TRAKYA ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

YÜKSEK LİSANS TEZ ÇALIŞMASI
Konu:Web Uygulamalarında Farklı Veri
Tabanı Erişim Yöntemlerinin Karşılaştırılması
ve Performans Analizinin Çıkarılması

Hazırlayan:Cem ÇUHADAR

Danışman:Yrd. Doç. Dr. Cavit TEZCAN

TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

WEB UYGULAMALARINDA FARKLI VERİ TABANI ERİŞİM YÖNTEMLERİNİN KARŞILAŞTIRILMASI VE PERFORMANS ANALİZİNİN ÇIKARILMASI

Yüksek Lisans Tezi

CEM ÇUHADAR

Bu tez 02-06-2000 tarihinde aşağıdaki jüri tarafından kabul edilmiştir.

Prof. Dr. Mesut RAZBONYALI
Üye

Prof. Dr. Şaban EREN
Üye

Yrd. Doç. Dr. Cavit TEZCAN
Danışman

ÖZET

Internet teknolojilerinin hızlı gelişimi, bilginin saklanması ve paylaşımı ile ilgili yeni olanakları da beraberinde getirmektedir. Özellikle ticaret, bankacılık gibi sektörler için düşük maliyetli ve coğrafi sınırlara bağımlı olmayan bir ağ yapısı olan Internet büyük bir avantaj haline gelmiştir.

Internet' in sunmuş olduğu hizmetler içerisinde en büyük kullanım oranına sahip olanı World Wide Web (WWW)' dir. WWW' in bu popülerliği, etkileşimli yapısı ve çoklu ortam öğelerini kullanabilme yeteneği sayesinde gerçekleşmektedir.

Web sayfaları üzerinden veri tabanlarına erişim sağlayabilmek de mümkündür. Bu amaçla, derlenen ya da yorumlanan programlama dilleri ile oluşturulmuş bir CGI ya da buna alternatif olarak geliştirilen teknikler kullanılmaktadır. Bu tekniklerden birisi de Java programlama dili için geliştirilmiş olan Servlet API kullanımıdır.

Bu tezde, Visual Basic 5.0 ve Java programlama dilleri kullanılarak birer Web tabanlı veri tabanı uygulaması oluşturulmuştur. Bu uygulamalar yardımı ile CGI ve Servlet API' sinin avantaj ve sınırlılıkları ortaya konmuş ve her iki uygulama kullanılarak bir performans analizi gerçekleştirilmiştir.

SUMMARY

Along with the rapid development of internet technologies, new possibilities of storing and sharing of information are being .ntroduced. Internet free from geographic boundaries provides low cost services especially for the sectors such as banking and commerce.

World Wide Web is the most common and largely used service of internet. The popularity of WWW lies in the fact that it possesses the capacity of using multi-platform components and interactive structure.

It is also possible to have an access to databases through web pages. To serve this purpose, CGI made by compiled and interpreted programming languages or techniques developed as an alternative to CGI are used. On of these techniques is the use of Servlet API which was developed for Java programming language.

In this thesis, a web based database application has been created by using Visual Basic 5.0 and Java programming language. The limitations and advantages of CGI and Servlet have been demonstrated with the help of this application and a performance analysis has been done by using both applications.

İÇİNDEKİLER

GİRİŞ

World Wide Web tabanlı bir veri tabanı uygulaması geliştirmek, yazılım geliştiriciler ve kullanıcılar açısından önemli bir alternatif olarak Internet teknolojileri içindeki yerini almıştır.

Düşük maliyetli olmaları, değişik platformlar üzerinde çalışabilme yetenekleri ve coğrafi olarak çok geniş bir alanda kullanılabilme imkanları, Web tabanlı uygulamaların en önemli avantajları arasındadır.

Web tabanlı bir veri tabanı uygulamasında veri girişi sağlanabilmesi amacıyla standart HTML form öğeleri ile oluşturulmuş bir GUI (Graphical User Interface – Grafiksel Kullanıcı Arabirimi) kullanılmaktadır. Bu arabirim yardımı ile istemci tarafından alınan veri, Web sunucusu üzerinde çalışmakta olan harici bir veri tabanı uygulamasına istek olarak aktarılmaktadır. Bir sonraki adım, veri tabanı uygulamasının istemci tarafından gelen bu istek doğrultusunda işlemi gerçekleştirmesi ve sonuçların Web sunucusu aracılığı ile yeniden istemcinin tarayıcı programına aktarılmasıdır.

