gizlenir.

Sondan bir önceki örneğimizde de gönderim yapılabilir. Lakin güvenlik açığı oluşur bazı durumlarda bazı işlerin POST metoduyla yapılması daha evladır demiştik. Peki yukarıdaki gibi bir ifadeyi POST metoduyla nasıl kullanabiliriz? Hemen göstereyim.

```
ajaxObject.open("POST", "ksorgula.php", true);
ajaxObject.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
ajaxObject.send("kadi=emre&ksifre=123");
ajaxObject.send();
```

Şeklinde olmak zorundadır. send () fonksiyonu isteğimizi server'a iletacaktır. Peki setRequestHeader'da nerden çıktı? Dediğinizi duyar gibiyim. Sakin olun şimdilik bunu bir kenara bırakalım. Üzerinde fazla düşünmeyelim. Biraz sonra detaylı olarak bahsedeceğiz.

Herneyse, kodlarımızı tamamlayalım. Kodlarımız aşağıdaki gibi olacaktır.

```
ajaxObject.open("GetVeyaPost","dosyaVeyaAdresYolu", trueYadaFalse);
// POST metodu kullanılırsa asagidaki satir eklenmelidir.
ajaxObject.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
ajaxObject.send(Post ise parametreler);
ajaxObject.send();
```

Yukarıdaki kodlarımız sadece metin bazlı işlemleri gerçekleştirmek içindi. Şayet elimizde bir MIME (Multipurpose Internet Mail Extensions) varsa? O zaman kodlarımız arasına bir kod satırı daha eklememiz gerekir. Kodlarımıza geçmeden önce MIME'den bahsedelim. Bir MIME data herhangi bir veri olabilir. Örneğin; sıkıştırılmış bir dosya, flash uygulaması, video, müzik v.s yani metin dışında aklınıza gelebilecek her türlü veri. MIME veri tiplerine bakmak isterseniz aşağıdaki adrese bir gözetmanızı öneririm.

<http://www.freeformatter.com/mime-types-list.html>

Aşağıdaki kod parçasıyla MIME verisinin server'a iletilmesini sağlayabiliriz.

```
ajaxObject.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
```

Şeklinde olacaktır.

Şimdi POST metoduna geri dönelim. MIME tipindeki verileri göndermek için kullandığımız satırı, POST metodu ile çalışırken kullanıyoruz. Acaba neden? GET metodu basit işlemler için kullanılan bir metottur. Bundan bahsetmiştik. POST metodu ise GET metoduna göre daha spesifik bir yapıdadır. Veri

gönderiminin sınırı yoktur ve MIME tipindeki veriler AJAX yapısında ancak POST metodu ile gönderilebilir. Dolayısıyla POST metodunu kullandığımız anda sistem bunu bir MIME tipi gibi algılayacak ve hemen bu satırı arayacaktır. Bu satırı yazmadığımız zaman kodlarımız çalışmaz ya da kısaca POST ile işlem yapmaz. Parametreleri almaz ve hata verir. O yüzden POST metodu kullanılırken bu satırın eklenmesi çok önemlidir. Buna dikkat edelim.

Şimdi tüm kodlarımızı biraraya getirelim. Kodlarımızın son şekli aşağıdaki gibi olacaktır.

```
var ajaxObject;
try {
    ajaxObject = new XMLHttpRequest();
} catch (err) {
    try {
        ajaxObject = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (err) {
        try {
            ajaxObject = new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch (err) {
            document.write(err.message);
        }
    }
}

ajaxObject.open("GetVeyaPost","dosyaVeyaAdresYolu", trueYadaFalse);
// POST metodu kullanılırsa aşağıdaki satır eklenmelidir.
ajaxObject.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
ajaxObject.send(Post ise parametreler);
```



GET metodunu kullanırken send() fonksiyona parametre yazmak zorunda değilsiniz. Lakin null parametresini de kullanabilirsiniz. Her iki kullanımda doğrudur. Herhangi bir hata ile karşılaşmazsınız.

Birinci Kullanım:

```
ajaxObject.open("GetVeyaPost","dosyaVeyaAdresYolu",
trueYadaFalse);
ajaxObject.send();
```

İkinci Kullanım:

```
ajaxObject.open("GetVeyaPost","dosyaVeyaAdresYolu",
trueYadaFalse);
ajaxObject.send(null);
```

BÖLÜM 4:

Response

Bir önceki bölümde bir Request (İstek) nasıl yapılır değinmiştik. Şimdi de Response (Cevap) nasıl alınır ve kullanılır bundan bahsedeceğiz.

Öncelikle; 2 farklı şekilde istek alınabilir. Bunlar: Text ve XML şeklinde. JSON olarakta istek alınabilir lakin şimdilik buna değinmeyeceğiz. Çünkü JSON ile dönen veriler Text mantığındadır. Bundan ilerleyen bölümlerde detaylı olarak değineceğiz. Text ve XML şeklinde veri nasıl alınırndan önce bilmemiz gereken bir kaç kavram var. İsteği yaptığımız zaman, istek yapılan adresin durumunu kontrol etmemiz gerekiyor. Bu kontrol işlemi de onreadystatechange ile sağlanır. Yani;

```
ajaxObject.onreadystatechange
```

Peki herşey bu kadar basit mi? Tabiki değil. Bu fonksiyonu dinlememiz gerekiyor. Peki dinleme işlemini nasıl yapmamız lazım? Anonymous (Anonim) bir fonksiyonla. Yani bu fonksiyona başka bir fonksiyon kullanarak dinlememiz gerekiyor. Peki anonim fonksiyon nedir? Anonim fonksiyonların isimleri olmaz. Yani bir nevi Listener (Dinleyici) gibi düşünebilirsiniz. Aşağıdaki kodlarımızda olduğu gibi.

```
ajaxObject.onreadystatechange = function(){  
}
```

Yukarıdaki kodlarımız genel olarak bir dinleyici. Basit olarak istek yapılan adresi dinliyor. Gelen cevapları alıp bize iletiyor. Bu böyle kullanılabilceği gibi genel olarak iki farklı parametrelerin kontrolü ile kullanımı daha evladır. Peki bu parametreler nelerdir? Bu parametreler; readyState ve status. readyState basit olarak; istekleri ve cevapları kontrol eder, diyebiliriz. status ise adresin durumunu kontrol eder. Yani istek yapılan adres çalışıyor mu? Herhangi bir problem var mı gibi yönlerden kontrol eder. Kodlarımızı yeniden düzenleyecek olursak;

```
ajaxObject.onreadystatechange = function(){  
    if (ajaxObject.readyState == readyStateDurum && ajaxObject.status  
    == statusDurum) {  
        // yapılacaklar  
    }  
}
```

Yukarıdaki yazmış olduğumuz kodlarımız tam bir kod aslında. AJAX'ın %80'ini öğrendik sayılır. Buraya kadar herhangi bir sıkıntı yoksa kendinizi tebrik edebilirsiniz.

Gelin şimdi yukarıdaki yazmış olduğumuz kodlarımızı inceleyelim. Öncelikle readyState 5 farklı durumda bulunabilir. Bu durumlar aşağıdaki gibidir.

readystateDurum	Açıklama
0	İstek başlatılamadı!
1	Sunucu bağlantısı kuruldu!
2	İstek alındı!
3	İstek işleniyor!
4	İstek yapıldı ve cevap hazır!

readystate yukarıdaki durumlardan herhangi birini alabilir. Ya da şöyle söyleyim; hangi durumu kontrol etmek istiyorsanız ona uygun durumu yazabilirsiniz. Örneğin; büyük bir web uygulamasında her durumu kontrol edip kullanıcıya iletmek isteyebilirsiniz. Böyle bir durumda farklı fonksiyonlar ve dinleyiciler oluşturmanız gerekecektir.

status bir çok farklı durumda olabilir. Bu durumların listesine aşağıdaki adresten ulaşabilirsiniz.

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

Yukarıdaki adresten kullanmak ya da kontrolünü yapmak istediğiniz durumları seçebilirsiniz. Lakin genel olarak 2 farklı status kontrolü yapılır. Aşağıdaki tablomuza bakalım.

statusDurum	Açıklama
200	Bağlantı Tamamlandı.
404	Sayfa bulunamadı!

Buraya kadar herhangi bir sıkıntı yok. Şimdi kodlarımızı tamamlayalım. Tamamlanmış kodlarımız aşağıdaki gibi olacaktır.

```
var ajaxObject;
try {
    ajaxObject = new XMLHttpRequest();
} catch (err) {
    try {
        ajaxObject = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (err) {
        try {
            ajaxObject = new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch (err) {
            document.write(err.message);
        }
    }
}

ajaxObject.onreadystatechange = function(){
    if (ajaxObject.readyState == 4 && ajaxObject.status == 200) {
        // yapılacaklar
    }
}

ajaxObject.open("GetVeyaPost","dosyaVeyaAdresYolu", trueYadaFalse);
ajaxObject.send(Post ise parametreler);
```

Yukarıdaki kodlarda bir yere dikkatini çekmek istiyorum. Nesnemizi oluşturduk, tamam. Lakin istek en

sonda yapılıyor. İsteğin üzerinde de belirttiğimiz adresi dinleme işlemi gerçekleştiriliyor. Size de garip gelmedi mi yukarıdan Top - Bottom (Yukarı - Aşağı) doğru satır satır işlenen bir dilde? Bu durumu hemen açıklayalım. AJAX nesnesi oluşturulduktan sonra onreadystatechange özelliğine bakılır. Henüz ortada birşey yoktur. Daha sonraki satıra geçilir. İstek yapılır. İstekten dönen değer onreadystatechange özelliği içinde işlenir. Bu, yazmış olduğumuz anonim fonksiyon aracılığıyla gerçekleştirilir. Eğer isteği önce yapıp daha sonra dinleme yaparsanız kodlarınız çalışmaz. Lakin open () fonksiyonunu onreadystatechange özelliğinin hemen üzerinde yani nesnemizi oluşturduktan sonra da yazabilirsiniz.

Yukarıdaki kodlarımız size karmaşık geldiyse aşağıdaki kodları da kullanabilirsiniz.

```
var ajaxObject;
try {
    ajaxObject = new XMLHttpRequest();
} catch (err) {
    try {
        ajaxObject = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (err) {
        try {
            ajaxObject = new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch (err) {
            document.write(err.message);
        }
    }
}
//nesne olusturulduktan sonra open() fonksiyonu kullanılabilir.
ajaxObject.open("GetVeyaPost","dosyaVeyaAdresYolu", trueYadaFalse);

ajaxObject.onreadystatechange = function(){
    if (ajaxObject.readyState == 4 && ajaxObject.status == 200) {
        // yapılacaklar
    }
}

ajaxObject.send(Post ise parametreler);
```

Yukarıdaki örneğimizde open () fonksiyonunu nesne oluşturulduktan hemen sonra yazdık. Daha önce de dediğim gibi bir önceki kodlarımız size karmaşık gelmişse yukarıdaki şekilde yazılmış bir kod daha mantıklı gelecektir. Burada dikkat etmenizi istediğim konu send () metodunu en sonda kullanmanız. Eğer open () fonksiyonunu nesne oluşturduktan sonra yazıp hemen ardından send () metodunu kullanırsanız hata alırsınız. Çünkü dinleme işlemi hiç gerçekleşmez. Buna dikkat edelim.

Evet, kodlarımızı tamamladık. Geriye kalan tek şey; alınan cevapların kullanılması. Burada da devreye Text ve XML şeklinde alınan verinin türüne göre kullanmamız kaldı. Şimdi de bu iki kavrama sırasıyla bakalım.

4.1 responseText

Server'a istek yaptık diyelim. Dönen değer bir text yani metinsel bir ifadeyse responseText özelliğini kullanmamız gerekir. responseText özelliği aşağıdaki gibi kullanılır.

```
ajaxObject.responseText
```

Peki bu bahsetmiş olduğumuz özelliğimizi nasıl sayfamıza yansıtacağız? Burada da devreye DOM

(Document Object Model) girecek. Yani belirlediğiniz HTML elementlerini manipüle edeceğiz. DOM bir API'dir. Eğer hiç bilginiz yoksa; kesinlikle bakmanızı öneririm. DOM için, daha önce yazmış olduğum "Simple JavaScript" kitabına bakabilirsiniz, detaylıca anlatmaya çalışmıştım. Herneyse konumuza geri dönecek olursak; responseText özelliğini, herhangi bir HTML elementini DOM ederek sayfamıza yansıtabiliriz. Bazı durumlarda sayfamıza yansıtmaya da biliriz. İstek yapılmıştır ve gerekenler gerektiği yere ulaşmıştır. Şayet kodlarımızı doğru yazmış isek. Bildiğiniz gibi JavaScript ile yazılan kodun doğruluğu adım adım takip edilmediği sürece anlaşılabilir. Herhangi bir hata mesajını da kendiliğinden vermez.

Aşağıdaki kodlarımızda örnek bir responseText özelliğini kullandık.

```
document.getElementById("elementId").innerHTML =  
ajaxObject.responseText;
```

veya

```
document.getElementById("elementId").value = ajaxObject.responseText;
```

elementId alanı, herhangi bir elementin id değeridir. Yani o elementi DOM ettik ve gelen değeri atadık. Bir anlamda sayfamıza yansıttık. Durum bu kadar basit.

4.2 responseXML

Gelen cevabımız bir XML şeklindeyse; responseXML özelliğini kullanmalıyız. responseText özelliğinde olduğu gibi yine HTML elementlerini DOM etmeliyiz. Burada DOM işlemi elementId değerine göre değil tagName yani elementin adına göre olmalıdır. responseXML özelliği aşağıdaki gibi kullanılır.

```
ajaxObject.responseXML
```

Aşağıdaki kodlarımızda örnek bir responseXML özelliğini kullandık.

```
document.getElementsByTagName("elementTagName").innerHTML =  
ajaxObject.responseXML;
```

Daha sonra gelen veriler XML formatında olması gerektiği için ona göre işlemlerimizi gerçekleştirmeliyiz. Dilerseniz bu işlemin devamını bir sonraki bölüme bırakalım. Bir sonraki bölümde detaylı olarak örneklerimiz üzerinde duracağız.

BÖLÜM: 5

Uygulamalar

Artık kolları sıvayıp AJAX uygulamalarımızı geliştirebiliriz. Bu bölümde temel olarak bir `.txt`, `.xml` ve `.json` uzantılı dosyadan veri nasıl alınır bunun üzerinde duracağız. Bir sonraki bölümde; PHP ile AJAX kullanımı ondan sonraki bölümde de JSP ile AJAX kullanımını göreceğiz. Ayrıca AJAX ile veri tabanı işlemlerini de anlatmaya çalışacağız.

Ayrıca bu bölümü 3 ana başlığa ayırdım. Bunlar; `responseText`, `responseXML` ve `JSON`. Bu 3 farklı veri tipi demektir. Hepsini detaylı olarak incelemeye çalışacağız. O halde hazırsanız başlayalım.



Uygulamalarımızı daha anlaşılır olması için adım adım gideceğiz. Yani kafada herhangi bir soru işareti kalmaması için en iyi yolun bu yöntem olduğunu düşünüyorum.

5.1 responseText Uygulamaları

Öncelikle `.txt` uzantılı bir dosya oluşturalım. Bu dosyaya ne yazdığımız önemli değil. Ben kısaca; `emrecan-oztas` yazdım. Aşağıda görüldüğü gibi.

```
textFile.txt (~/Desktop/AJAXExample) - gedit
Open Save Undo
textFile.txt x
emrecan-oztas|
Plain Text Tab Width: 8 Ln 1, Col 14 INS
```

Oluşturmuş olduğum dosyanın adına basitçe; `textFile.txt` dedim. Siz herhangi bir isim verebilir veya herhangi bir editörde çalışabilirsiniz.

Daha önce oluşturmuş olduğumuz `xHttp.js` dosyamızı da bu dosyamızın yanına konumlanduralım. İkinci bölümde hatırlarsanız; tarayıcılar için basit bir kod yazmıştık. O dosyamızın içeriği de aşağıdaki gibidir.

```
xHttp.js (~/Desktop/AJAXExample) - gedit
Open Save Undo
xHttp.js x
function controlXHttp(ajaxObject){
  try {
    ajaxObject = new XMLHttpRequest();
    return ajaxObject;
  } catch (err) {
    try {
      ajaxObject = new ActiveXObject("Msxml2.XMLHTTP");
      return ajaxObject;
    } catch (err) {
      try {
        ajaxObject = new ActiveXObject("Microsoft.XMLHTTP");
        return ajaxObject;
      } catch (err) {
        return null;
      }
    }
  }
}
}

JavaScript Tab Width: 8 Ln 17, Col 17 INS
```

Yukarıdaki ekran alıntısında da görüldüğü gibi; fonksiyonumuza controlXHttp adını verdim. Sizde herhangi bir isim verebilirsiniz. Önemli değil.

Şimdi son aşama olarak bir .html uzantılı dosya oluşturalım ve xHttp.js dosyamıza da bu .js uzantılı dosyamızı ekleyelim. Aşağıdaki ekran alıntısında olduğu gibi.

```
*index.html (~/Desktop/AJAXExample) - gedit
Open Save Undo
*index.html x
<!DOCTYPE html>
<html>
  <head>
    <title>AJAX - responseType Example</title>
    <meta charset="UTF-8">
    <script src="xHttp.js" type="text/javascript"></script>
  </head>
  <body>
  </body>
</html>

HTML Tab Width: 8 Ln 9, Col 1 INS
```

Yukarıdaki ekran alıntısında da .html uzantılı dosyamızı oluşturduk. Basit olması açısından; hem xHttp.js ve textFile.txt dosyalarını bu index.html dosyasının yanına konumlandırdık. Tabiki siz farklı konumlar verebilirsiniz. Path (Yol veya adres)'leri de ona göre değiştirmek kaydıyla.

Kısaca hiyerarşimiz aşağıdaki gibi olacaktır.


```
AJAXExample /
|- xHttp.js
|- index.html
|- textFile.txt
```

Şimdi kodlarımızı yazmaya başlayalım. İlk olarak index.html dosyamıza bir label ve bir de button ekleyelim.

```
<!DOCTYPE html>
<html>
<head>
<title>AJAX - responseText Example</title>
<meta charset="UTF-8">
  <script src="xHttp.js" type="text/javascript"></script>
<body>
<label id="lbl">Merhaba!</label>
<input type="button" value="Click">
</body>
</html>
```

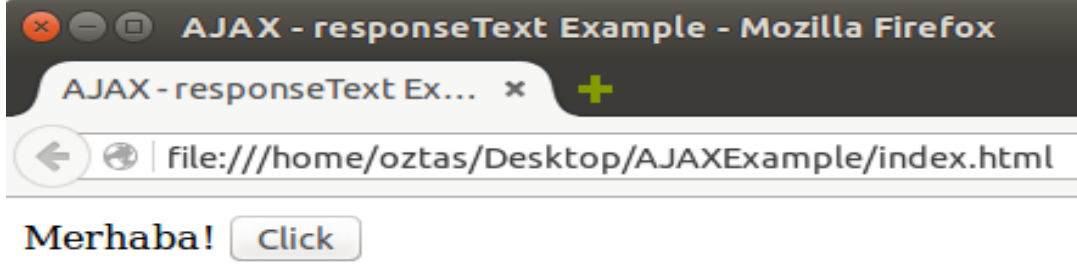
HTML sayfamız hazır olduğuna göre AJAX kodlarımızı yazmaya başlayabiliriz. Buton'a tıklandığı zaman textFile.txt'te içerisindeki yazımızı lbl id'li label'i değiştirelim yani DOM yapalım. Öncelikle yapmamız gereken bir fonksiyon oluşturmak ve önce; XMLHttpRequest'i oluşturmak. Zaten daha önceden XMLHttpRequest'imizi oluşturmuştuk. Sadece çağırıp kullanacağız. Daha sonra ise geriye kalan; request (istek)'imizi göndermek. Yazacağımız kodların tamamı aşağıdaki gibi olacaktır.

```
<!DOCTYPE html>
<html>
<head>
<title>AJAX - responseText Example</title>
<meta charset="UTF-8">
  <script src="xHttp.js" type="text/javascript"></script>
<body>
<label id="lbl">Merhaba!</label>
<input type="button" value="Click" onclick="sendRequest()">
<script>
  // request fonksiyonumuzu yazalım.
  function sendRequest() {
    // XMLHttpRequest nesnemizi oluşturalım.
    var ajaxObject = controlXHttp(ajaxObject);
    // response yapalım
    ajaxObject.onreadystatechange = function() {
      if (ajaxObject.readyState == 4 && ajaxObject.status == 200) {
        // gelen text değerini id'si lbl olan elemente atayalım.
        document.getElementById("lbl").innerHTML =
          ajaxObject.responseText;
      }
    };
    // request yapalım.
    ajaxObject.open("GET", "textFile.txt", true);
    ajaxObject.send();
  }
</script>
</body>
</html>
```

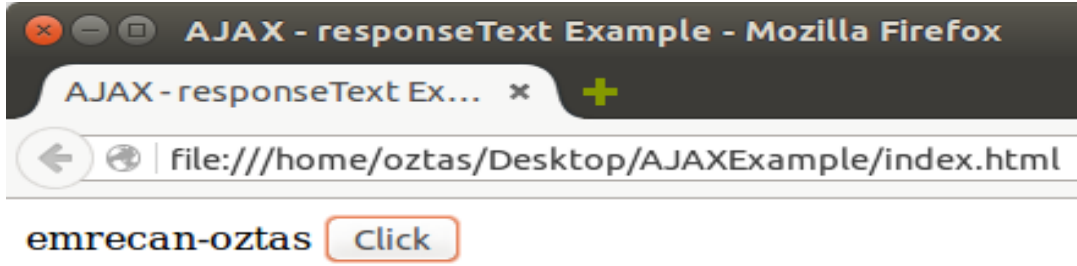
Peki yukarıdaki kodlarımızda neler yaptık şimdi sırasıyla bakalım. Öncelikle daha önce oluşturmuş

olduğumuz xHttp.js dosyasındaki fonksiyonumuza bir parametre gönderdik ve bu parametre bize AJAX nesnesi daha doğrusu tarayıcı nesnesi olarak geri döndü. Daha sonra bu nesneyi kullanarak bir istek yaptık. Peki isteği kime yaptık? textFile.txt'e istek yaptık. Burada dikkat ederseniz; herhangi bir parametre göndermedik ve GET metodunu kullandık. Sadece dosyadaki değeri okuduk. Gelen cevabı da label'e atadık yani DOM yaptık.

Sayfamızın ilk hali aşağıdaki gibidir.



Butona tıkladığımız zaman textFile.txt dosyasındaki değer okunup yazı değişecektir. Peki butona tıklayalım.



Görüldüğü gibi değeri aldık ve label'i DOM ettik. Herhangi bir ekstra sayfa kullanmadık veya sayfayı yenilemedik. İşte bu AJAX sayesinde oldu. Burada değinmek istediğim konu; dosyamızın içerisinde ne varsa hepsini getirdi. Eğer dosyamız içerisinde satırlarca metin olsaydı aynen onu da alıp getirecekti. Peki bu yöntem ile herhangi bir input alanını doldurabilir miyiz? Elbette dolurabiliriz.

Bir input alanını da .value değerini kullanarak doldurabiliriz. Yani;

```
document.getElementById("inpt").value = ajaxObject.responseText;
```

yukarıdaki şekilde bir input alanına da değerini atayabiliriz.

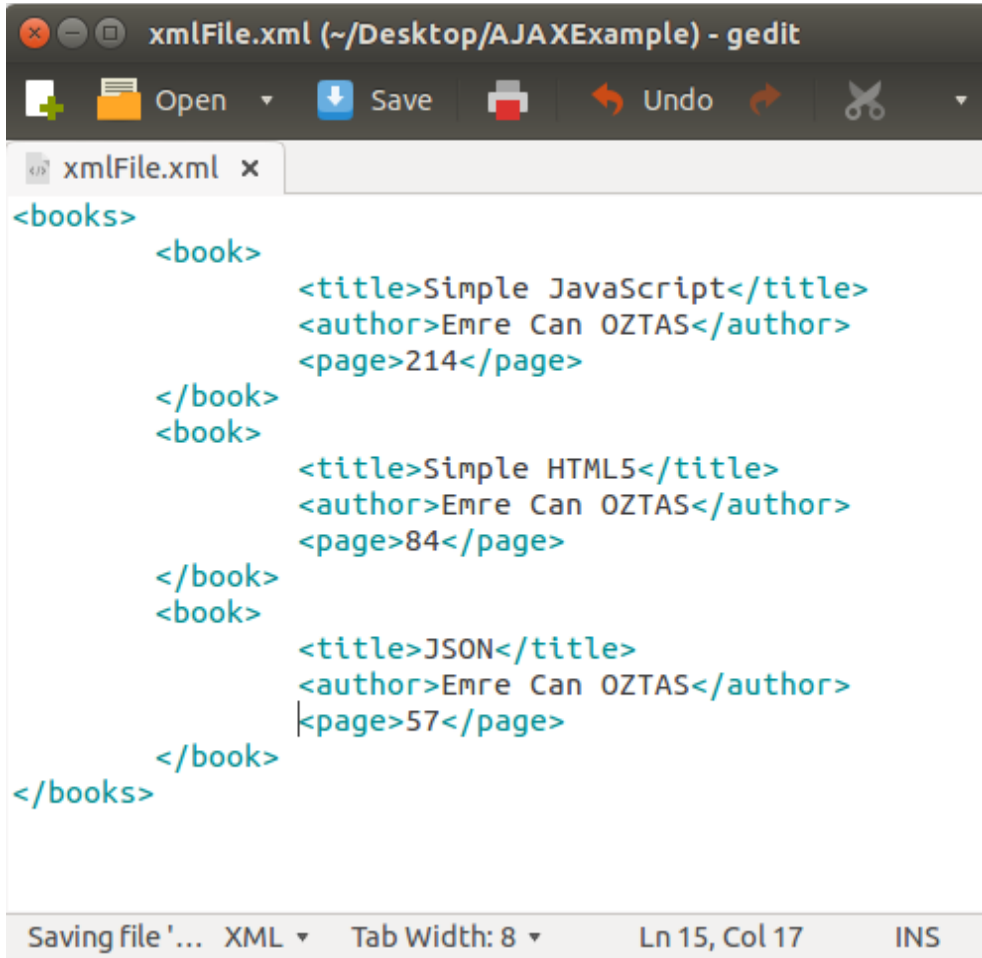
Tabiki bir text dosyasıyla yapacağımız işler çok sınırlı. O yüzden üzerinde fazla durmaya gerek yok. Bir sonraki başlığa geçelim.

5.2 responseXML Uygulamaları

Bu başlık altında da bir XML dosyasından verilerin nasıl okunacağı üzerinde duracağız. Öncelikle .xml uzantılı bir dosya oluşturalım. XML'in yapısından haberdar olduğunuzu varsayarak konularımızı anlatmaya çalışacağız. Herneyse, açmış olduğumuz .xml uzantılı dosyanın içeriği herhangi değerler bütünü olabilir. Biz örnek olarak bir XML yapısı kuracağız. Ayrıca bu .xml uzantılı dosyaya nasıl bir isim vereceğiniz size kalmış. Ben basitçe xmlFile.xml diyeceğim.

.xml uzantılı dosyamızı açtık. Örnek olması açısından bir veri kümesi oluşturalım. Daha öncede dediğim gibi herhangi bir XML dosyasıyla çalışabilirsiniz. Ben örnek olarak bir books (kitaplar) yapısı oluşturacağım.

Oluşturmuş olduğum xmlFile.xml dosyasının içeriği aşağıdaki gibidir.



```
<books>
  <book>
    <title>Simple JavaScript</title>
    <author>Emre Can OZTAS</author>
    <page>214</page>
  </book>
  <book>
    <title>Simple HTML5</title>
    <author>Emre Can OZTAS</author>
    <page>84</page>
  </book>
  <book>
    <title>JSON</title>
    <author>Emre Can OZTAS</author>
    <page>57</page>
  </book>
</books>
```

Bu dosyamızı da diğer dosyaların yanına konumlandıralım. Yani son yapımız aşağıdaki gibi olacaktır.

```
AJAXExample /
|- xHttp.js
|- index.html
|- textFile.txt
|- xmlFile.xml
```

Şimdi kodlarımızı yazmaya başlayabiliriz. Daha önce oluşturmuş olduğumuz index.html dosyası üzerinden devam edelim. Web sayfamızın içeriğini temizleyelim. Temel HTML iskeletimizi kuralım ve daha önce oluşturmuş olduğumuz xHttp.js dosyamızı ekleyelim. Daha sonra da AJAX kodlarımızı yazalım.

Yazacağımız kodlarımız aşağıdaki gibi olacaktır.

```
<!DOCTYPE html>
<html>
<title>AJAX - responseXML Example</title>
<meta charset="utf-8">
  <script type="text/javascript" src="xHttp.js"></script>
<body>
<input type="button" value="Show Books Details"
onclick="sendRequest()">
<br><br>
<table id="tbl"></table>
```

```

<script>
function sendRequest() {
//nesnemizi olusturalim
var ajaxObject = controlXHttp(ajaxObject);
// adresi dinleyelim
ajaxObject.onreadystatechange = function() {
if (ajaxObject.readyState == 4 && ajaxObject.status == 200) {
// gelen cevabi alalim
var xmlReq = ajaxObject.responseXML;
/*
bir tablo olusturalim.
bu tabloyu olusturmamizin amaci: gelen verileri,
duzenli bir sekilde gosterme icin.
*/
var table="<tr><th>Title</th><th>Author</th><th>Page</th></tr>";
/*
gelen veriler xml yapısında olacağı için ayrıca xml'in
yapısı tag (etiket) şeklinde olduğu için;
getElementsByTagName özelliğini kullandık.
*/
var dataRow = xmlReq.getElementsByTagName("book");
/*
xmlFile.xml içerisindeki tüm verileri alacağımız için bir
dongu olusturalim. bu dongumuz xml yapısının büyüklüğüne göre
olmalıdır.
*/
var i;
for (i = 0; i <dataRow.length; i++) {
table += "<tr><td>" +
// sirasiyla xml yapısındaki etiketlerin tuttukları verileri
alalım.
dataRow[i].getElementsByTagName("title")
[0].childNodes[0].nodeValue +
"</td><td>" +
dataRow[i].getElementsByTagName("author")
[0].childNodes[0].nodeValue +
"</td><td>"+
dataRow[i].getElementsByTagName("page")
[0].childNodes[0].nodeValue +
"</td></tr>";
}
// xml yapısından gelen verileri DOM edelim.
document.getElementById("tbl").innerHTML = table;
}
};
// istegimizi gonderelim.
ajaxObject.open("GET", "xmlFile.xml", true);
ajaxObject.send();
}
</script>

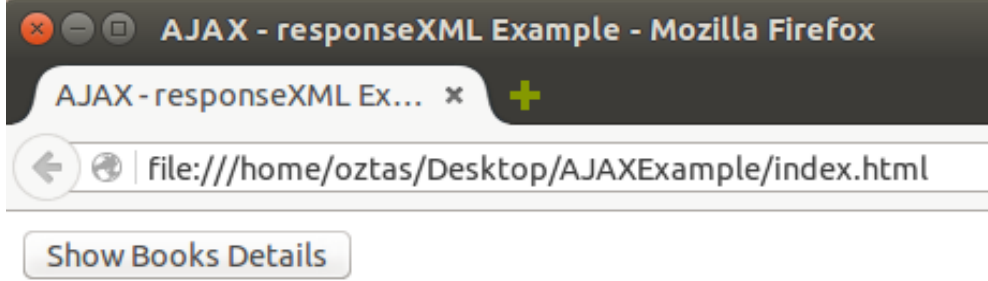
</body>
</html>

```

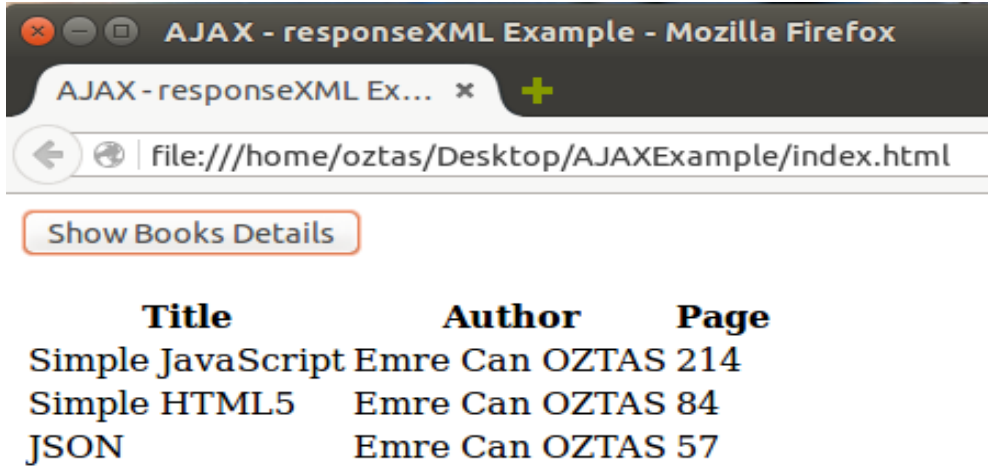
Kodlarımıza bolca açıklama satırları yazdık ama yine de biraz değinmekte yarar olduğuna inanıyorum. Yukarıdaki kodlarımızda neler yaptık? Aslında işin mantığı her zaman aynı. AJAX yapısını anladıktan sonra geriye sadece teferruatlar (DOM v.s) kalıyor. İlk olarak; daha önce oluşturmuş olduğumuz xHttp.js dosyasındaki controlXHttp() fonksiyonunu kullanarak bir tarayıcı nesnesi oluşturduk. Daha

sonra adresi dinlemek için `onreadystatechange` özelliğini kullandık. Gelen değerler XML yapısında olacak, bunu biliyoruz. Yine de kısaca XML yapısından bahsedeyim. XML yapısı `tag (etiket)`'lar şeklinde bulunur. Yani etiketler arasına veriler yazılır. Bu yapıyı bir ağaç diyagramı gibi düşünebilirsiniz. Tepede bir `root` (kök) olur. Daha sonra bu köke bağlı yapılar bulunur. İşte bu yapıları da alıp işledik. Kodlarımızın düzenli olması için bir `table` yapısı oluşturduk. Gelen verileri de bu `table` yapısına DOM ettik. En sonda da isteğimizi gönderdik.

Sayfamızın ilk hali aşağıdaki gibi olacaktır.



Buton tıkladıktan sonraki hali de aşağıdaki gibi olacaktır.



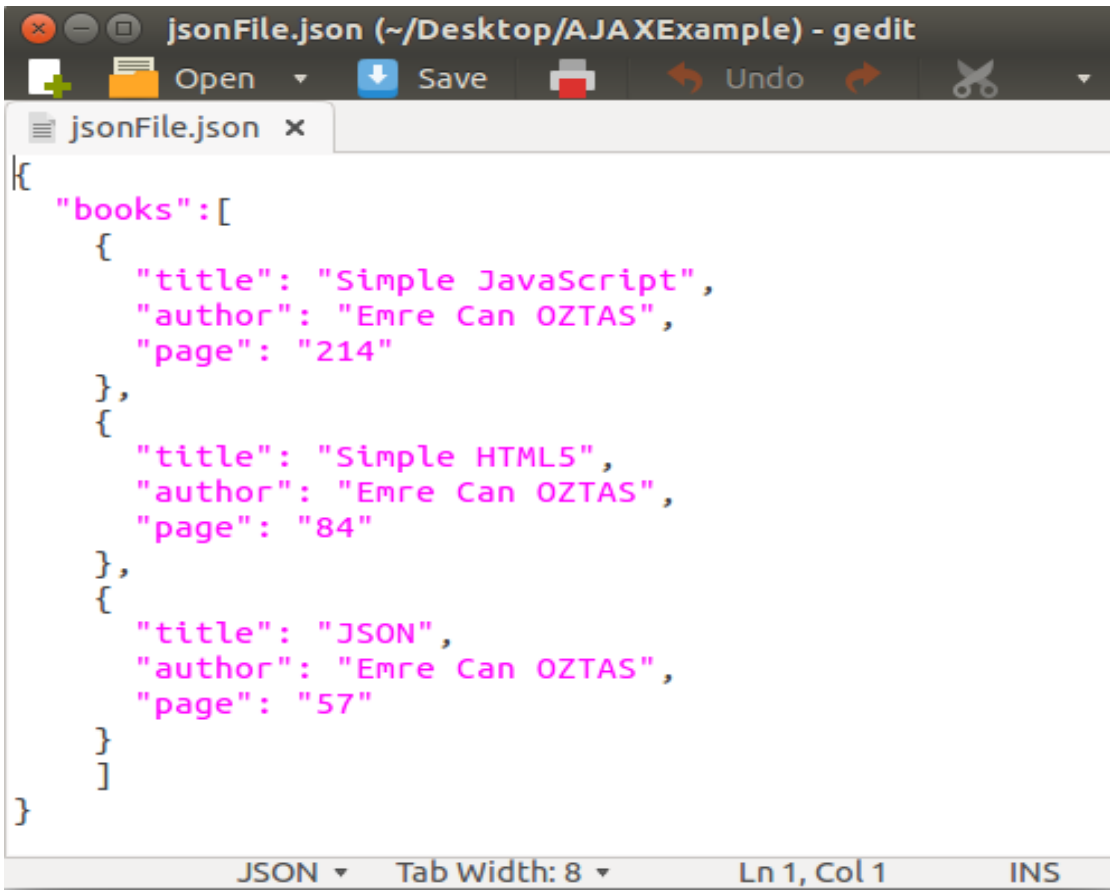
Görüldüğü gibi XML dosyasından verileri aldık ve basitçe sayfamızı `update` (güncelleme) yaptık. Yukarıdaki yapımızı CSS kullanarak daha düzenli bir hale dönüştürebilirsiniz.

5.3 JSON Uygulamaları

JSON (JavaScript Object Notation)'ın mantığı da XML'in mantığıyla aynıdır. İkisi de veri taşıma formatıdır. JSON hakkında gerekli bilgiyi almak isterseniz daha önce yazmış olduğum JSON kitabına bakabilirsiniz.

JSON ve XML için veri taşıma formatıdır dedik. Bunlara bir üçüncü olarakta YAML (YAML Ain't Markup Language) katıldı. YAML diğerlerine göre biraz daha okuması ve yazması kolay bir veri taşıma formatı. Şimdilik en çok XML kullanılıyor. JSON kullanımını da benim gördüğüm kadarıyla oldukça artmış durumda. Bakalım gelecekte YAML'ı iyi bir yerlerde görebilecek miyiz? Bekleyip göreceğiz.

Basit bir örnek yapalım. Öncelikle bir `.json` uzantılı bir dosya açalım. Eğer elinizde varsa kullanabilirsiniz. Herhangi bir sorun olmaz. Ben bu örneğimizde; `responseXML` başlığındaki örneğimizi kullanacağım. XML yapısındaki verilerimizi JSON yapısına çevirelim. Yapımız aşağıdaki gibi olacaktır.



```
{
  "books": [
    {
      "title": "Simple JavaScript",
      "author": "Emre Can OZTAS",
      "page": "214"
    },
    {
      "title": "Simple HTML5",
      "author": "Emre Can OZTAS",
      "page": "84"
    },
    {
      "title": "JSON",
      "author": "Emre Can OZTAS",
      "page": "57"
    }
  ]
}
```

Oluşturmuş olduğumuz bu dosyayı da diğer dosyalarımızın yanına konumlandıralım. Dosyalar kütlememizin son hali aşağıdaki gibi olacaktır.

```
AJAXExample /
|- xHttp.js
|- index.html
|- textFile.txt
|- xmlFile.xml
|- jsonFile.json
```

Şimdi gelelim bu aradaki işin mantığına. .txt uzantılı bir dosyadan verileri alırken; responseText özelliğini kullanıyorduk ve .xml uzantılı bir dosyadan da veri alırken; responseXML özelliğini kullanıyorduk. Peki .json uzantılı bir dosyadan veri alırken ne kullanmamız gerekli? Yani hangi anahtar kelimeyi kullanmalıyız? Önceklikle biraz düşünelim. JSON, JavaScript'ten türetilmiş bir yapı. XML de XHTML'den türetilmiş bir yapı. JavaScript'te JSON oluşturma ve JSON'dan metine dönüştürme fonksiyonları mevcut. Yani JSON'dan alınan verileri bir metin olarak kullanabiliriz. Zaten JavaScript'te bir script dili. Sözümüzü tamamlayacak olursak; AJAX, .json uzantılı bir dosyayı metin olarak tanır. Yani kullanmamız gereken anahtar kelime: responseText olacaktır.

Daha önceki örneklerimizde çalıştığımız index.html dosyamız üzerinden gidelim. Yine içeriği temizleyelim ve temel HTML iskeletini kurup xHttp.js dosyasını sayfamıza ekleyelim. Yine bir buton oluşturalım. Kullanıcı bu butona tıkladığı zaman jsonFile.json'deki verileri alalım.

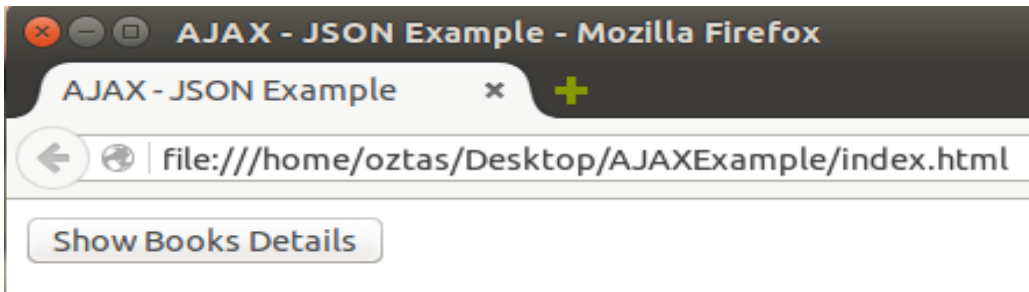
Yazacağımız kodlarımız aşağıdaki gibi olacaktır.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - JSON Example</title>
```

```
<script type="text/javascript" src="xHttp.js"></script>
<style type="text/css">
div{
width: 400px;      }
</style>
</head>
<body>
<input type="button" value="Show Books Details"
onclick="sendRequest()">
<br><br>
<div id="dv"></div>
<script type="text/javascript">
function sendRequest() {
var ajaxObject = controlXHttp(ajaxObject);
ajaxObject.onreadystatechange = function() {
if(ajaxObject.readyState == 4 && ajaxObject.status == 200) {
document.getElementById("dv").innerHTML =
ajaxObject.responseText;
}};
ajaxObject.open("GET", "jsonFile.json", true);
ajaxObject.send();
}
</script>
</body>
</html>
```

Kodlarımızı yazdık. Dikkat ederseniz; bir .txt uzantılı dosyadan veri alırken kullandığımız yapı. Lakin burada da küçük bir Trick (Hile) var aslında ve şuan biz o hile düştük. Bu hilenin ne olduğunu şimdi göreceğiz. Lakin bu hileden de kurtulacağız.

Sayfamızın ilk hali aşağıdaki gibi olacaktır.



Butonumuza tıklayalım.



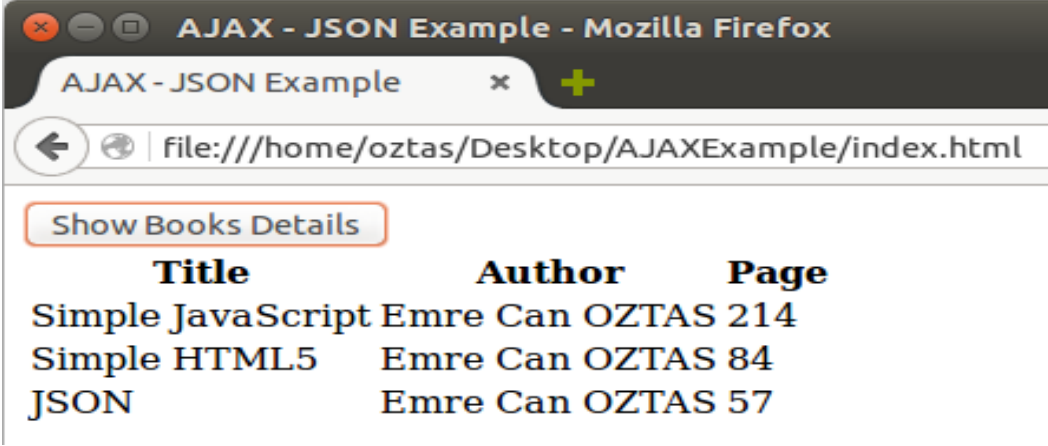
Görüldüğü gibi jsonFile.json içerisinde tüm verileri getirdi. Tamam sorun yok ama JSON'ın yapısını olduğu gibi yazdı. Aslında burada JSON dosyasından gelen veriyi direk olarak text'e çevirip daha sonra bu XML yapısında kullandığımız gibi bir table oluşturup içerisine yazmamız gerekiyordu lakin biz bunu yapmadık. Şimdi bunu yapalım. Kodlarımızın son hali aşağıdaki gibi olacaktır.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - JSON Example</title>
<script type="text/javascript" src="xHttp.js"></script>
</head>
<body>
<input type="button" value="Show Books Details"
onclick="sendRequest()">
<table id="tbl"></table>
<script type="text/javascript">
function sendRequest() {
    // nesnemizi olusturalim
    var ajaxObject = controlXHttp(ajaxObject);
    // verilerimizin daha duzenli gosterimi icin bir table
    olusturalim.
    var table="<tr><th>Title</th><th>Author</th><th>Page</th>
<td></td>";
    // istedigimizi dinleyelim
    ajaxObject.onreadystatechange = function() {
    if(ajaxObject.readyState == 4 && ajaxObject.status == 200) {
    /* Buraya dikkat!
    JSON olarak gelen verimizi text'e cevirelim.
    Daha sonra da text'e cevirdigimiz verimizi alalim
    Bir table olustururalim. Bu table'i gelen verilerle
    dolduralim.
    */
    var jsonObject = JSON.parse(ajaxObject.responseText);
    for(i = 0; i < jsonObject.books.length; i++){
    table += "<tr><td>" + jsonObject.books[i].title +
    "</td><td>" + jsonObject.books[i].author +
    "</td><td>" + jsonObject.books[i].page + "</td></tr>";
    document.getElementById("tbl").innerHTML = table;
    }
    }
    }
    // istedigimizi gonderelim.
```



```
    ajaxObject.open("GET", "jsonFile.json", true);
    ajaxObject.send();
  }
</script>
</body>
</html>
```

Kodlarımızı yazdık. Son olarak butonumuza tıklayalım ve ekran çıktısını alalım.