Sunucu üzerinde yer alan bir veri tabanına erişim, derlenen ya da yorumlanan programlama dilleri yardımı ile oluşturulan uygulamalar yardımı ile gerçekleşmektedir. Kullanılacak her dilin kendi yapısından kaynaklanan avantaj ve sınırlılıkları bulunmaktadır. Bunun yanında her bir programlama dilinin, verileri saklama ve kullanma teknikleri, değişik platformlarda çalışabilme özellikleri gibi kriterler de göz önüne alındığında, Web tabanlı veri tabanları için değişik erişim yöntemlerin ortaya çıktığı görülmektedir.

Bu yöntemler içerisinde, kullanımı en yaygın olanlarında birisi CGI (Common Gateway Interface – Ortak Ağgeçit Arayüz) programlarıdır. CGI programları, derlenebilen ya da yorumlanabilen diller yardımı ile meydana getirilmektedir. Diğer bir

Web veri tabanı erişim yöntemi ise, yorumlanan bir programlama dili olan Java ile Servlet API kullanılarak oluşturulacak uygulamalardır.

Yapılan ön araştırmalarda Web veri tabanı uygulaması oluşturmada kullanılacak yöntem seçiminin performans açısından önemli bir faktör olacağı görülmüştür. Java' nın yüksek performanslı ve güvenilir bir dil olduğu iddialarına karşılık; yorumlanan bir dil olması sebebiyle C++, Visual Basic gibi derlenen dillere oranla daha yavaş çalışacağı savı ortaya sürülmektedir. Bunun yanında derlenebilen dillerde run-time (çalışma zamanı) problemleri ortaya çıkacağı, böylelikle de performansın düşeceği söylenmekte, bu durumun da ancak yorumlanan bir dil kullanımı ile ortadan kalkacağı düşünülmektedir.

Bu çalışmanın temel hedefi, derlenen bir programlama dili olan Visual Basic 5.0 kullanılarak oluşturulmuş bir CGI' ın ve Servlet API' si ile meydana getirilmiş bir Java programının birer Web veri tabanı uygulaması olarak teknik açıdan karşılaştırılmasıdır. Bunun yanında bu iki yöntemin Internet ortamında uygulanması sonucu ortaya çıkması muhtemel performans farklılıklarının tespit edilmesi de tezin bir başka hedefi olarak belirlenmiştir.

1 WORLD WIDE WEB VE VERİ TABANI KAVRAMLARINA GİRİŞ

1.1 Internet' in Tanımı ve Tarihçesi

Dünya üzerinde mevcut birçok bilgisayar ağının TCP/IP adı verilen ortak bir protokol çerçevesinde birbirleriyle iletişim kurmalarını ve birbirlerinin kaynaklarını paylaşabilmelerini sağlayan ağ sistemine Internet adı verilmektedir.

A.B.D. Savunma Bakanlığı' na bağlı olarak çalışan ARPA (Advanced Research Projects Agency) adında bir kurum 1969 yılında bir bilgisayar ağı kurmuştur. Bu ağın kurulmasındaki amaç, herhangi bir nükleer savaş sırasında hiç kesintiye uğramayacak bir iletişim mekanizması geliştirmekti.

1973 yılında bu kurum DARPA (Defense Advanced Research Projects Agency) adını alarak "Internetting Project" adlı bir proje başlatmıştır. Bu projenin temel hedefi ise mevcut bilgisayar ağlarının birbirine nasıl bağlanabileceğini araştırmaktı. Böylece çeşitli ağlar tarafından kullanılmakta olan farklı yöntemler bir potada eritilebilecek ve ortak bir iletişim yöntemi geliştirilecekti. Böylece Gateway adı verilen geçitler yardımıyla ağlar birbirine bilgi aktarabileceklerdi. (Akın, 1996)

1970' li yılların başlarında, Stanford Üniversitesi çoklu paket değişim teknolojisini kullanarak bir araştırma yapmakla görevlendirilmişti. Bu mekanizma, ağ bağlantılarının çöktüğü ya da güvenilir olmadığı durumlarda iletişim güvenilirliğini arttırmaktaydı. Sonraki araştırma ve fizibilite deneyleri, Transmission Control Protocol/Internet Protocol olarak bilinen TCP/IP' nin gelişimine yol açmıştır. TCP/IP daha sonra, 1983' de bir iletişim standardı olarak kabul edilmiş ve Berkeley' de ki California Üniversitesi' nde Unix' in bir BSD versiyonuna eklenmiştir. BSD, Unix, ARPANET' e eklenerek birçok bilgisayar ve bilgisayar ağının birlikte çalışmasını sağlayan bir sistemdi.