Görüldüğü gibi verilerimizi düzenli bir şekilde ekrana yazdırabildik. Daha öncede bahsettiğimiz gibi ve sizlerinde gördüğü gibi JSON yapısı responseType yapısına benzerdir. Hem text halindeki hemde JSON yapısındaki verilerin kullanımında da responseType kullanılır.

BÖLÜM 6:

PHP ile AJAX

Bu bölümde PHP ile AJAX kullanımını anlatmaya çalışacağız. Temel olarak PHP dosyalarıyla nasıl çalışır, veri tabanı işlemleri nasıl yapılır gibi işlemler üzerinde duracağız. Dikkat ederseniz önceki bölümde çeşitli veri tipleri ile çalışmıştık fakat bu çalışmalarımız tamamen Static (Statik) bir yapıdaydı. Yani sadece olan veriyi alıp yazdık. Bu verilerimizde bir dosyanın içerisinde bulunuyordu. GET metodunu kullandık. POST metodunu kullanamazdık zaten.

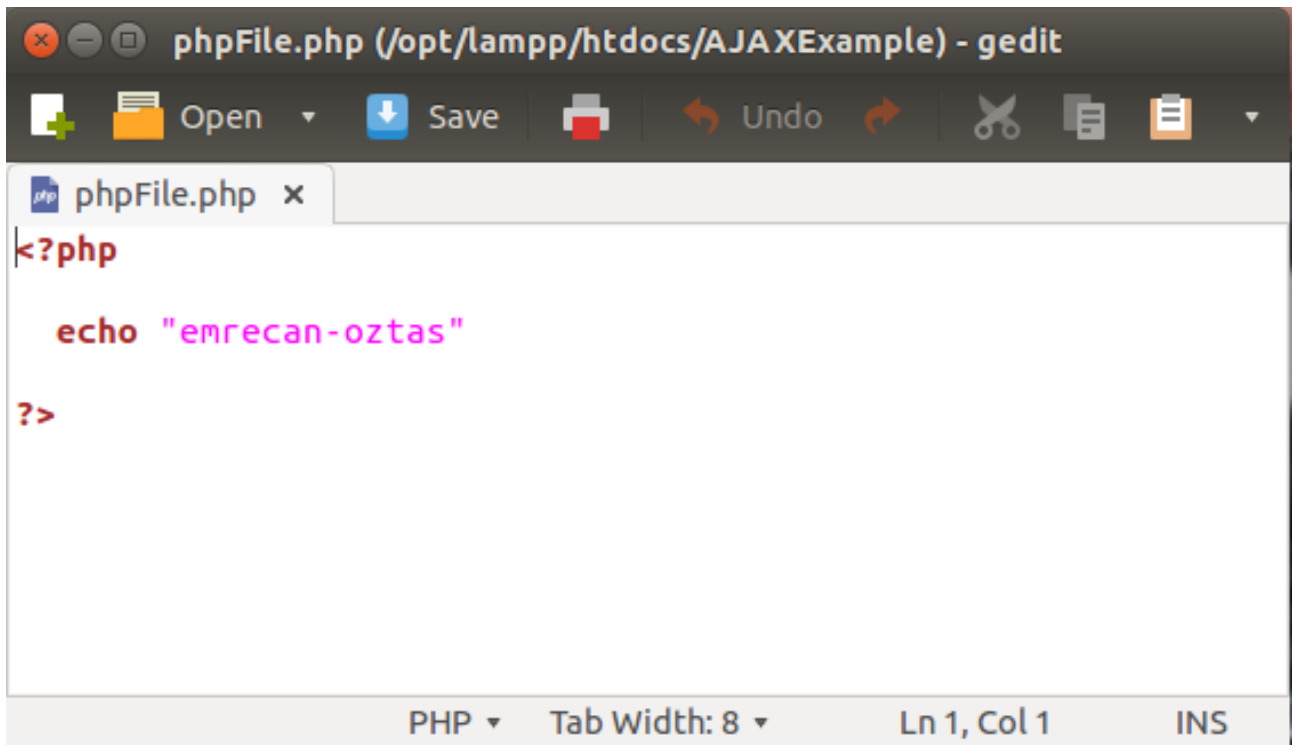
Bölüm boyunca çeşitli örnekler üzerinde durmaya ve basitten zora doğru hareket edeceğiz. O yüzden her örneğimizi farklı bir başlık altında incelemeye çalışacağız.

Herneyse, PHP ile çalışacağımız için bir Local Server'a ihtiyacımız olacak. PHP bilginiz olduğunu ve bilgisayarınızda bir PHP sunucusu olduğunu varsayıyorum. O zaman lokale bir çalışma dosyası açalım. XAMPP kullanıyorsanız; htdocs, WAMPP veya EasyPHP kullanıyorsanız; www dosyası lokal olacaktır. Ben yine çalışma dosyamızın adını: AJAXExample koyuyorum. Bu dosyamızın içerisinde bir index.html uzantılı dosya oluşturalım. Daha önce oluşturmuş olduğumuz xhttp.js dosyamızı da bu çalışma dosyamıza konumlanduralım. Son olarak bir de .php uzantılı dosya oluşturalım. Bu dosyanın ismi size kalmış, herhangi bir isim verebilirsiniz. Ben basitçe phpFile.php koyuyorum. Son olarak oluşturmuş olduğumuz hiyerarşimiz aşağıdaki gibi olacaktır.

```
localhost: AJAXExample /
|- index.html
|- xhttp.js
|- phpFile.php
```

ÖRNEK 1:

Çok basit bir örnek yapalım. phpFile.php dosyamızı açalım ve aşağıdaki gibi bir yapı oluşturalım.



```
<?php
    echo "emrecan-oztas"
?>
```

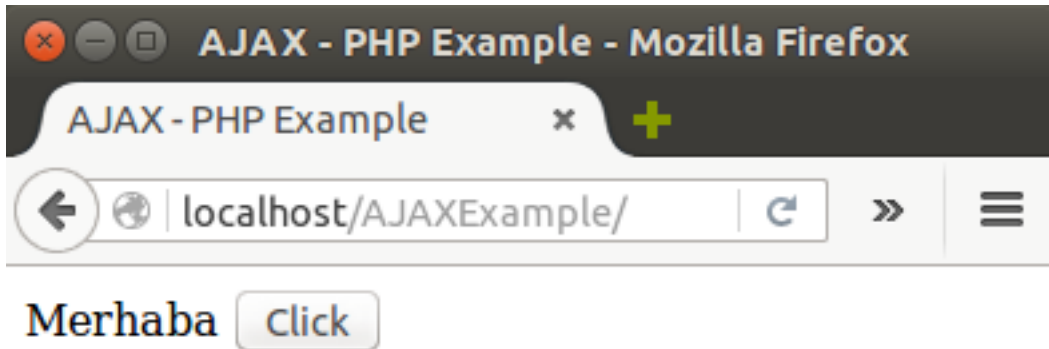
PHP dosyamızın içerisinde; echo ile basit bir yazdırma işlemi yaptık. Ben kendi adımları yazdım sizde herhangi bir ifade yazabilirsiniz. Basit bir örnek olması ve PHP dosyalarıyla çalışmak için bir örnek yapacağız. Şimdi artık index.html dosyamızı düzenleyebiliriz. Yine temel AJAX yapımızı kuralım. Burada dikkat etmemiz gereken konu; dosyanın adı. Dosyamızın adı phpFile.php olacak.

Yazacağımız kodlarımız aşağıdaki gibi olacaktır.

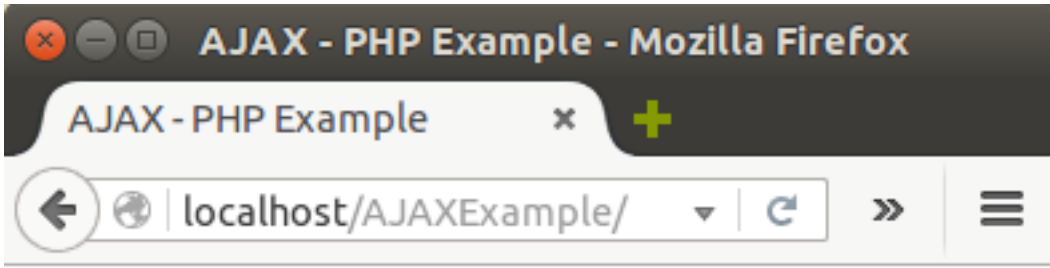
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - PHP Example</title>
<script type="text/javascript" src="xHttp.js"></script>
</head>
<body>
<label id="lbl">Merhaba</label>
<input type="button" value="Click" onclick="sendRequest()">
<script type="text/javascript">
function sendRequest() {
var ajaxObject = XMLHttpRequest(ajaxObject);
ajaxObject.onreadystatechange = function() {
if(ajaxObject.readyState == 4 && ajaxObject.status == 200) {
document.getElementById("lbl").innerHTML =
ajaxObject.responseText;
}};
ajaxObject.open("GET", "phpFile.php", true);
ajaxObject.send();
}
</script>
</body>
</html>
```

Kodlarımız, herhangi bir text dosyasından veri alırkenki kullandığımız kodlar. Herhangi bir değişiklik yapmadık. Sadece dosyanın adını değiştirdik o kadar. İşte AJAX ku kadar basit. Yeterki nasıl kullanacağımızı bilin.

Sayfamızı tarayıcıda açalım.



Görüldüğü gibi sayfamız çalışıyor. Hemen butona tıklayalım ve değişikliklere bakalım.



emrecan-oztas

Click

Görüldüğü gibi sayfamız değişti.

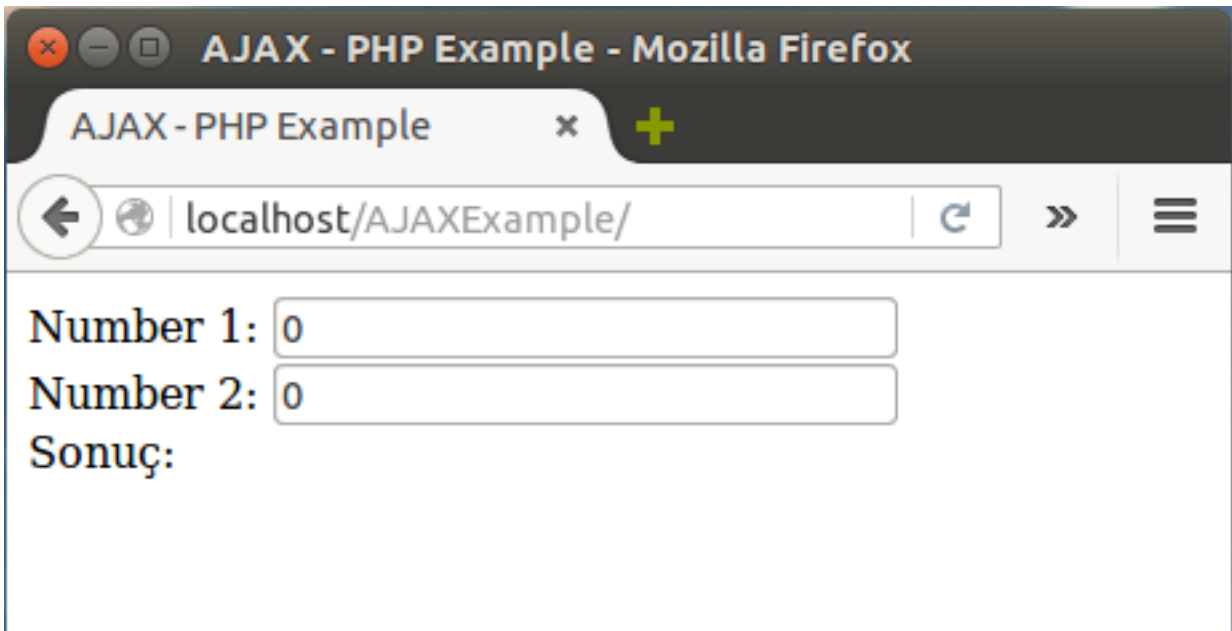
Şimdi işin inceliklerinden bahsedelim. Bu işlem nasıl gerçekleşti? Halbuki PHP dosyasında sadece echo'yu kullandık. AJAX ile istek yapıldığı anda; istek yapılan adres dinlenmeye çalışılır. Yani bir nevi değişiklikler dinlenir. PHP dosyamızda ise istek yapıldığı anda ekrana “emrecan-oztas” yazılacaktır. İşte bu bir değişikliktir aslında. Yani değişiklik olduğu anda `ajaxObject.onreadystatechange` özelliği tetiklenecektir. Örneğin bir değişken kullansaydık ve bu değişkene ilk değerini atasaydık bu bir değişiklik olmayacak, ekrana herhangi bir şey yazılmayacaktı. Lakin bu değişkeni ekrana yazdırdığımızda bu bir değişiklik olacaktır. Buna dikkat edelim. Peki birden fazla echo kullansaydık? O zamanda bütün yazılanlar getirilecekti.

Geçelim bir diğer konuya. Ön sayfamızda `index.html` kullandık. Kullanmak zorunda değiliz. İstersek; `index.php` olarakta kullanabiliriz ya da başka bir isim verebiliriz. Yeterki AJAX yapımızı düzgün bir şekilde kuralım.

ÖRNEK 2:

AJAX ile PHP yapısını anlamak için değişik örnekler üzerinde duralım. Örneğin iki sayının toplamını AJAX ile yapalım. Bir sonraki örneğimizde de 4 işlemi AJAX ile gerçekleştirelim. Yine `index.html` ve `phpFile.php` sayfalarımız üzerinden örneğimizi anlatmaya çalışalım. İki sayının toplamını bulacağımız için bize 2 tane input alan lazım. Sonuçları göstermek içinde bir label oluşturalım. Butona ihtiyacımız olmayacak. Çünkü input alanlarına ne yazarsak hemen işlemimiz gerçekleşmeli.

Basit olarak sayfamızın tasarımı aşağıdaki gibi olsun. Siz isterseniz farklı bir tasarım yapabilirsiniz.



Yukarıdaki sayfamızın HTML kodları da aşağıdaki gibi olacaktır.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - PHP Example</title>
  <script type="text/javascript" src="xHttp.js"></script>
</head>
<body>
<label>Number 1: </label><input type="text" value="0" id="n1"><br>
<label>Number 2: </label><input type="text" value="0" id="n2"><br>
<label id="lbl">Sonuç: </label>
</body>
</html>
```

Sayfamız tamam. Şimdi kodlarımızı yazmaya başlayabiliriz. Yapmak istediğimiz şu; n1 ve n2 id'li input alanlarına sayı girildiği zaman hemen arkaplanda yani phpFile.php içerisinde bu iki sayının toplamını bulup lbl id'li label'e sonucu atamak.

Burada dikkat etmemiz gereken bir kaç konu var. Eğer jquery kullanmış olsa idik input alanlarındaki değerleri basitçe alıp gönderebilirdik. Lakin klasik JavaScript ortamında bu işlem farklı bir yolla hallediliyor. Önce input alanlarındaki değerleri almalıyız daha sonrada bu değerleri AJAX ile phpFile.php'e istek olarak göndermeliyiz. Yani DOM yapacağız. Daha iyi anlamamız açısından kodlarımızı yazalım ve kodlarımız üzerinden anlatalım.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - PHP Example</title>
  <script type="text/javascript" src="xHttp.js"></script>
</head>
<body>
<!--
```

Buraya dikkat! input alanlarındaki değişiklikleri algılamak için; onkeyup event(olay)'ini kullandık. Yani kullanıcı herhangi bir değer girdiği anda sendRequest() fonksiyonumuz çalışacak.

```
-->
<label>Number 1: </label><input type="text" value="0" id="n1"
onkeyup="sendRequest()"><br>
<label>Number 2: </label><input type="text" value="0" id="n2"
onkeyup="sendRequest()"><br>
<label id="lbl">Sonuç: </label>
<script type="text/javascript">
function sendRequest() {
    // input alanlarındaki degerleri alip, int yani sayisal veri
    turune cevirelim.
    // bunu yapmamizdaki amac: javascript'te input alanlarindan
    alinan degerler string'tir.
    var n1 = parseInt(document.getElementById("n1").value);
    var n2 = parseInt(document.getElementById("n2").value);
    var ajaxObject = XMLHttpRequest(ajaxObject);
    ajaxObject.onreadystatechange = function() {
    if(ajaxObject.readyState == 4 && ajaxObject.status == 200) {
    // istekten donen degerler lbl id'li label'e atanacak.
    document.getElementById("lbl").innerHTML = "Sonuç: " +
    ajaxObject.responseText;
    }};
    /*
    Buraya dikkat! php dosyasina verilerimizi gonderecegimiz icin;
    input alanlarindan
    alinan degerleri parametre olarak yazmamiz gerekiyor. n1 ve n2
    gidecek olan degerler.
    */
    ajaxObject.open("GET", "phpFile.php?n1="+n1+"&n2="+n2, true);
    ajaxObject.send();
    }
</script>
</body>
</html>
```

Kodumuza bolca açıklama satırı yazdık. Yine de basitçe değinmek gerekirse; input alanlarında meydana gelen herhangi bir değişikliği anlayabilmek için; input'ların onkeyup event(olay)'ına sendRequest() fonksiyonu atadık. sendRequest() fonksiyonunda input alanlarındaki değerleri her değişiklikte aldık. Daha sonra bu değerleri parametre olarak phpFile.php'a gönderdik. Buradaki parametre olarak gönderime dikkat edelim ve anlamaya çalışalım.

```
"phpFile.php?n1="+n1+"&n2="+n2
```

Bu satırda demek istediğimiz; input alanlarında meydana gelen değişiklikleri al daha sonra n1 ve n2 değerlerinin parametresi olarak gönder.

index.html sayfamızda herhangi bir sıkıntı yok. Peki phpFile.php dosyamızın içeriği nasıl olmalı? Ne yazmamız gerekiyor? Öncelikle istek bir GET metodu şeklinde. Yani gelen değerleri \$_GET[] ile almalıyız. Daha sonrada gerekli işlemleri gerçekleştirmeliyiz. phpFile.php dosyamızın içeriği de aşağıdaki gibi olacaktır.

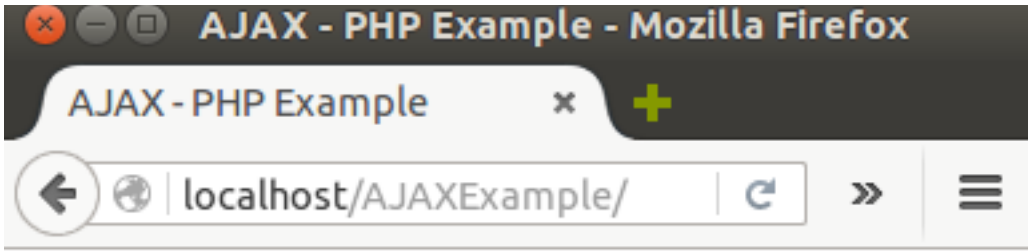
```
<?php
// gelen parametreleri al(n1 ve n2)
$n1 = $_GET["n1"];
$n2 = $_GET["n2"];
```

```
// parametreleri topla ve yazdır.  
echo ($n1 + $n2);  
  
?>
```

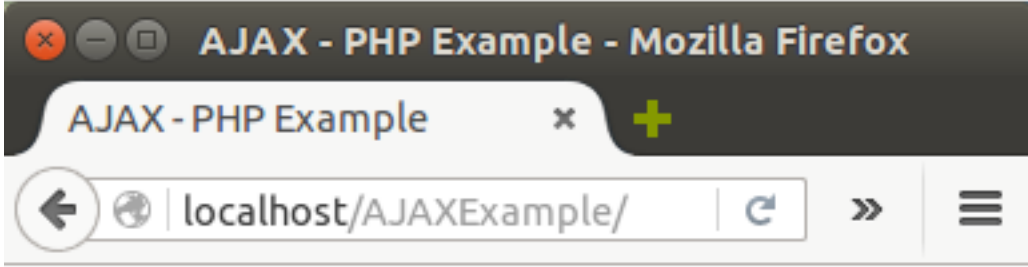
Yazdığımız PHP kodları çok basit. Gelen değerleri istek yapılan metoda (GET) göre aldık. Eğer POST kullanmış olsaydık; \$_POST[] ile almamız gerekiyordu. Bu arada aklınızda bulunması açısından; web ortamında default (varsayılan) olarak GET metodu kullanılır. Çünkü GET metodu çok hızlı çalışır. Bu yüzden tercih sebebidir.

Daha önce sayfamızın ilk halini paylaşmıştık. Şimdi deneyelim bakalım, yazdığımız kodlarımız doğru çalışıyor mu?

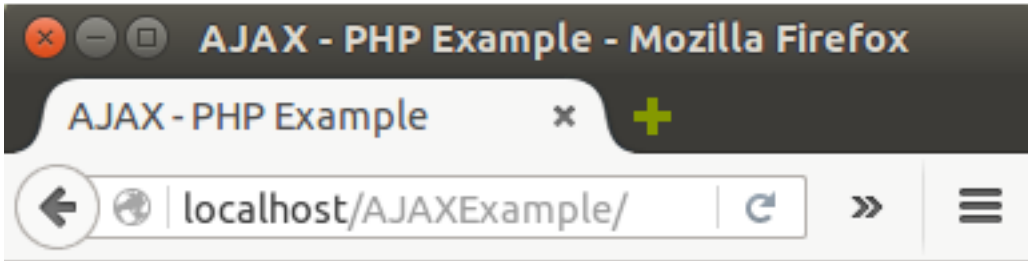
Örnek çıktılarımız aşağıdaki gibi olacaktır.



Number 1:
Number 2:
Sonuç: 48



Number 1:
Number 2:
Sonuç: 11



Number 1:
Number 2:
Sonuç: 20

Görüldüğü gibi sayfamız düzgün bir şekilde çalışıyor. Burada işin mantığını anladığımızı sanıyorum. Tek sorun değerleri almak bunu parametre olarak iletme. Eğer burayı iyi kavramış iseniz herhangi bir sıkıntınız kalmadı demektir. Yani bir anlamda AJAX sizin için bitmiş demektir. Çünkü veri tabanı işlemlerinde de mantık hep aynıdır.

Yine de daha iyi anlamak için bir başka örneğe geçelim.

ÖRNEK 3:

Bir önceki örneğimizde bahsettiğimiz gibi 4 işlemi AJAX ile yapmaya çalışalım. Toplama işlemini gerçekleştirdik. Bu örneğimizde PHP dosyası içerisinde fonksiyonları ve bu fonksiyonları nasıl

tetikleyeceğimize değinelim. Çünkü kullanıcı hangi işlemi gerçekleştirmek istiyorsa seçsin ve ona göre sonuçlar gelsin. Şimdi 4 işlemi de aynı anda yapmamız mantıksızlık olur.

Yine index.html ve phpFile.php dosyalarımız üzerinden gidelim. Input ve label alanlarımız kalsın. Ekstra olarak select alanı açalım ve dört işlemi tanımlayalım. Sayfamızın HTML kodları aşağıdaki gibi olacaktır.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - PHP Example</title>
<script type="text/javascript" src="xHttp.js"></script>
</head>
<body>
<label>Number 1: </label><input type="text" value="0" id="n1"><br>
<label>Number 2: </label><input type="text" value="0" id="n2"><br>
<select id="opt">
<option value="1">Toplama</option>
<option value="2">Çıkarma</option>
<option value="3">Çarpma</option>
<option value="4">Bölme</option>
</select><br>
<label id="lbl">Sonuç: </label>
</body>
</html>
```

Sayfamızın görüntüsü de aşağıdaki gibi olacaktır.



Number 1:

Number 2:

Toplama ▾

Sonuç:

Şimdi burada bir şeye dikkatinizi çekmek istiyorum. Tamam 2 tane input ve 1 tane label alanımız var. Ekstra olarak bir de select alanı açtık. Şimdi burada şöyle bir yol izlememiz daha doğru olacaktır. Kullanıcı sayıları girsin ama AJAX çalışmasın. Lakin kullanıcı select alanını kullandığı zaman AJAX tetiklensin ve bir istek göndersin. Bu durumda ne yapmamız gerekir? Input'ların onkeyup olayı yerine select'in onchange olayını kullanalım. Yani select alanı değiştiği zaman AJAX çalışmaya başlasın. Peki, kodlarımız aşağıdaki gibi olacaktır.

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - PHP Example</title>
    <script type="text/javascript" src="xHttp.js"></script>
</head>
<body>
<label>Number 1: </label><input type="text" value="0" id="n1"><br>
<label>Number 2: </label><input type="text" value="0" id="n2"><br>
<!--
Buraya dikkat! Kullanici herhangi bir islemi sectigi zaman;
AJAX calismaya baslamasi icin onchange event'ini kullandik.
-->
<select id="opt" onchange="sendRequest()">
<option value="1">Toplama</option>
<option value="2">Çıkarma</option>
<option value="3">Çarpma</option>
<option value="4">Bölme</option>
</select><br>
<label id="lbl">Sonuç: </label>
<script type="text/javascript">
    function sendRequest() {
        /*
        Buraya dikkat! input alanlarindaki degerleri aldik.
        Ayrica opt id'li select alanindaki degeri de almamiz gerekiyor.
        Yani kullanici;
        Toplama: 1
        Cikarma: 2
        Carpma: 3
        Bolme: 4
        soldakilerden herhangi birini sectigi zaman sagdaki deger
        uretilecek ve parametre (opt) olarak gonderilecek.
        */
        var n1 = parseInt(document.getElementById("n1").value);
        var n2 = parseInt(document.getElementById("n2").value);
        var opt = parseInt(document.getElementById("opt").value);
        var ajaxObject = controlXHttp(ajaxObject);
        ajaxObject.onreadystatechange = function() {
            if(ajaxObject.readyState == 4 && ajaxObject.status == 200) {
                document.getElementById("lbl").innerHTML = "Sonuç: " +
                ajaxObject.responseText;
            }
        };
        /*
        n1 ve n2 alanlarına ekstra olarak opt degerini de gondermemiz
        lazim.
        Neden? Cunku php dosyasinda kullandigimiz fonksiyonlari
        tetiklememiz gerekiyor.
        Bu farkli bir yolla yapilacagi gibi bana gore en evla yol budur.
        */
        ajaxObject.open("GET", "phpFile.php?
        n1="+n1+"&n2="+n2+"&opt="+opt, true);
        ajaxObject.send();
    }
</script>
</body>
</html>

```

Kodlarımıza bolca açıklama satırı yazdık. Lakin mantık çok basit. Önceki örnekte 2 değer gönderirken bu sefer 3 değer göndereceğiz. Ayrıca artık input alanları kontrol etmek yerine sadece select alanını kontrol edeceğiz.

Peki phpFile.php dosyamızın içeriği nasıl olacak? Hemen PHP kodlarımızı da yazalım. Kodlarımız aşağıdaki gibi olacaktır.

```
<?php

// gelen parametreleri al(n1, n2 ve opt)
$n1 = $_GET["n1"];
$n2 = $_GET["n2"];
// opt kullanıcının işlem tercihi
$opt = $_GET["opt"];

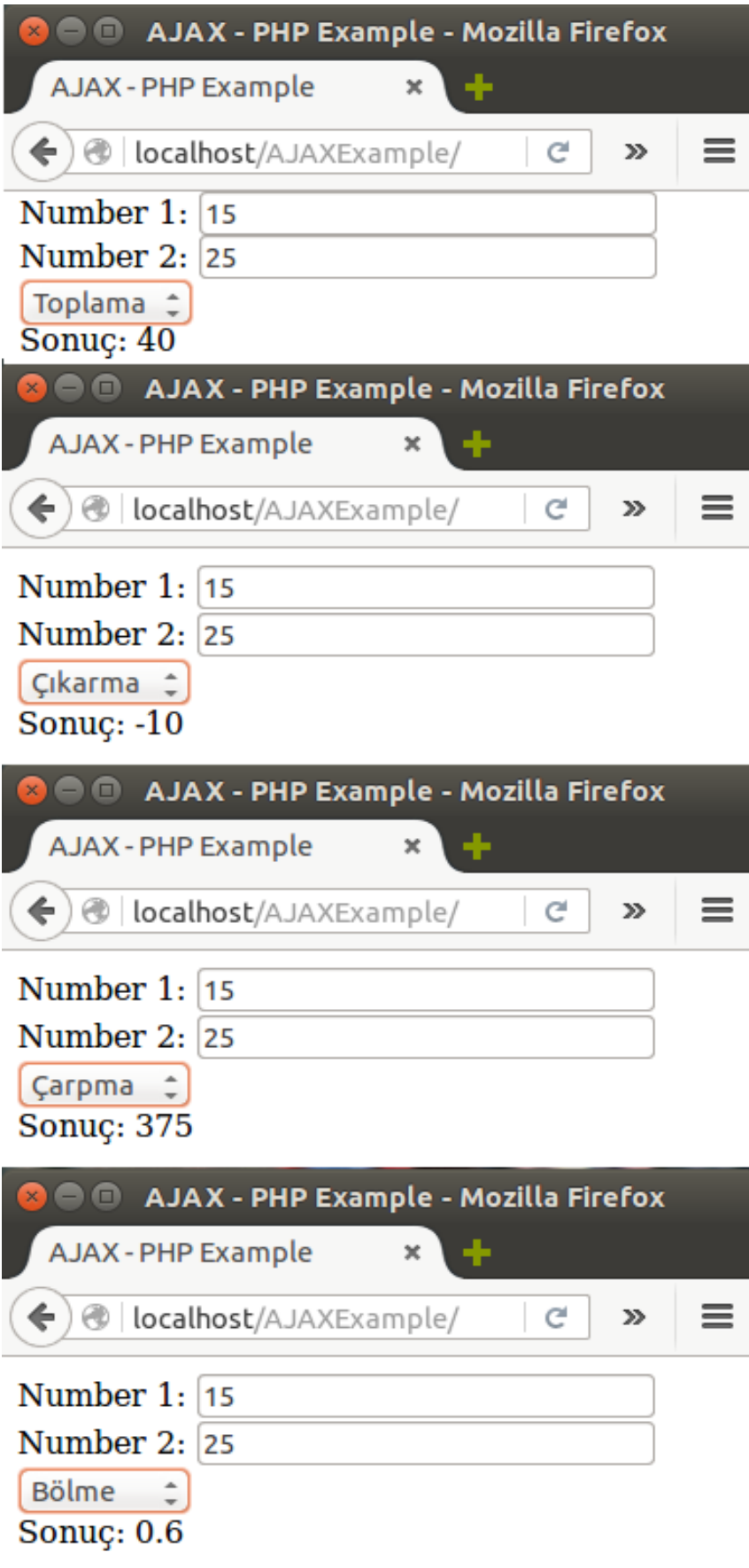
// opt'yi kontrol edelim. ve gerekli
// fonksiyonları çalıştıralım
if($opt == 1) topla($n1,$n2);
else if($opt == 2) cikar($n1,$n2);
else if($opt == 3) carp($n1,$n2);
else bol($n1,$n2);

function topla($n1,$n2){
    echo ($n1 + $n2);
}
function cikar($n1,$n2){
    echo ($n1 - $n2);
}
function carp($n1,$n2){
    echo ($n1 * $n2);
}
function bol($n1,$n2){
    echo ($n1 / $n2);
}

?>
```

PHP kodlarımız oldukça basit. Sadece opt'ye göre gelen değerlere işlem yaptıracak ve ekrana yazdıracak.

Şimdi işlemlerimize geçelim. Bakalım yazdığımız kodlarımız doğru çalışıyor mu?



Görüldüğü gib yazdığımız kodlarımız güzel bir şekilde çalışıyor. Bu mantıkla isterseniz başka uygulamalarda geliştirebilirsiniz. Örneğin YGS – LYS, KPSS veya YDS puan hesaplama uygulaması geliştirip, ücretsiz bir web alanı alıp kullanabilirsiniz.

Dosyalarla çok fazla uğraştık. Biraz da veri tabanı işlemlerine değinelim. Sonraki örneklerimizde de veri tabanı üzerinde duralım.

ÖRNEK - 4:

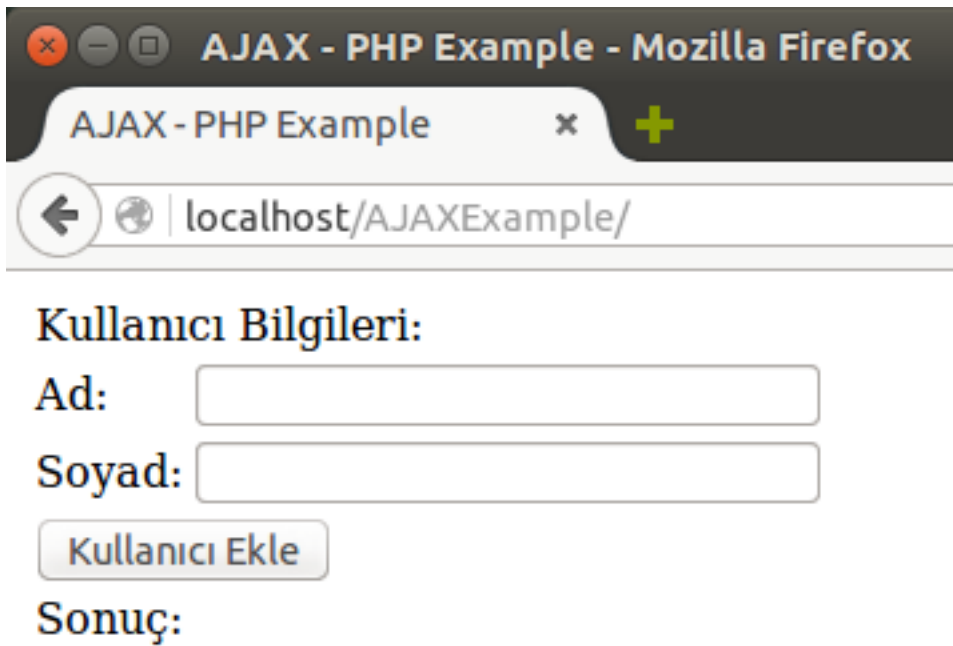
Bu örneğimizde basit bir veri tabanı oluşturalım ve AJAX ile gönderdiğimiz parametreleri veri tabanına kaydedelim. Bunun için öncelikle MySQL (Evet, örneğimizde MySQL kullanacağız)'de bir veri tabanı oluşturalım. Veri tabanımızın adı uye_ler olsun. Veri tabanımızda uye_ler Tablosu adında bir de tablo oluşturalım. Oluşturmuş olduğumuz tablomuzun alanları aşağıdaki gibi olsun.

- Kullanıcı ID'si (kid)
- Kullanıcı Adı (kad)
- Kullanıcı Soyadı (ksoyad)

Parantez içlerindeki ifadeler veri tabanında kullanacağımız isimler olacaktır. Kullanıcı bu alanlardan 2 tanesini girecek. ID değeri bildiğiniz gibi veri tabanına kayıt işleminde otomatik oluşturulacak (AUTO_INCREMENT). Yani biz öyle ayarlayacağız. Şimdi veri tabanı tasarımına geçelim. Veri tabanı tasarımını phpMyAdmin'de gerçekleştirelim. Veri tabanımız aşağıdaki gibi olacaktır.

#	Adı	Türü	Karşılaştırma	Öznitelikler	Boş	Varsayılan	Ekstra	Eylem
<input type="checkbox"/>	1	kid	int(11)		Hayır	Yok	AUTO_INCREMENT	Değiştir Kaldır Birincil Daha fazla
<input type="checkbox"/>	2	kad	varchar(60) utf8_turkish_ci		Hayır	Yok		Değiştir Kaldır Birincil Daha fazla
<input type="checkbox"/>	3	ksoyad	varchar(90) utf8_turkish_ci		Hayır	Yok		Değiştir Kaldır Birincil Daha fazla

Görüldüğü gibi tablomuzu hazırladık. Şimdi tasarım kısmına geçebiliriz. Sayfamızın tasarımını yapalım. Yaptığımız tasarım aşağıdaki gibi olacaktır.



Kullanıcı Bilgileri:

Ad:

Soyad:

Sonuç:

Tasarlamış olduğumuz sayfamızın HTML kodları da aşağıdaki gibi olacaktır.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - PHP Example</title>
<script type="text/javascript" src="xHttp.js"></script>
</head>
```

```

<body>
<table>
<tr><td colspan="2">Kullanıcı Bilgileri:</td></tr>
<tr>
<td>Ad:</td><td><input type="text" id="kad"></td>
</tr>
<tr>
<td>Soyad:</td><td><input type="text" id="ksoyad"></td>
</tr>
<tr>
<td colspan="2"><input type="button" value="Kullanıcı Ekle"></td>
</tr>
<tr>
<td>Sonuç: </td><td><label id="sonuc"></label></td>
</tr>
</table>
</body>
</html>

```

Şimdi AJAX kodlarımızı yazabiliriz. Öncelikle biraz düşünelim. Nasıl yeni kullanıcı kaydedeceğiz? Eğer input'ların onchange olayına AJAX fonksiyonumuzu bağlarsak; kullanıcının her girdiği karakterde veri tabanına kayıt yapılır. Örneğin kullanıcı kad alanına emre can yazmak isterse veri tabanında şöyle bir durum oluşur:

```

e
em
emr
emre
emre
emre c
emre ca
emre can

```

Bu durum ksoyad alanı içinde geçerlidir. Tabiki bu istenmeyen bir durumdur. Peki ne yapmalıyız? Burada yapmamız gereken; oluşturmuş olduğumuz butona tıklayarak kayıt işlemini yapmak yani AJAX işlemini gerçekleştirmek. Yazacağımız kodlarımız AJAX'ı açıklamak için oldukça basit olacak ama eğer isterseniz alanların boşluğunu / doluluğunu, karakter uyumunu v.s gibi konulara göre JavaScript ile kontrolünü yapabilirsiniz. Lakin biz yapmayacağız.

Bu örneğimizde POST metodunu kullanacağız. Peki neden GET değil de POST? Veri tabanına kayıt işlemini gerçekleştireceğimiz için GET metodunu kullanmak üçüncü şahıslara açık kapı bırakmaktır. O yüzden doğrudan veri tabanı işlemlerinde POST metodunu kullanmaya özen göstermeliyiz. POST metodunun kullanımı ile GET metodunun kullanımı arasında küçük bir nüans farkı var. Bu farkı da kodlarımızı yazdıktan sonra açıklayalım.

HTML etiketleri arasında yazacağımız AJAX kodlarımız aşağıdaki gibi olacaktır.

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - PHP Example</title>
<script type="text/javascript" src="xHttp.js"></script>
</head>
<body>
<table>
<tr>
<td colspan="2">Kullanıcı Bilgileri:</td>

```

```

</tr>
<tr>
<td>Ad:</td>
<td><input type="text" id="kad"></td>
</tr>
<tr>
<td>Soyad:</td>
<td><input type="text" id="ksoyad"></td>
</tr>
<tr>
<td colspan="2"><input type="button" value="Kullanıcı Ekle"
onclick="sendRequest()"></td>
</tr>
<tr>
<td>Sonuç:</td>
<td><label id="sonuc"></label></td>
</tr>
</table>
<script type="text/javascript">
function sendRequest() {
var kad = document.getElementById("kad").value;
var ksoyad = document.getElementById("ksoyad").value;
var ajaxObject = XMLHttpRequest();
ajaxObject.onreadystatechange = function() {
if (ajaxObject.readyState == 4 && ajaxObject.status == 200) {
document.getElementById("sonuc").innerHTML =
ajaxObject.responseText;
}};
ajaxObject.open("POST", "phpFile.php", true);
ajaxObject.setRequestHeader("Content-type",
"application/x-www-form-urlencoded");
ajaxObject.send("kad=" + kad + "&ksoyad=" + ksoyad);
}
</script>
</body>
</html>

```

Buraya kadar herhangi bir sıkıntı yok ise PHP tarafında yapacaklarımıza gelelim. Yazacağımız kodlarımız aşağıdaki gibi olacaktır.

```

<?php

$kad = $_POST["kad"];
$ksoyad = $_POST["ksoyad"];
saveToDb($kad, $ksoyad);

function saveToDb($kad, $ksoyad){
    $conn = new mysqli("localhost", "root", "", "uyeler");
    mysqli_set_charset($conn,"utf8");
    $charset = $conn->character_set_name();
    echo "$charset";

    if ($conn->connect_error) {
        die("Bağlantı başarısız!: " . $conn->connect_error);
    }

    $sql = "INSERT INTO uyelerTablosu (kad, ksoyad)

```

```
VALUES ('$kad', '$ksoyad');"

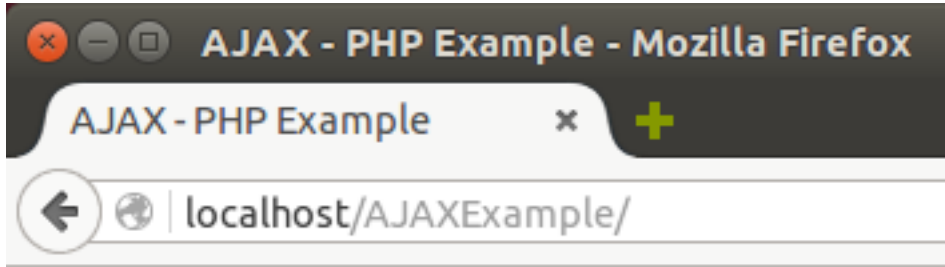
if ($conn->query($sql) === TRUE) {
    echo "Yeni kullanıcı eklendi: " . $kad . " " . $ksoyad;
} else {
    echo "Kayıt Başarısız!: ";
}

$conn->close();
}
```

?>

PHP kodlarımızı yazdık. Bu yazdığımız kodlarımızı açıklamaya çalışalım. Öncelikle POST metoduyla gelen değerleri aldık. Daha sonra bir fonksiyon oluşturduk. Bu fonksiyon iki parametre alıyor. Bu parametreler: kad ve ksoyad. Daha sonra fonksiyon içerisinde bu gelen parametreleri veri tabanına kayıt ediyoruz. Burada isterseniz gelen parametreleri boşluk ve doluluğa göre test edebilirsiniz. Gördüğümüz gibi kodlarımız oldukça basit.

Şimdi yazdığımız kodlarımızı test edelim. <http://localhost>'u açalım. Sırasıyla değerlerimizi girelim.





















Kullanıcı Bilgileri:

Ad:

Soyad:

Yeni kullanıcı eklendi: James GOSLING

Görüldüğü gibi herhangi bir problem yok. Kodlarımız gayet düzgün bir şekilde çalışıyor. Başka kayıtlarda ekleyelim. Bir de isterseniz; kayıtlarımızı ekledikten sonra veri tabanı kısmına bakalım. Veri tabanımızın ekran alıntısı aşağıdaki gibi olacaktır.

				kid	kad	ksoyad
<input type="checkbox"/>	 Düzenle	 Kopyala	 Sil	1	Emre Can	ÖZTAŞ
<input type="checkbox"/>	 Düzenle	 Kopyala	 Sil	2	Rıdvan	KARATAŞ
<input type="checkbox"/>	 Düzenle	 Kopyala	 Sil	3	Cem	İKTA
<input type="checkbox"/>	 Düzenle	 Kopyala	 Sil	4	Akın	KALDIROĞLU
<input type="checkbox"/>	 Düzenle	 Kopyala	 Sil	5	Rozerin	AKTAŞ
<input type="checkbox"/>	 Düzenle	 Kopyala	 Sil	6	James	GOSLING

Görüldüğü gibi AJAX ile gönderdiğimiz parametreler veri tabanına düzgün bir şekilde kaydolmuş durumda. Herşey basit. Sadece AJAX'ı nasıl kullanacağınızı bilmeniz yeterli.

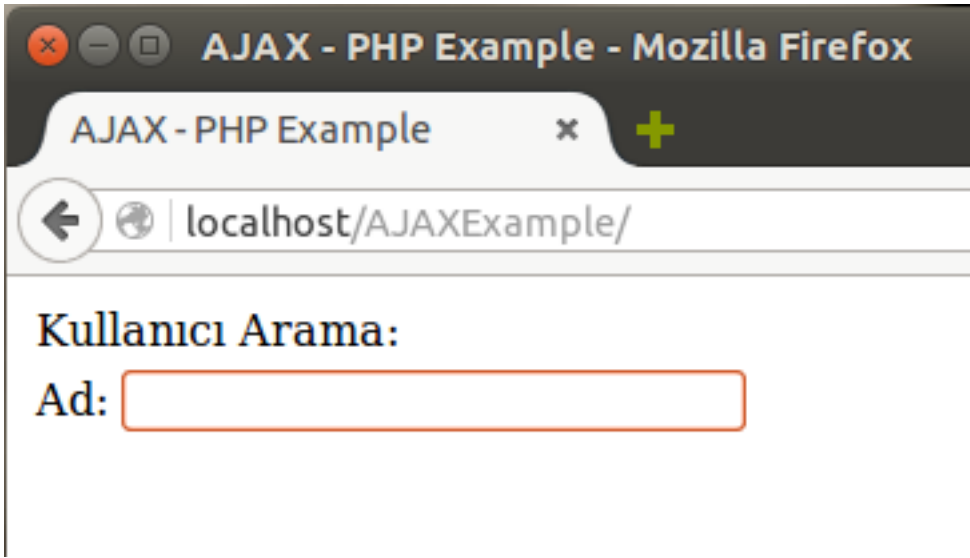
Örneklerimiz oldukça basit. Burada dikkat etmeniz gereken belirli durumlara göre önlemlerinizi almak. Örneğin; SQL Injection, alanların boşluk/doluluk kontrolü, karakterlerin kontrolü v.s gibi durumları gözönünde bulundurmalısınız. Bu kitabın temel mantığı AJAX'ı öğretmek. O yüzden sıraladığım durumlar üzerinde durmuyorum. Ki zaten eğer AJAX öğrenecek seviyeye gelmişseniz bu durumlardan haberdarsınızdır.

Herneyse bir sonraki örneğimizde; AJAX ile basit bir arama işlemi nasıl gerçekleştirilir bunun üzerinde duralım. Veri tabanımız aynı kalsın.

ÖRNEK - 5:

Bu örneğimizde kullanıcının girdiği karakterlere göre veri tabanından arama işlemi gerçekleştirilelim. Bir önceki örneğimizde kullandığımız veri tabanımız üzerinden örneğimizi anlatmaya çalışalım.

Basit bir form tasarlayalım. Bu formda bir input alan ve bir label olsun. Kullanıcı input alanına yazdığı karaktere göre veri tabanından sorgulama yapalım ve sonuçları label alanında gösterelim. Bu sorgulama işlemi veri tabanındaki kad alanına göre yapalım. Dilerseniz ksoyad alanında sorgulama yapabilirsiniz veya ekleyeceğimiz bir select elementi ile kullanıcının hangi alanda sorgulama yapmak istediğini de seçtirebilirsiniz. Tasarlayacağımız form aşağıdaki gibi olsun.



Yukarıdaki tasarlamış olduğumuz sayfanın HTML kodları da aşağıdaki gibi olacaktır.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
  <title>AJAX - PHP Example</title>
<script type="text/javascript" src="xHttp.js"></script>
</head>
<body>
<table>
<tr><td colspan="2">Kullanıcı Arama:</td></tr>
<tr>
<td>Ad:</td><td><input type="text" id="kad"></td>
</tr>
<tr>
<td colspan="2"><label id="sonuc"></label></td>
</tr>
</table>
</body>
</html>
```

PHP tarafında yapacağımız işlemlere geçmeden önce yazacağımız kodumuz hakkında biraz düşünelim. Kullanıcı gelecek, input alanına aramak istediği ismi yazacak. Tamam buraya kadar herhangi bir sıkıntı yok. Peki kullanıcının girdiği ismi nasıl arayacağız? Kullanıcı tam ismi girdiği zaman mı arama yapılacak yoksa Google veya Facebook'taki gibi girilen karakterlere göre mi arama yapılacak? Bu örneğimizde girdiğimiz karaktere göre arama yapalım. Yani kullanıcı e harfine bastığı zaman isminin baş harfi e olanları listeleyelim. Ya da kullanıcı c harfine bastığı zaman isimleri c ile başlayanları listeleyelim. Eğer kullanıcı karakterleri girmeye devam ederse; tam isme göre aramamız daralacaktır.

O halde AJAX kodlarımızı yazalım. Sayfamızın tam hali aşağıdaki gibi olacaktır.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - PHP Example</title>
  <script type="text/javascript" src="xHttp.js"></script>
</head>
```

```

<body>
<table>
<tr><td colspan="2">Kullanıcı Arama:</td></tr>
<tr>
<td>Ad:</td><td><input type="text" id="kad"
onkeyup="sendRequest()"></td>
</tr>
<tr>
<td colspan="2"><label id="sonuc"></label></td>
</tr>
</table>
<script type="text/javascript">
function sendRequest() {
var ajaxObject = XMLHttpRequest(ajaxObject);
var kad = document.getElementById("kad").value;
ajaxObject.onreadystatechange = function() {
if(ajaxObject.readyState == 4 && ajaxObject.status == 200) {
document.getElementById("sonuc").innerHTML =
ajaxObject.responseText;
if(kad == "") document.getElementById("sonuc").innerHTML = "";
}};
ajaxObject.open("GET", "phpFile.php?kad=" + kad, "TRUE")
ajaxObject.send();
}
</script>
</body>
</html>

```

Bu örneğimizi gerçekleştirebilmek için input alanının onkeyup event (olay)'ını kullanmamız gerekli. Bunu unutmayalım. Kullanıcının input alanında her bastığı karakterde sendRequest () fonksiyonumuz çalışacak ve aldığı bu karakteri PHP dosyasına ileticek. Burada dikkat etmemiz gereken bir diğer konuda; eğer input alanı boşsa veya kullanıcı yazdığı karakterleri silmişse sonuçları temizlememiz gerekiyor. Aksi halde yaptığı aramadaki sonuçlar kalır.

Bu AJAX kodlarımızda GET metodunu kullandık. Daha öncede dediğim gibi bu tarz arama işlemlerinde POST metodunu kullanmalısınız ve SQL Injection için önlemler almalısınız. POST metoduyla gönderimi öğrenmiştik bir önceki örneğimizde. Siz de bu örneğimizi POST ile gerçekleştirmeye çalışın. Bunu da ödev olarak kabul edin.

Herneyle PHP tarafına geçelim. Kodlarımız aşağıdaki gibi olacaktır.

```

<?php
$kad = $_GET["kad"];
searchOnDb($kad);

function searchOnDb($kad){
$conn = new mysqli("localhost", "root", "", "uyeler");
mysqli_set_charset($conn,"utf8");
if ($conn->connect_error) {
die("Bağlantı başarısız!: " . $conn->connect_error);
}

$sql = "SELECT * FROM uyelerTablosu WHERE kad like '$kad%'";
$result = $conn->query($sql);

if ($result->num_rows>0) {
while($row = $result->fetch_assoc()) {
echo $row["kad"]. " " . $row["ksoyad"]. "<br>";
}
}
}

```

```
}  
} else {  
    echo "Herhangi bir kayıt bulunamadı!";  
}  
$conn->close();  
}  
?>
```

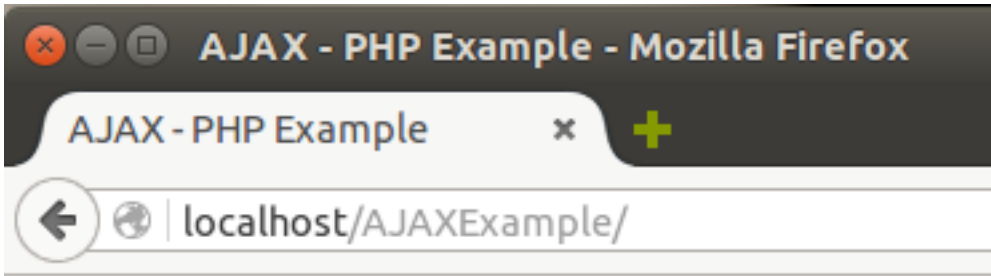
Kodlarımız hakkında biraz konuşalım. Öncelikle gelen değeri yani kad değerini alacağız. Daha sonra yazdığımız fonksiyonumuzu çağıracağız. Örneklerimizi fonksiyonlar şeklinde anlatmaya çalışıyorum. Çünkü bu fonksiyon mantığını anladığınız takdirde diğer işlemler içinde yeni fonksiyonlar yazarak gelişmiş bir sonuç ortaya koyabilirsiniz.

kad değeri geldiği zaman veri tabanından kayıtları çekip bu kad değerine göre sorgulama yapmamız gerekiyor. Sorgulama işleminde dikkat edeceğimiz kavram SQL yazımı. SQL ifademize yakından bakalım.

```
$sql = "SELECT * FROM uyelerTablosu WHERE kad like '$kad%'";
```

Yukarıdaki ifademizin sonunda % işaretini kullandık. Bunu yapmamızın nedeni daha öncede bahsettiğimiz gibi kullanıcının gireceği karaktere göre isimlerin başlarında girilen bu kelimeyi aramak. Eğer %\$kad şeklinde yazmış olsaydık isimlerin sonunda ve %\$kad% şeklinde yazmış olsaydık; kullanıcının gireceği karakterleri isimlerin içerisinde arama yapardı. Buna çok dikkat edelim. Bundan sonra geriye belirtilen kriterlerdeki kayıtları alıp listelemek kalıyor.

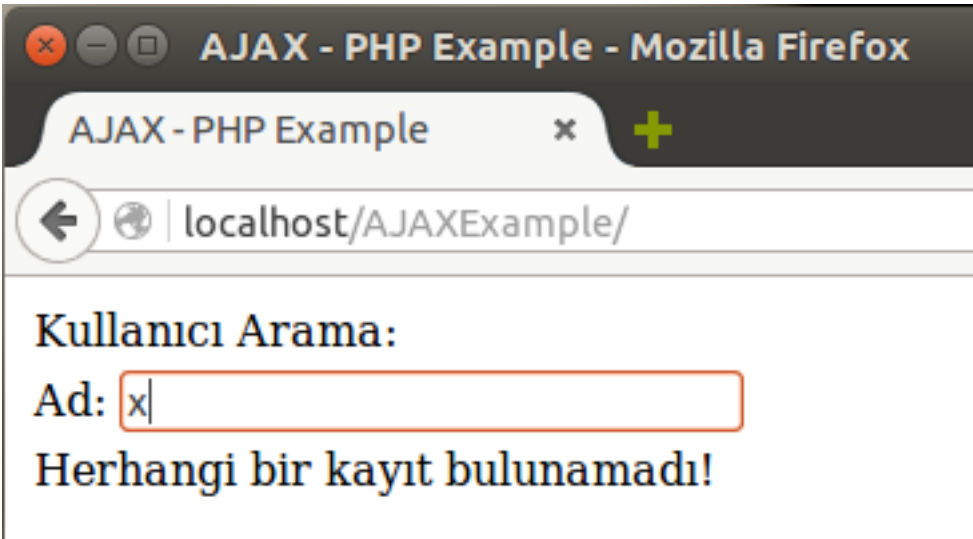
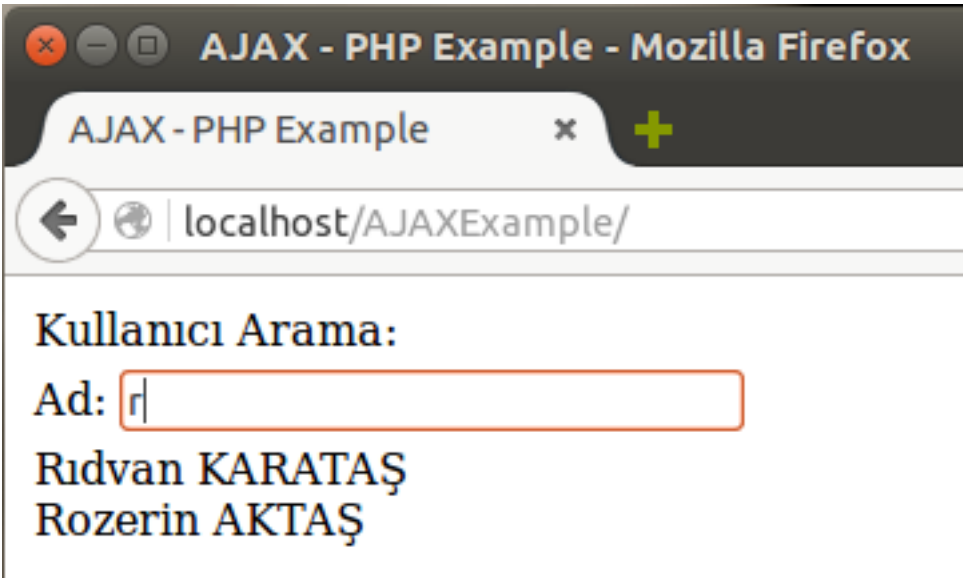
Gördüğümüz gibi kodlarımız çok basit. Şimdi bu kodlarımızı test edelim. Bakalım doğru sonuçlar veriyor mu?



Kullanıcı Arama:

Ad:

Emre Can ÖZTAŞ



Gördüğümüz gibi kodlarımız doğru çalışıyor. Kullanıcı birden fazla karakter girerse veya tam isim girerse arama alanı daralacak ve ona uygun sonuçları getirecektir.

BÖLÜM 7:

JSP ile AJAX

Bu bölümde temel olarak JSP ile AJAX kullanımını üzerinde durmaya çalışacağız. Aslında burada ki amacım Servlet mimarisini kullanarak AJAX kullanımını anlatmak. Java Web'te Servlet mimarisi oldukça yararlı. Şunu da belirtmek isterim ki Java Web yapısında Servlet olmasaydı herhalde Web programlama yapamazdım. Bu açık ve net. Ayrıca bu bölüm bir önceki bölüm kadar uzun olmayacak. Çok basit bir örnek yapacağız. Çünkü bir önceki bölümde oldukça detaylı olarak AJAX kullanımını anlattık. Temelde hepsinin mantığı aynı. Sadece AJAX yapısını iyi bilmek ve kullanılacağı platforma hakim olmak gerekiyor.

Herneyse, öncelikle hiç JSP veya Java Web bilgisi olmayan kullanıcılar için basit olarak bir çalışma dosyası nasıl oluşturulur ve bir Server nasıl tanımlanır bundan bahsetmek istiyorum. Şayet bu konulara yabancı değilseniz sizi bölüm sonuna alayım. Çünkü bölüm sonunda örneğimizi yapacağız.

Öncelikle kullanacağımız ya da benim kullandığım platform ve araçlar aşağıdaki gibidir.

- JDK 1.8.0_73
- Eclipse EE (Mars - 4.5.1)
- Apache Tomcat 8.0.32

Bilgisayarınızda JDK ve Eclipse'nin kurulu olduğunu varsayıyorum. Dilerseniz NetBeans ile de çalışabilirsiniz lakin NetBeans'teki aşamaları adım adım anlatmayacağım. Eclipse üzerinde duracağız.

Bilgisayarınızda JDK veya Eclipse yoksa sorun değil.

Aşağıdaki adresten JDK'yı indirebilirsiniz.

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Aşağıdaki adresten de Eclipse'yi indirebilirsiniz.

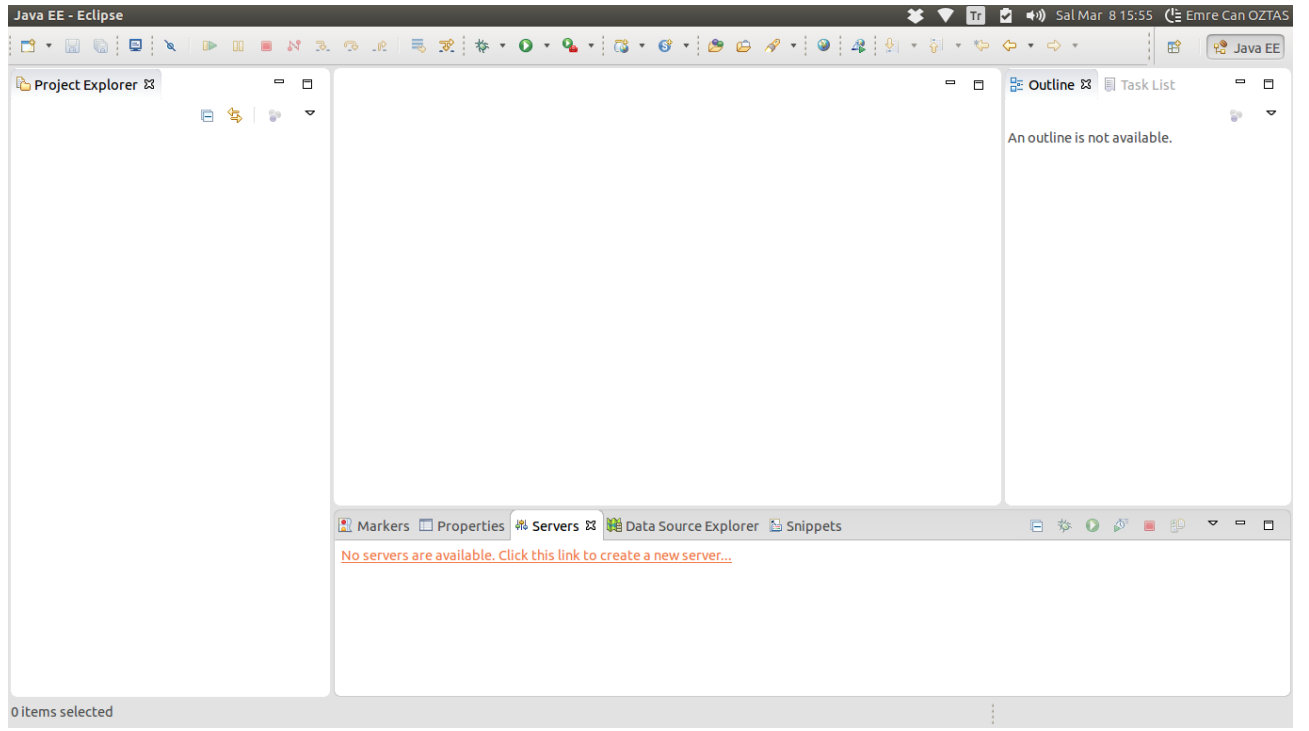
<https://eclipse.org/downloads/>

Şimdi gelelim en önemli meseleye: Apache Tomcat'i indirmeye ve Eclipse'ye tanıtımını yapmaya.

Aşağıdaki adrese gidelim Apache Tomcat'i indirelim. Kurulum dosyası yerine; paketlenmiş sürümü indirmemiz gerekiyor. Zira bilgisayarınıza kurulumunu da yapabilirsiniz lakin Tomcat 8080 no'lu portu kullandığı için diğer programlarla çakışma yaşayabilirsiniz. Örneğin Oracle DB de 8080 no'lu portu kullanır. Tomcat'in portunu değiştirebilirsiniz fakat ben bu işlerle fazla uğraşmak yerine ihtiyacım olduğunda açıp, işim bittiğinde kapatmayı yeğliyorum. Tabi ki karar sizin.

<http://tomcat.apache.org/download-80.cgi>

Herneyse Apache Tomcat indirme sayfasında Core başlığı altından; Windows kullanıcıları: .zip uzantılı dosyayı ve GNU / Linux kullanıcıları .tar.gz uzantılı dosyayı indirmelidirler. İndirmiş olduğumuz Apache Tomcat'i herhangi bir dizine çıkaralım. Örneğin ben Ubuntu kullandığım için Documents dizinine çıkartacağım. Her şey bu kadar basit şimdi geriye sadece Eclipse'de Apache Tomcat'in tanıtılması kalıyor. Eclipse'yi açalım.



Eclipse'ye Tomcat'i tanıtmannın bir kaç yolu var. Bu yollardan bahsedecek olursak;

Aşağıdaki panelde görüldüğü gibi Servers başlığı açık durumda. Eğer sizde Servers başlığı açık değilse; Menü: **Window > Show View > Server > Servers**'ü tıklayarak bu pencereyi açabilirsiniz. Bu penceredeki **No servers are available. Click this link to create a new server...**'a tıklayarak Tomcat'i tanıtabilirsiniz.

Diğer bir yöntemde; Menü: **File > New > Server > Server** seçeneğidir. Buradan da Tomcat'i tanıtabiliriz.

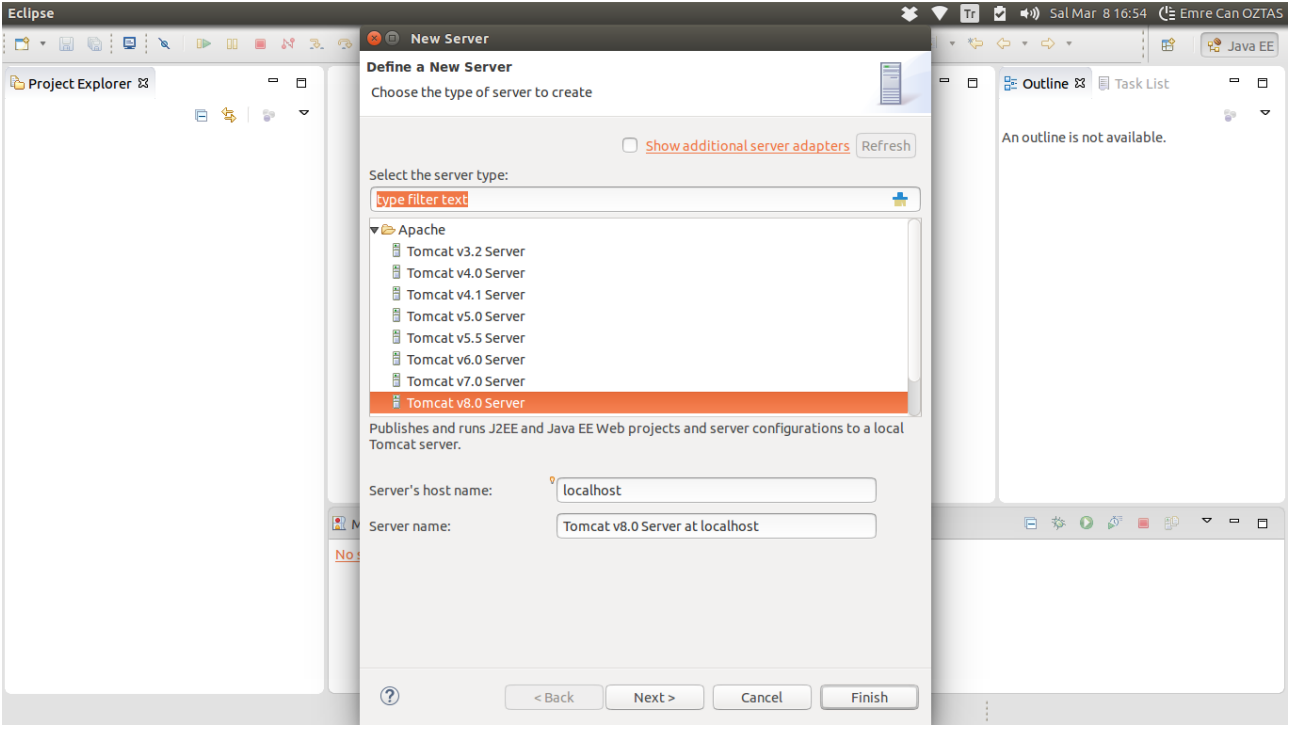
Yine aynı şekilde soldaki Project Explorer penceresi üzerinde sağ tıklayıp; **New > Window > Show View > Server > Server** diyerek yeni bir Server tanımlayabilirsiniz. Tabi ki biz Server olarak Apache Tomcat'i kullanacağız.



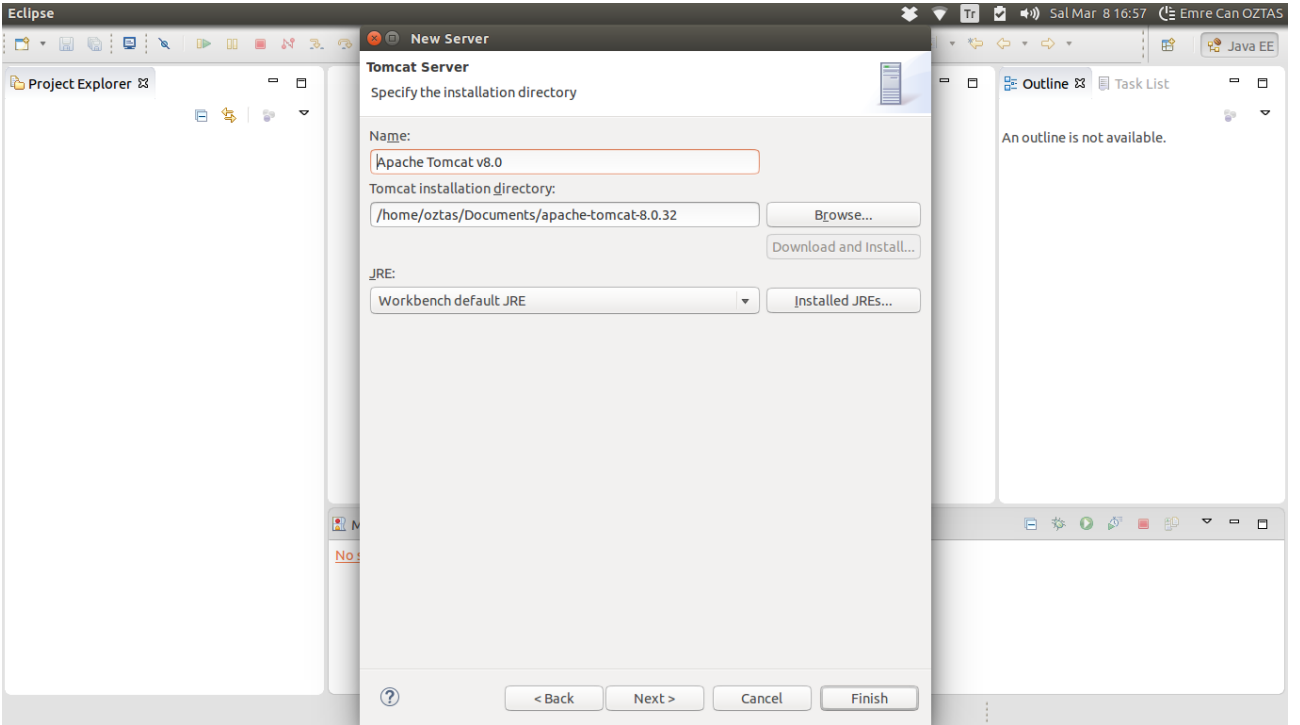
Eclipse üzerinde herhangi bir pencere açık değilse aşağıdaki yol ile istediğiniz pencereyi açabilirsiniz.

Menü: **Window > Show View > acilmekIstenilenPencere**

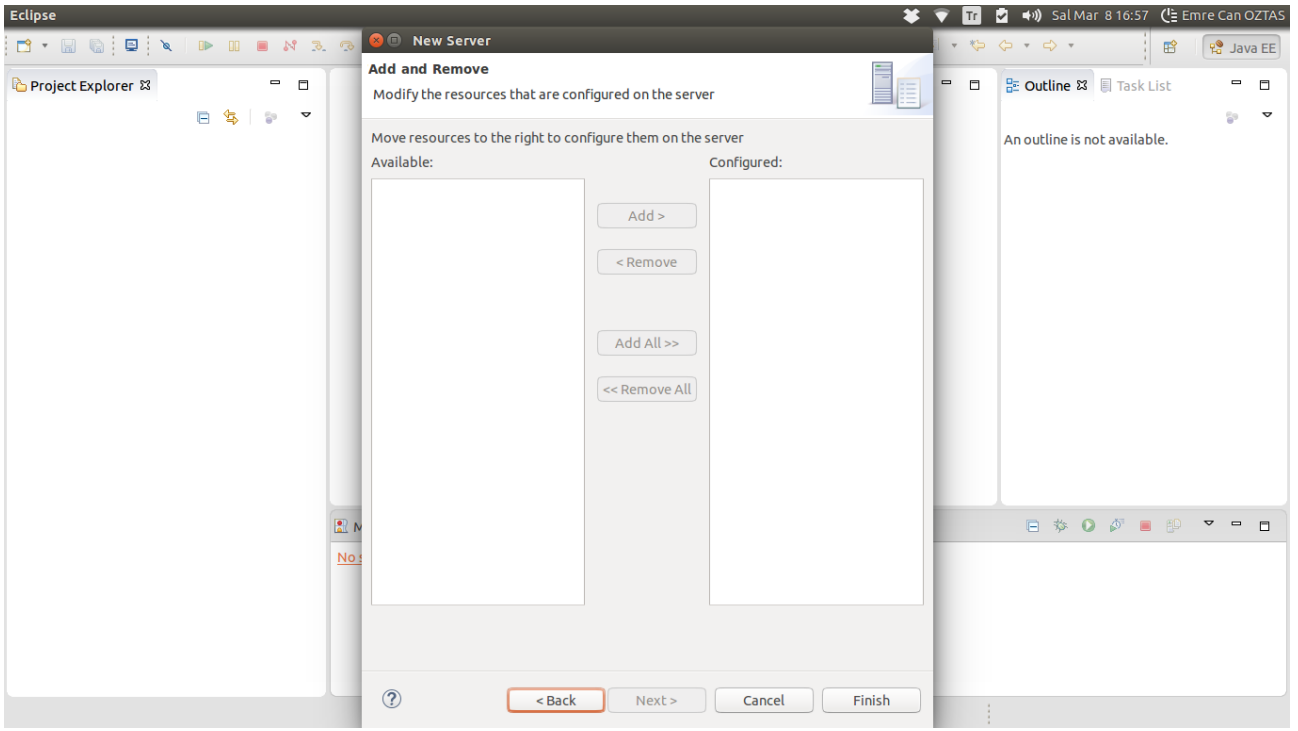
Yukarıdaki seçeneklerden herhangi birisini seçelim. Amacımız Eclipse'e Apache Tomcat'i tanıtmak.



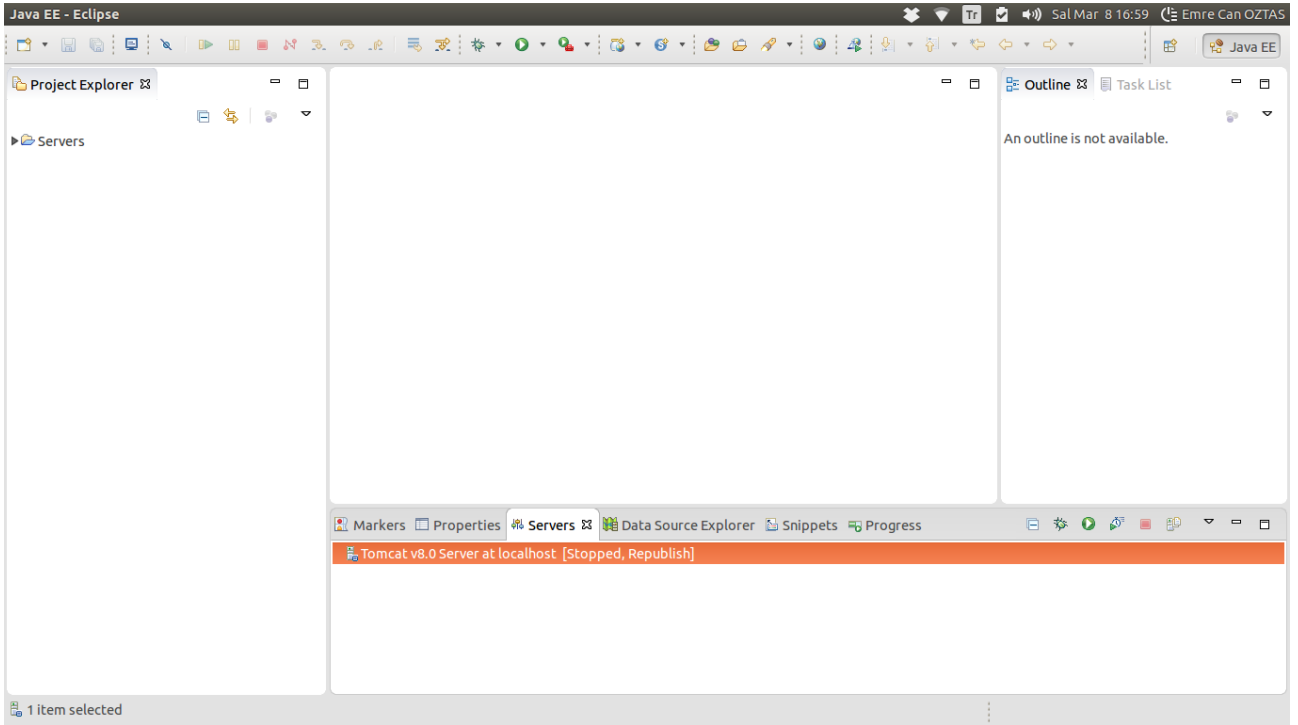
Açılan pencereden; Apache başlığı altından Tomcat v8.0 Server seçeneğini seçelim ve Next >'i tıklayalım. Karşımıza aşağıdaki pencere gelecektir.



Bu pencerede daha önce indirip bir dizine (bende Documents dizini) açtığımız Tomcat'in yolunu göstermeliyiz. Tomcat installation directory başlığı altında, hemen sağ tarafta bulunan Browse butonuna tıklayalım ve Tomcat'in bulunduğu dizini gösterelim. Bunun dışında şimdilik başka herhangi bir değişiklik yapmamıza gerek yok. O yüzden yine Next >'ti tıklayalım.

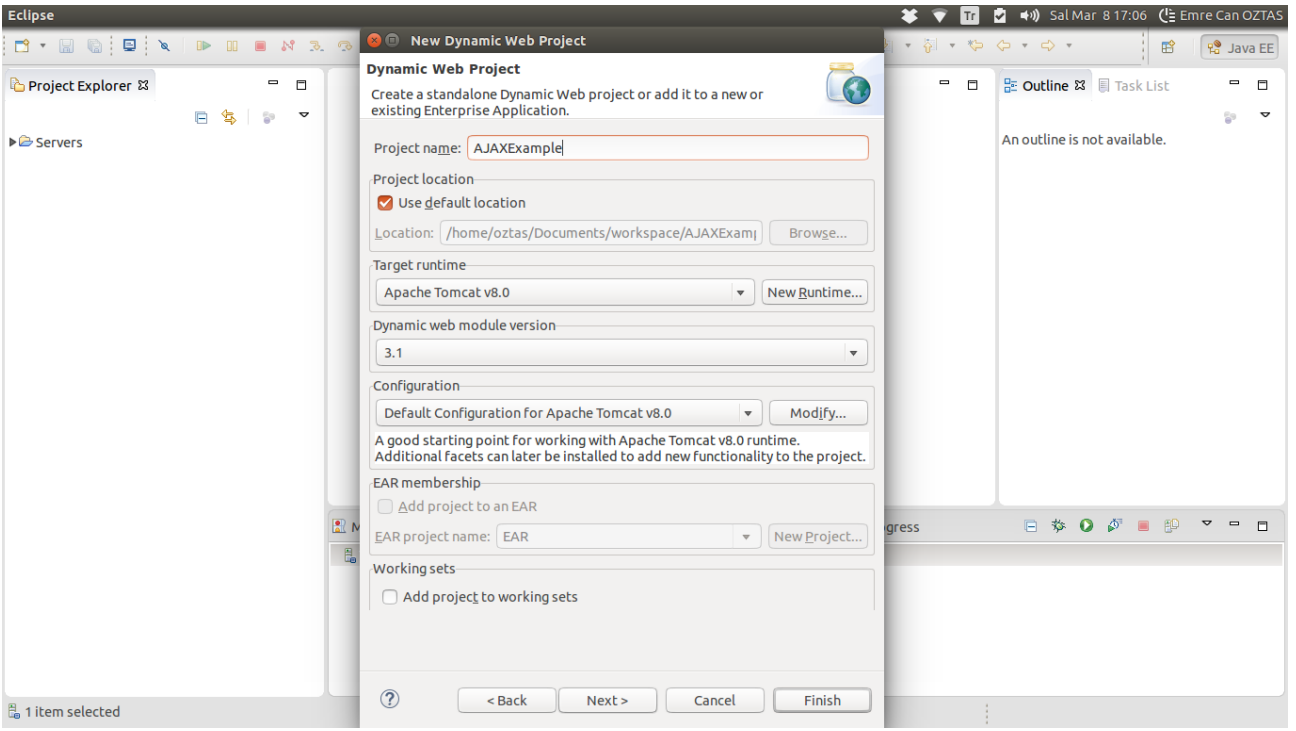


Henüz bir proje açmadığımız için sol taraftaki alan boş olacaktır. Şayet açılmış bir projemiz varsa bunu sağ tarafa taşıyıp Tomcat üzerinde çalıştırabilirdik. Finish'i tıklayalım.

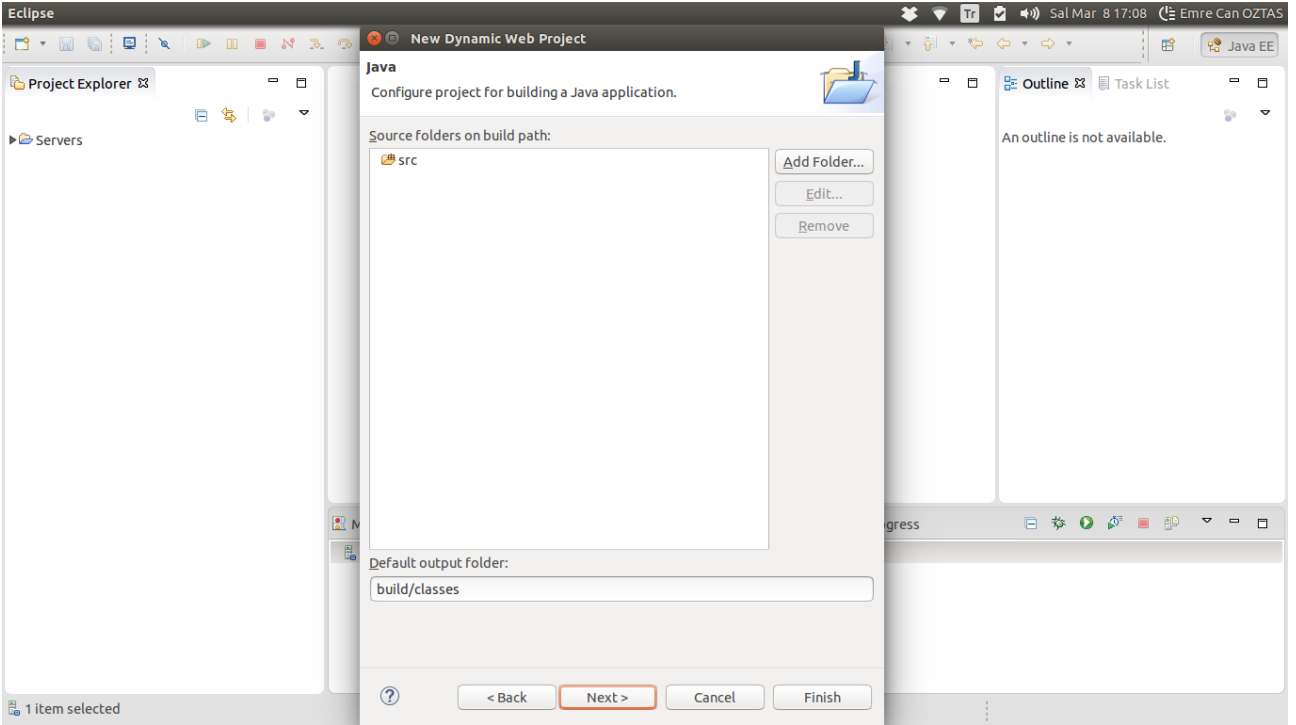


Servers ekranında da görüldüğü gibi yeni Server'ımız oluşturuldu.

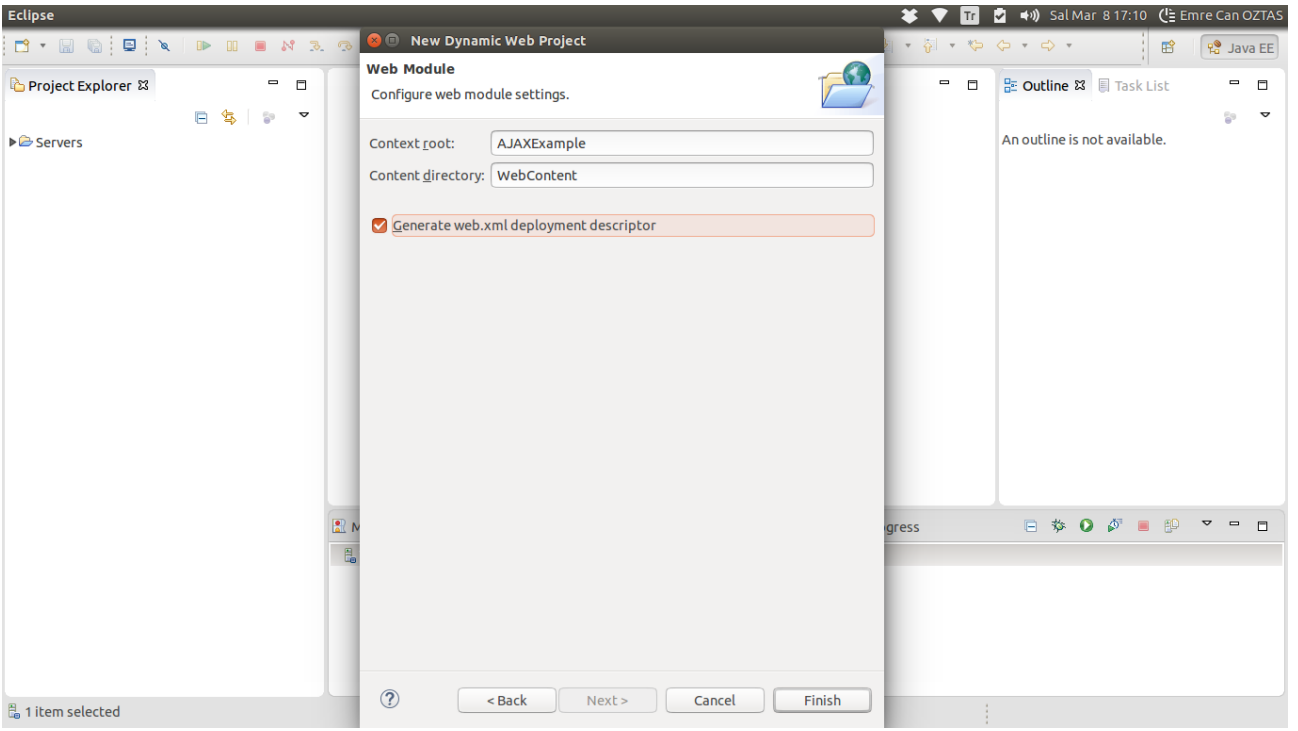
Şimdi bir de proje açalım. Menü > File > New > Web > Dynamic Web Project'i tıklayalım. Aşağıdaki pencere açılacaktır.



Açılan bu sayfadan Project name yazan alana projemizin adını yazalım. Ben her zaman olduğu projemize gibi AJAXExample adını verdim. Şimdilik bu ekranda herhangi bir değişiklik yapmaya gerek yok. O yüzden Next >'ti tıklayalım.

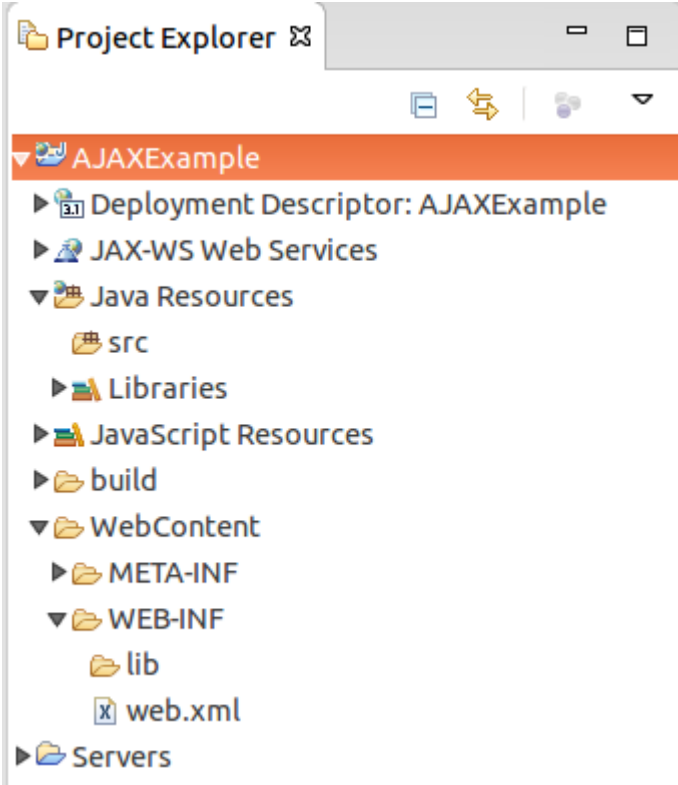


Bu ekranda oluşturulacak src klasörümüz görülmekte. Dilerseniz projenize yeni dosyalar ekleyebilirsiniz lakin biz temel haliyle kullanacağız. O yüzden yine Next >'ti tıklayalım.



Bu son ekranımız. Bu ekranda projemizin adı ve web sayfalarımızın WebContent dosyasında tutulacağı belirtiliyor. Bu isimler üzerinde bir değişiklik yapmayalım. Buraya dikkat. Generate web.xml deployment descriptor seçeneğini işaretleyelim. Çünkü web.xml dosyası bizim için önemli. Projemizin çeşitli bilgilerini barındıracak. web.xml dosyasını oluşturmasınızda sorun yok daha sonra kendiniz tekrar oluşturabilirsiniz.

Finish'i tıklayalım ve projemizi oluşturalım.



Görüldüğü gibi projemiz oluşturuldu. Bu oluşturulan proje dosyasından detaylı olarak bahsetmeyeceğim. Lakin WebContent içerisinde sayfalarımız yer alacak. Java Resources > src dizininde de Servlet'lerimiz yer alacak. Bunu unutmayalım.

Şimdi önemli bir konuya geçelim: web.xml dosyasının düzenlenmesine. Dosyamızı açalım. Dosyamızın içeriği aşağıdaki gibidir.

```
web.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http
3   <display-name>AJAXExample</display-name>
4   <welcome-file-list>
5     <welcome-file>index.html</welcome-file>
6     <welcome-file>index.htm</welcome-file>
7     <welcome-file>index.jsp</welcome-file>
8     <welcome-file>default.html</welcome-file>
9     <welcome-file>default.htm</welcome-file>
10    <welcome-file>default.jsp</welcome-file>
11  </welcome-file-list>
12 </web-app>
```

Design Source

Bu dosyamızda görüldüğü gibi web sitesimizin ilk açılışta hangi sayfadan başlayacağını bahsediyor. Bu örneğimizde .jsp uzantılı bir dosya ile çalışacağımız için index.jsp dışında olan satırları silelim. Çünkü gereksiz çakışmalara sebep olabilirler. Burada belirteceğim diğer bir konuda isterseniz index.html uzantılı bir dosya ya da başka bir isme sahip .jsp veya .html sayfa oluşturabilirsiniz. Web sitenizin hangi sayfadan başlamasını istiyorsanız o dosyayı buraya eklemelisiniz. Örneğin;

```
<welcome-file>emrecaoztas.html</welcome-file>
//veya
<welcome-file>emrecaoztas.jsp</welcome-file>
```

gibi olabilir.

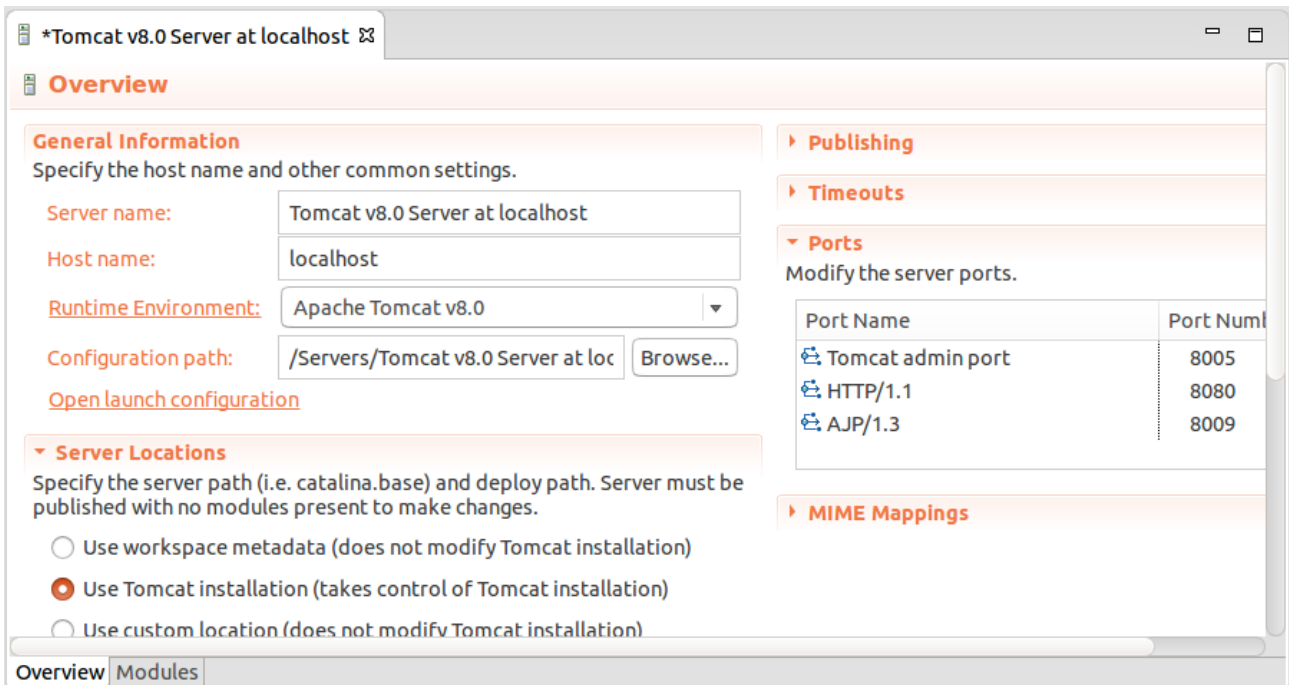
web.xml dosyamızın son hali aşağıdaki gibi olacaktır.

```
web.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   <display-name>AJAXExample</display-name>
4   <welcome-file-list>
5     <welcome-file>index.jsp</welcome-file>
6   </welcome-file-list>
7 </web-app>
```

Design Source

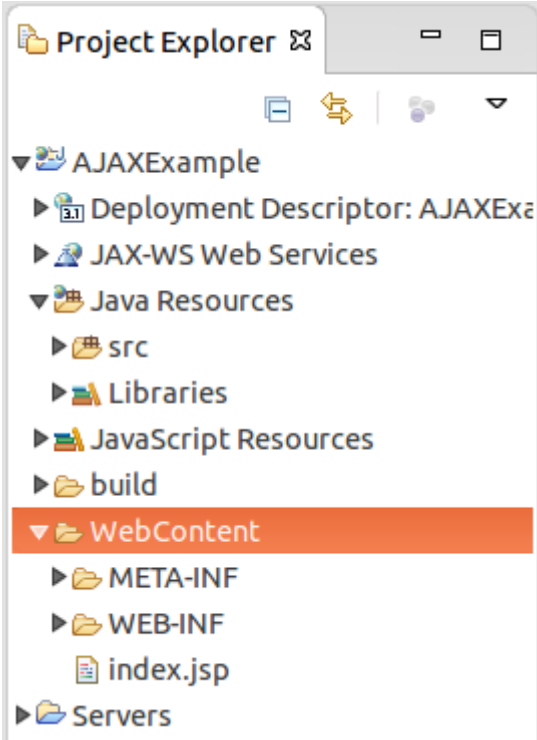
Dosyamızı kaydedelim ve kapatalım.

Son olarak Apache Tomcat'in kontrolünü alalım. Bunun içinde Servers penceresinde tanımladığımız Tomcat'in üzerine gelip çift tıklayalım. Aşağıdaki pencere açılacaktır.



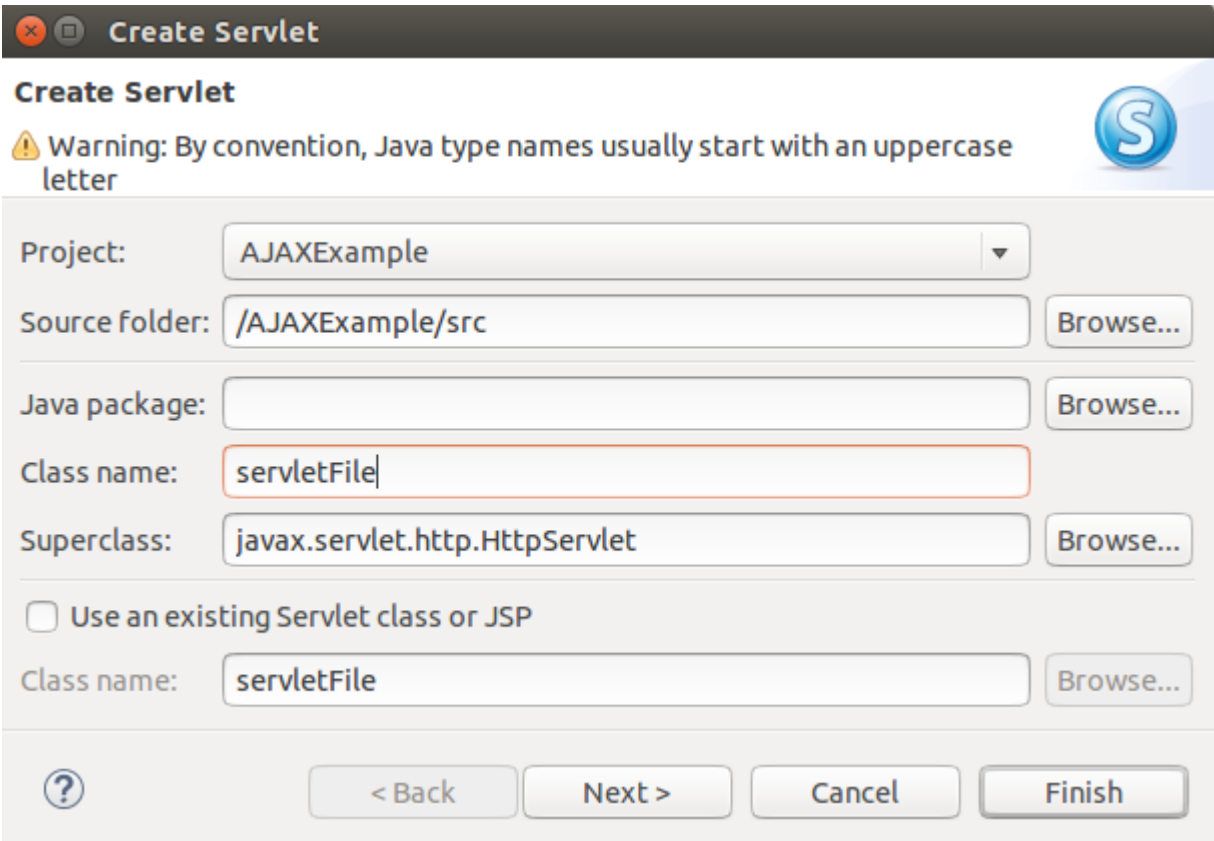
Bu pencerede Server Locations başlığı altındaki seçeneklerden ikinci sırada bulunan Use Tomcat installation (takes control of Tomcat installation) seçeneğini işaretleyelim. Böylelikle artık Tomcat'in kontrolü elimizde olacak. Bu pencereyi de kaydedip çıkalım. Ne kadar çok yorulduk değil mi? Ama daha işimiz bitmedi. Bir de JSP sayfası oluşturalım. Ne demiştik? Ana sayfamızın adı index.jsp olacaktı. Bizde JSP sayfası oluşturalım. Bunun içinde en basitinden

WebContent'in üzerine gelip sağ tıklayıp; New > JSP File'ı seçelim. Açılan pencereden sayfamıza bir isim verelim ve Finish'i tıklayalım. Sayfamızı oluşturalım.



Görüldüğü gibi sayfamız hazır. Son olarak daha önce oluşturduğumuz xHttp.js dosyamızı da WebContent'in içerisinde yani index.jsp sayfamızın yanına koyalım.

Bir de Servlet dosyası oluşturalım. Oluşturacağımız Servlet arkaplanda çalışacak ve bizim isteklerimize yanıt verecek. Bunun içinde basitçe Java Resources altındaki src klasörünün üzerine gelip sağ tıklayıp, New > servlet dememiz yeterli. Aşağıdaki pencere açılacaktır.



Açılan bu pencereden Class name yazan alana Servlet'imizin adını yazalım. Ben basitçe servletFile dedim. Pencere bize uyarı veriyor. Çünkü herhangi bir paket oluşturmadık. Şimdilik oluşturmaya da gerek yok. Bu sayfada kesinlikle başka bir alanı değiştirmeyin. Tecrübesizseniz sonradan başınıza iş açarsınız. O yüzden herşey aynı kalsın. Next >'i tıklayalım.

The screenshot shows the 'Create Servlet' dialog box. The 'Name' field is filled with 'servletFile'. The 'Description' field is empty. Below the 'Initialization parameters' section, there is a 'URL mappings' section with a list containing '/servletFile'. A 'URL Mappings' sub-dialog is open, showing a 'Pattern' field with '/servletfile' entered. The 'URL mappings' section in the main dialog has buttons for 'Add...', 'Edit...', and 'Remove'. At the bottom, there are buttons for '< Back', 'Next >', 'Cancel', and 'Finish'.

Açılan bu sayfadan URL mappings penceresindeki /servletFile'i seçelim ve sağ taraftaki Edit... butonuna tıklayalım. Görüldüğü gibi ekranın ortasında küçük pencere açılacaktır. Burada aslında basitçe mapping yapıyoruz. Örneğin Servlet'imizin adı servletFile ama biz servletfile olarak değiştirdik. Peki neden böyle yaptık? Web sayfaları genellikle küçük harflerle yazılır. O yüzden bizde Servlet'imizin bi nevi mahlasını değiştirdik. İsmi yine servletFile fakat Server'da artık servletfile olarak bilinecek.

Servlet'imizin ismini değiştirdikten sonra Next >'i tıklayalım. Bu gelen pencere son penceredir.

Create Servlet

Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: public abstract final

Interfaces:

Which method stubs would you like to create?

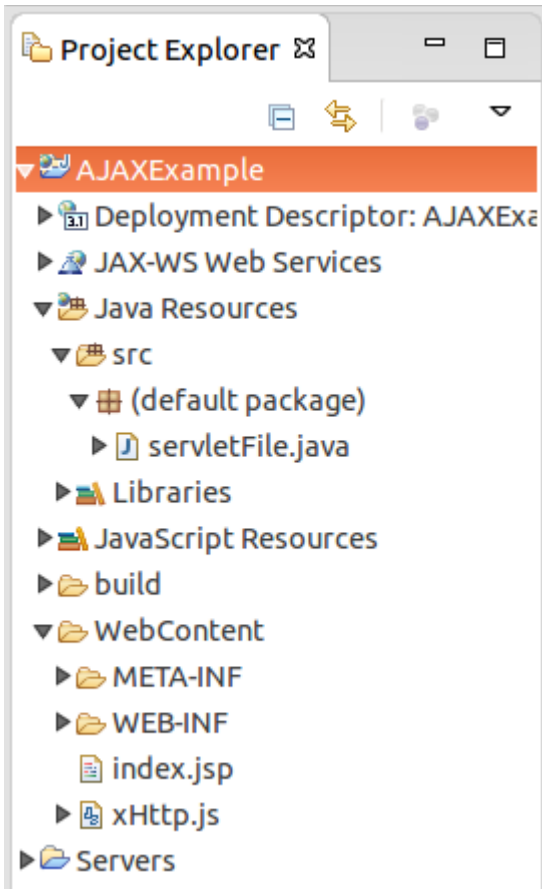
Constructors from superclass

Inherited abstract methods

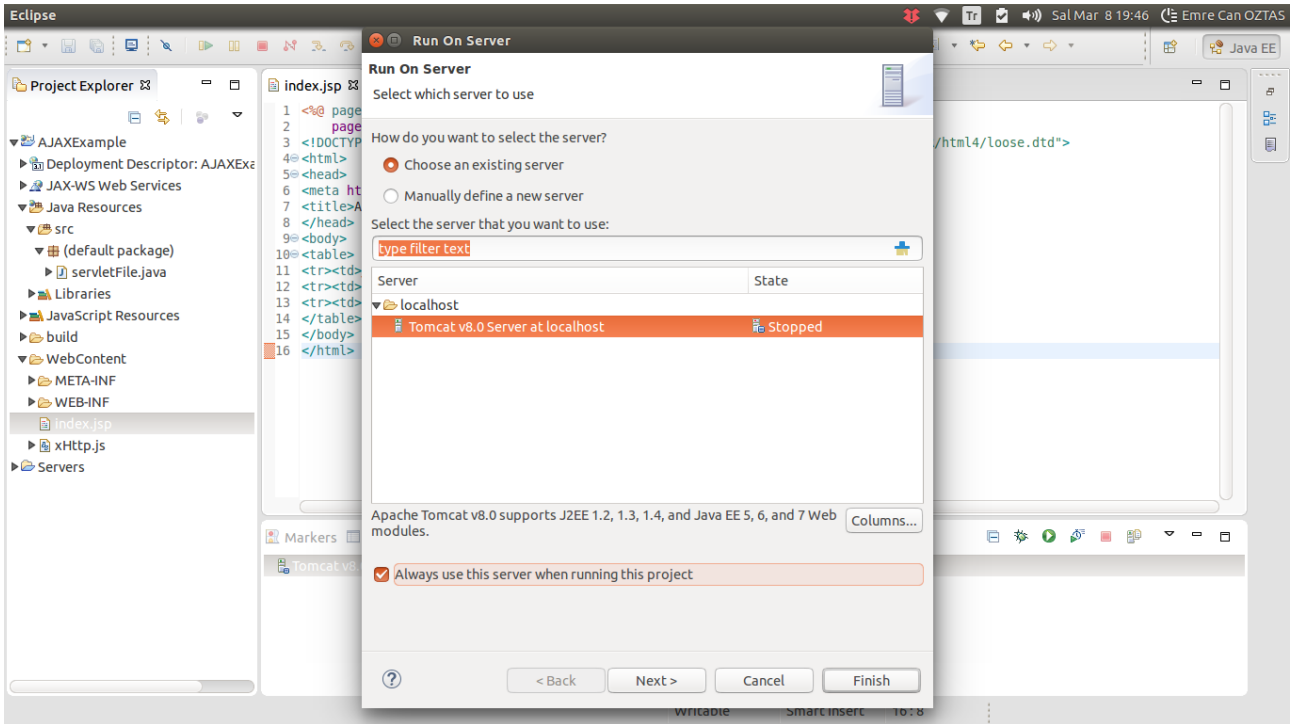
<input type="checkbox"/> init	<input type="checkbox"/> destroy	<input type="checkbox"/> getServletConfig
<input type="checkbox"/> getServletInfo	<input type="checkbox"/> service	<input checked="" type="checkbox"/> doGet
<input checked="" type="checkbox"/> doPost	<input type="checkbox"/> doPut	<input type="checkbox"/> doDelete
<input type="checkbox"/> doHead	<input type="checkbox"/> doOptions	<input type="checkbox"/> doTrace

Bu pencerede sizden ricam Constustors from superclass seçeneğini seçmemeniz. Sakın böyle bir hataya düşmeyin. Sonra başınıza çok büyük bir bela alırsınız benden söylemesi. Herneyse, sadece Inherited abstract methods, doGet ve doPost seçeneklerini işaretleyelim. Son aşama olarak Finish'i seçerek Servlet'imizi oluşturalım.

Project Explorer penceresinin son hali aşağıdaki gibi olacaktır.



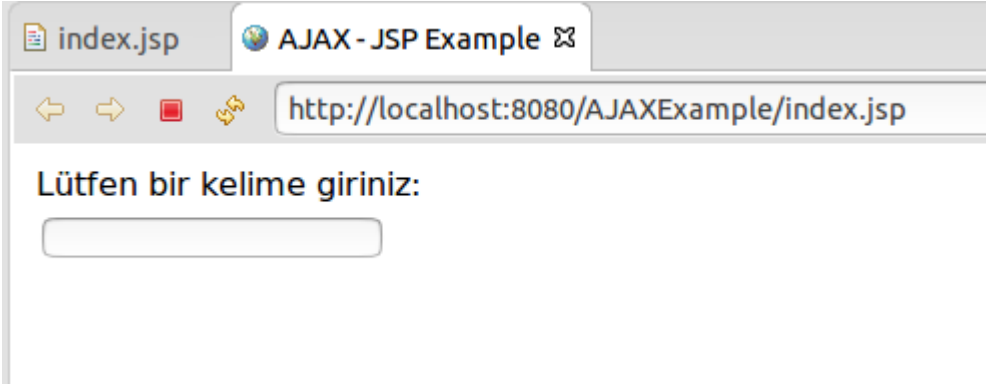
Artık hazırız. AJAX kodlamaya başlayabiliriz. Öncelikle index.jsp sayfasını açalım ve sayfamıza xHttp.js dosyasını ekleyelim. Bu örneğimizde basit olarak; kendisine gelen kelimenin karakterlerini sıra numarası olarak çift / tek olarak ayırсын ve çiftleri yazdırsın. Bir form tasarlayalım bu formda sadece bir input alanı ve bir de label alanı olsun. Formumuzu tasarladıktan sonra run butonuna tıklayalım ya da sayfamız üzerindeyken sağ tıklayıp Run As > Run on Server seçeneğini seçerek sayfamızı Tomcat üzerinde çalıştıralım. Sayfamızı ilk çalıştırdığımızda aşağıdaki pencere gelecektir.



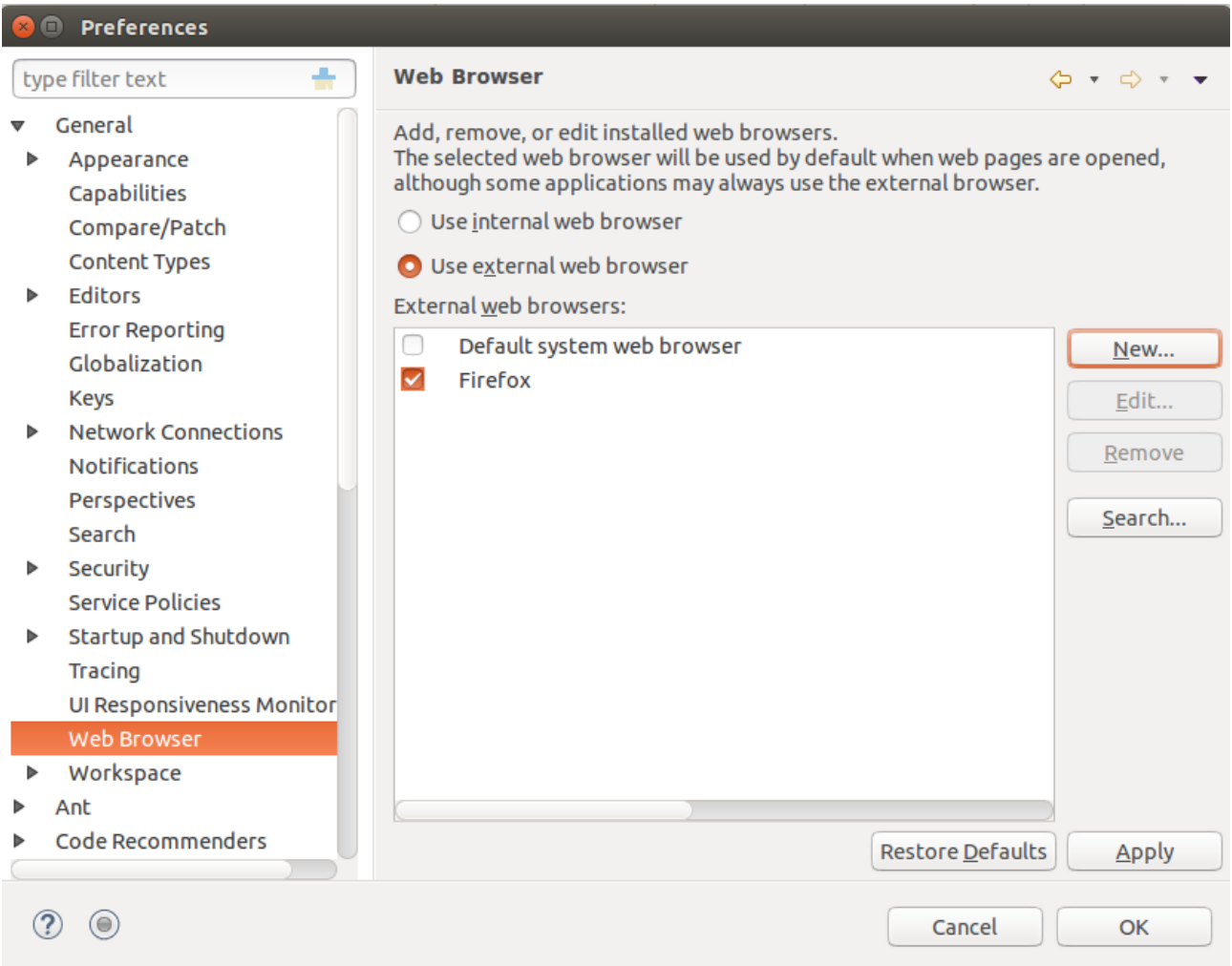
Bu pencerenin en altındaki Always use this server when running this project seçeneğini işaretleyelim. Bu seçeneği işaretlememizin amacı her run ettiğimizde Tomcat otomatik olarak başlaması

içindir. Finish'i tıklayalım.

Sayfamız Eclipse içinde açılan bir pencerede gösterecektir. Aşağıdaki ekran alıntısında olduğu gibi.

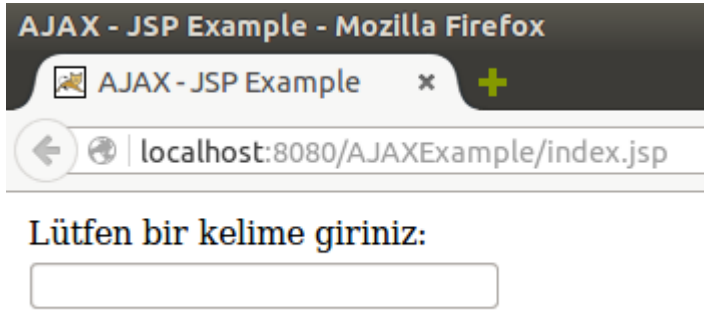


Görüldüğü gibi çok çirkin bir ekran. Ama mühim değil. Browser (Tarayıcı)'yı değiştirebiliriz. Bunun için: Menü > Window > Web Browser > istenilenWebBrowser seçeneği ile istediğimiz tarayıcıda sayfamızı çalıştırabiliriz. Şayet bilgisayarınızdaki yüklü olan tarayıcı burada listelenmemişse önemli değil. Menü > Window > Preferences > General > Web Browser seçeneği ile istediğiniz ama listede olmayan tarayıcıyı ekleyebilirsiniz. Aşağıdaki ekranda olduğu gibi.



Yukarıdaki resimde de görüldüğü gibi sağ taraftaki New... butonuna tıklayarak istediğiniz tarayıcıyı buraya ekleyebilirsiniz. Ben genelde Firefox kullandığım için HTML 5 projeleri dışındaki diğer projelerde başka tarayıcılara pek fazla ihtiyaç duymuyorum.

Herneyse konuyu fazla dağıtmayalım. Hazırladığımız formun Firefox üzerindeki görüntüsü aşağıdaki gibi olacaktır.



Bu daha güzel bir görüntü değil mi?

Hazırlamış olduğumuz formun HTML kodları da aşağıdaki gibi olacaktır.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - JSP Example</title>
    <script type="text/javascript" src="xHttp.js"></script>
</head>
<body>
<table>
<tr><td>Lütfen bir kelime giriniz:</td></tr>
<tr><td><input type="text" id="word"></td></tr>
<tr><td><label id="result"></label></td></tr>
</table>
</body>
</html>
```

Şimdi AJAX kodlarımızı yazabiliriz. Mantık yine aynı. AJAX yapımızı kuracağız. Daha sonra girilen kelimeyi servletfile'a göndereceğiz. Yani değişen herhangi bir şey olmayacak.

Yazdığımız kodların tamamı da aşağıdaki gibi olacaktır.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - JSP Example</title>
    <script type="text/javascript" src="xHttp.js"></script>
</head>
<body>
<table>
<tr><td>Lütfen bir kelime giriniz:</td></tr>
<tr><td><input type="text" id="word"
onkeyup="sendRequest()"></td></tr>
<tr><td><label id="result"></label></td></tr>
</table>
```

```

<script type="text/javascript">
function sendRequest() {
    var ajaxObject = XMLHttpRequest(ajaxObject);
    var word = document.getElementById("word").value;
    ajaxObject.onreadystatechange = function() {
        if(ajaxObject.readyState == 4 && ajaxObject.status == 200)
        {
            document.getElementById("result").innerHTML =
ajaxObject.responseText;
        }
    };
    /*
    Buraya dikkat!
    Servlet'lerin uzantısı olmaz. Daha önce mahlas'ini
    verdiğimiz şekildeki
    gibi kullanılırlar.
    */
    ajaxObject.open("GET", "servletfile?word=" + word, "true")
    ajaxObject.send();
}
</script>
</body>
</html>

```

JSP sayfamıza AJAX kodlarımızı yazdık. Kodlarımız arasında açıklama yaptım ama yine de değineyim. Diğer web sayfaları gibi Servlet'lerin uzantısı olmaz. Hadd-i zatında Servlet'ler bir Java Class (Sınıf)'dır. HttpServlet sınıfından Extends (Kalıtım anahtar kelimesi, Türkçe genişletme) türemişlerdir. Diğer önemli bir konuya hatırlarsanız Servlet'imizi oluştururken bir mahlas vermiştik. Örneğin; servletFile yerine servletfile demiştik. İşte burada da mahlasını kullanıyoruz yani servletfile. Bu kavramlara çok dikkat edelim.

JSP tarafında yapacaklarımız bitti. Şimdi Servlet tarafına geçelim. Servlet kısmında iki tane metodumuz bulunacak. Bunlar: doGet ve doPost metotları. Biz isteğimizi GET ile yaptık. doGet metodunun içerisini biraz düzenleyelim. Öncelikle gelen parametreyi almamız gerekiyor. Peki alalım o zaman.

```

protected void doGet(HttpServletRequest request, HttpServletResponse
response){
    String word = request.getParameter("word");
}

```

Görüldüğü gibi bir değişken tanımladık ve HttpServletRequest sınıfının bir nesnesi olan request'in metodu olan getParameter() ile gelen parametreyi aldık. Şimdi yapmamız gereken; aldığımız bu parametredeki kelimenin içerisindeki karakterleri çift / tek olarak ayırmak ve çiftleri yazdırmak. Zaten String bir ifade bir Array (Dizi) yapısındadır. Dolayısıyla bu parametrenin ya da kelimenin uzunluğunu bulup, karakterlerini tek tek kontrol etmemiz kalıyor. Bunun için yazacağımız kodlarımızda aşağıdaki gibi olacaktır.

```

protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {
    String word = request.getParameter("word");
    PrintWriter out = response.getWriter();
    for (int i = 0; i < word.length(); i++) {
        if (i % 2 == 0) {
            out.print(word.charAt(i));
        }
    }
}

```

Kodlarımızı yazdık. Bir döngü açtık. Bu döngü; gelen parametrenin uzunluğu kadar olacak. Döngünün her adımında; artan sayının mod 2'ye göre değerine bakacağız. Eğer sonuç çiftse; çift, tekse; tek olacak. Dolayısıyla bizde parametrenin çift karakterlerini yazacağız. Burada Java bilgisine sahip olmayanlar için iki konudan bahsedeceğim. Öncelikle Java Web'te ekrana daha doğrusu sayfaya herhangi bir şey yazdırmak için aşağıdaki kodu yazmamız lazım.

```
// bir yazıcı tanımlamalıyız
PrintWriter out = response.getWriter();
// sonra bu yazıcıyı kullanarak ekrana yazdırmalıyız
out.print("Biseyler...Biseyler...");
```

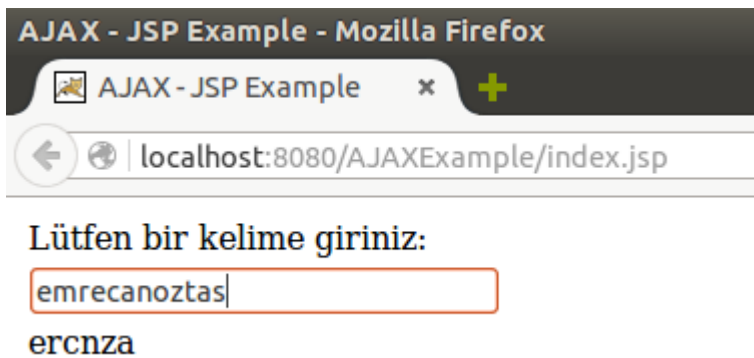
Diğer bahsedeceğim konu ise String yapısıyla ilgili. Daha önce String için bir dizi yapısındadır demiştik. Aşağıdaki satır ile gelen parametrenin karakterleri tek tek alabiliriz.

```
word.charAt(i) // donen deger char olur
```

Kodlarımızın tam hali aşağıdaki gibi olacaktır.

```
servletFile.java
1 import java.io.IOException;
2 import java.io.PrintWriter;
3 import javax.servlet.ServletException;
4 import javax.servlet.annotation.WebServlet;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8
9 @WebServlet("/servletfile")
10 public class servletFile extends HttpServlet {
11     private static final long serialVersionUID = 1L;
12
13     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
14         String word = request.getParameter("word");
15         PrintWriter out = response.getWriter();
16         for (int i = 0; i < word.length(); i++) {
17             if (i % 2 == 0) {
18                 out.print(word.charAt(i));
19             }
20         }
21     }
22 }
23
24
25     protected void doPost(HttpServletRequest request, HttpServletResponse response)
26         throws ServletException, IOException {
27     }
28 }
29
30 }
31
```

Şimdi yazdığımız kodlarımızı deneyelim. Bakalım doğru çalışıyor mu? Örnek ekran çıktımız aşağıdaki gibi olacaktır.



Ekran çıktısında da görüldüğü gibi kodlarımız düzgün bir şekilde çalışıyor.

Son olarak bir şeyler söyleyip bölümü bitirmek istiyorum. AJAX'ı hangi ortamda kullandığımızın bir önemi yok aslında. Sadece nasıl kullanmasını bilmeniz önemli. Eğer AJAX'ı düzgün öğrenmişseniz;

geriye sadece istediđiniz dil ile beraber kullanmanız kalıyor. Bir önceki bölümde PHP ile bolca örnek yaptık. Yaptığımız örneklerin aynısını JSP ile de yapabiliriz lakin bu sadece vakit kaybına neden olur. Hangi programlama dilini kullanırsanız kullanın hepsi aynı temele dayanmakta. O yüzden bir programlama dilinde çok iyi olursanız diğer programlama dillerini de öğrenmek kolaylaşır. Eğer iyi öğrenmek istediđiniz bir dil ararsanız size Java'yı öneririm. Niye diye sormayın. Öğrendikçe neden Java dediđimi çok iyi anlayacaksınız.

BÖLÜM: 8

AJAX: Çoklu Kullanım

Bu bölümde; kitabımızın en başında belirttiğimiz AJAX'ın çoklu kullanımı üzerinde duracağız. Bu çoklu kullanımdan kastım; XMLHttpRequest nesnesi oluşturulduktan sonra bu nesneyi birden fazla işlem için kullanabiliriz. Misal; örneğin bir sayfada 3, 5 belki 10 yerde AJAX kullanmamız gerekebilir. İşte bu ve buna benzer durumlarda yani birden fazla kullanım gereken durumlarda, oluşturmuş olduğumuz nesnemizi kullanabiliriz. Her iş için ayrı bir nesne oluşturmaya gerek yoktur.

Ne demek istediğimizi bir örnek üzerinde anlatmaya çalışalım. Hatırlarsanız beşinci bölümde; Text, XML ve JSON olarak 3 farklı dosya oluşturmuş ve bu dosyalardaki bilgileri okumuştuk. Gelin, bu örneklerimizi birleştirelim ve hepsini bir sayfada paylaşalım. Öncelikle dosyalarımızın konumları aşağıdaki gibiydi, hatırlarsanız.

```
AJAXExample /
|- xHttp.js
|- index.html
|- textFile.txt
|- xmlFile.xml
|- jsonFile.json
```

index.html dosyamızı düzenleyelim ve üç farklı örneğimizde yaptığımız formları birleştirelim. Yeni formumuz aşağıdaki gibi olacaktır.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - JSON Example</title>
  <script type="text/javascript" src="xHttp.js"></script>
</head>
<body>
<!-- TEXT -->
<label>-* TEXT *-</label><br>
<label id="txt">Merhaba!</label>
<input type="button" value="Click"><br><br>
<!-- XML -->
<label>-* XML *-</label><br>
<input type="button" value="Show Books Details"
onclick="sendRequest()"><br><br>
<table id="XML"></table><br><br>
<!-- JSON -->
<label>-* JSON *-</label><br>
<input type="button" value="Show Books Details"
onclick="sendRequest()">
<table id="json"></table>
</body>
</html>
```

Oluşturmuş olduğumuz formun ekran çıktısı da aşağıdaki gibi olacaktır.



.* TEXT *-

Merhaba!

.* XML *-

.* JSON *-

Şimdi kodlarımızı yazalım. Öncelikle bir AJAX nesnesi oluşturmamız gerekiyor. İsterseniz bu nesneyi sayfa Onload (Yükleme) olduğunda oluşturalım. Böylelikle sayfamız aktif olduğu anda nesnemiz hazır olacaktır. Bu işlemi de aşağıdaki gibi yapalım.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - JSON Example</title>
  <script type="text/javascript" src="xHttp.js"></script>
  <script type="text/javascript">
    var ajaxObject
    function createAjaxObject() {
      var ajaxObject = controlXHttp(ajaxObject);
    }
  </script>
</head>
<body onload="createAjaxObject()">
```

Yukarıdaki kodlarımızda görüldüğü gibi sayfa yüklendiği anda AJAX nesnemiz oluşturulacaktır. Daha sonra istediğimiz yerde çağırıp kullanacağız. Diğer üç işlem içinde (Text, XML ve JSON) farklı fonksiyonlar yazmamız gerekiyor. Çünkü kullanıcı hangisini seçerse o işlem yapılmalı. Bundan sonrası sadece AJAX kodları. Daha önce yaptığımız örneklerimizi birleştireceğiz. O yüzden kodlarımızın tam hali aşağıdaki gibi olacaktır.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AJAX - JSON Example</title>
  <script type="text/javascript" src="xHttp.js"></script>
  <script type="text/javascript">
    var ajaxObject;
    function createAjaxObject() {
      ajaxObject = controlXHttp(ajaxObject);
    }
  </script>
```



```

</head>
<body onload="createAjaxObject()">
<!-- TEXT -->
<label>-* TEXT *-</label><br>
<label id="txt">Merhaba!</label>
<input type="button" value="Click" onclick="getText()"><br><br>
<!-- XML -->
<label>-* XML *-</label><br>
<input type="button" value="Show Books Details"
onclick="getXML()"><br><br>
<table id="xml"></table><br><br>
<!-- JSON -->
<label>-* JSON *-</label><br>
<input type="button" value="Show Books Details" onclick="getJSON()">
<table id="json"></table>

<!-- txtFile > TEXT -->
<script type="text/javascript">
    function getText() {
        ajaxObject.onreadystatechange = function() {
            if (ajaxObject.readyState == 4 && ajaxObject.status == 200) {
                document.getElementById("txt").innerHTML =
ajaxObject.responseText;
            }
        };
        ajaxObject.open("GET", "txtFile.txt", true);
        ajaxObject.send();
    }
</script>

<!-- xmlFile > XML -->
<script type="text/javascript">
    function getXML() {
        ajaxObject.onreadystatechange = function() {
            if (ajaxObject.readyState == 4 && ajaxObject.status == 200) {
                var xmlReq = ajaxObject.responseXML;
                var table="<tr><th>Title</th><th>Author</th><th>Page</th></tr>";
                var dataRow = xmlReq.getElementsByTagName("book");
                var i;
                for (i = 0; i <dataRow.length; i++) {
                    table += "<tr><td>" + dataRow[i].getElementsByTagName("title")
[0].childNodes[0].nodeValue +
                    "</td><td>" + dataRow[i].getElementsByTagName("author")
[0].childNodes[0].nodeValue + "</td><td>"+
                    dataRow[i].getElementsByTagName("page")[0].childNodes[0].nodeValue
+ "</td></tr>";
                }
                document.getElementById("xml").innerHTML = table;
            }
        };
        ajaxObject.open("GET", "xmlFile.xml", true);
        ajaxObject.send();
    }
</script>

<!-- xmlFile > XML -->
<script type="text/javascript">
    function getJSON() {
        var
table="<tr><th>Title</th><th>Author</th><th>Page</th><td></td></tr>";

```

```

ajaxObject.onreadystatechange = function() {
  if(ajaxObject.readyState == 4 && ajaxObject.status == 200) {
    var jsonObject = JSON.parse(ajaxObject.responseText);
    for(i = 0; i < jsonObject.books.length; i++){
      table += "<tr><td>" + jsonObject.books[i].title + "</td><td>" +
      jsonObject.books[i].author +
      "</td><td>" + jsonObject.books[i].page + "</td></tr>";
      document.getElementById("json").innerHTML = table;
    }
  }
};
ajaxObject.open("GET", "jsonFile.json", true);
ajaxObject.send();
}
</script>
</body>
</html>

```

Kodlarımızı yazdık. Daha doğrusu önceki örneklerimizle birleştirdik sadece. Artık AJAX biliyorsunuz. O yüzden fazla açıklama yapmıyorum. Formumuzun ilk hali de yukarıda, o yüzden yeni bir ekran alıntısı eklemiyorum.

Sayfamızı tarayıcıda görüntüleyelim ve bakalım herhangi yanlış bir şey yaptık mı ya da kodlarımız doğru çalışıyor mu? Örnek ekran çıktısı aşağıdaki gibi olacaktır.

-* TEXT *-
emrecan-oztas

-* XML *-

Title	Author	Page
Simple JavaScript	Emre Can OZTAS	214
Simple HTML5	Emre Can OZTAS	84
JSON	Emre Can OZTAS	57

-* JSON *-

Title	Author	Page
Simple JavaScript	Emre Can OZTAS	214
Simple HTML5	Emre Can OZTAS	84
JSON	Emre Can OZTAS	57

Formumuzdaki üç butona da sırasıyla tıkladım. Sonuçlar doğru bir şekilde iletildi. Hangi butonu kullanmak isterseniz o butona tıkladığınızda sonuçlar gelecektir. Benim gibi butonların hepsine de tıklayabilirsiniz. Yani burda demek istediğim şey; bir kere oluşturulan AJAX nesnesi ile birden fazla

işlem yapabilirsiniz. Dikkat ederseniz örneğimizde sadece bir AJAX nesnesiyle 3 farklı işlem yaptım. Bu sayı daha da artabilir. Sözün kısası dostlar; her işlem için yeni bir AJAX nesnesi oluşturmanıza gerek yok. Bir defa oluşturmuşsanız; o bütün işlemlerinizi gerçekleştirecektir.

<http://www.w3schools.com/ajax/>

<http://www.tutorialspoint.com/ajax/>

<http://www.tizag.com/ajaxTutorial/>

https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started

<http://www.xul.fr/en-xml-ajax.html>

http://www.webmonkey.com/2010/02/ajax_for_beginners/

<https://www.codeschool.com/courses/jquery-the-return-flight>

<http://www.javatpoint.com/ajax-tutorial>