1985' de Ulusal Bilim Kurumu (NSF) programını ortaya koymuştur. NSF' nin süper bilgisayar uygulamalarına olan ilgisi araştırmacıları NSF süper bilgisayar merkezlerine bağlayan yüksek hız iletişim gereksinimlerini doğurmuştur. Bu amaçla ARPANET' i kullanamamaktan doğan durumla birlikte NSF, MCI, IBM ve Michigan Üniversitesi yardımı ile temel bir alt yapı oluşturmuştur. Bu alt yapı belli sayıda bölgesel ağları içermekteydi.

1989' da ARPA, yeni adıyla DARPA, ARPANET' e yeni bir yaklaşım sağlamış ve NSFNET, bugün Internet olarak bilinen yerel ve bölgesel TCP/IP ağlarının toplamından oluşan bir belkemiği yaratmıştır. (Swank ve Kittel, 1996)

1.2 World Wide Web

1.2.1 World Wide Web Nedir?

Herşeyden önce açıkça belirtilmelidir ki WWW(World Wide Web), Internet değildir. Bununla birlikte her iki kavram da birbirine çok yakındır. Internet, kelimenin tam anlamıyla bir ağdır. WWW ise daha çok iletişim uygulama sistemlerinin ve yazılım sistemlerinin aşağıdaki özelliklerle birlikte dağılmış bir dizisidir. World Wide Web' in önemli özellikleri şunlardır:

- Sunucular ve kullanıcılar üzerine kurulu bir sistemdir.
- Tipik olarak TCP/IP ağ protokolünü kullanır.
- HTML dilini anlayabilir.
- Kullanıcı sunucu modelini izler. Bu da çift yönlü veri sistemleri bilgi toplama ve kaynak içeren sunucuların takip edilmesidir.
- Kullanıcılara, http, ftp, telnet ve gopher gibi protokolleri kullanarak sunuculara giriş izni verir.

- URL ler aracılığıyla doküman ve kaynakların incelenmesine yardımcı olur.
- Ses, video gibi çeşitli çoklu ortam öğelerinin kullanımını sağlayabilir.

Tarihsel olarak bakıldığında bugün web i oluşturan fikirlerin İsviçre' de CERN-Pratik Fizik Labaratuvarı' nda çalışmakta olan Tim Berners-Lee tarafından ortaya atıldığını görmekteyiz. 1989 yılında Lee, aşağıda özellikleri verilecek hypertext sistemlerini içeren bir teklif sunmuştur. Bu özellikler şunlardır:

- Tüm platformlar içerisinde aynı kalabilen bir kullanıcı arabirimi olmalıdır. Bu arabirim çeşitli bilgisayarlar üzerinde farklı bilgi kaynaklarına ulaşabilmeyi sağlamaktadır.
- Bilgiye dünya bazında ulaşabilme imkanı sağlamalıdır. Ağ üzerinde ki herhangi bir kullanıcı, herhangi bir bilgiye ulaşabilmelidir. Kullanıcı arabirimi, çeşitli protokol ve türdeki dokümanlara girebilme imkanı sağlamalıdır. (Swank ve Kittel, 1996)

1.3 Web Uygulamaları geliştirmenin Avantajları

Web tabanlı teknolojileri kullanarak uygulama geliştirmek, son kullanıcı uygulamaları ve geleneksel bilgi sistemleri açısından birçok avantaja sahiptir. Bu avantajlar kullanıcı ve geliştiriciler açısından aşağıdaki bölümde ele alınmaktadır.

1.3.1 Kullanıcı Açısından Avantajları

Uygulama geliştirmenin ilk amacı son kullanıcıya ulaşmaktır. Örneğin, Intranet kullanıcıları, organizasyon bilgilerine; Internet kullanıcıları ise veritabanlarına kullanım hakları ölçüsünde kolayca erişebilmek istemektedirler. Web uygulamaları son kullanıcılar için birçok avantaj sunmaktadır:

- Grafiksel kullanıcı arabirimi
- Soyut uygulamalar ve formlarda sorgulama dillerinin kullanımı
- Tarayıcıların kolayca konfigüre edilebilir özellikleri
- Uygulama, veritabanı ve bilgiye çabuk ve kolay ulaşım

1.3.2 Uygulama Geliştiriciler Açısından Avantajları

Web teknolojileri, geleneksel araçların aksine uygulama geliştiriciler açısından da birtakım avantajlar sunmaktadır:

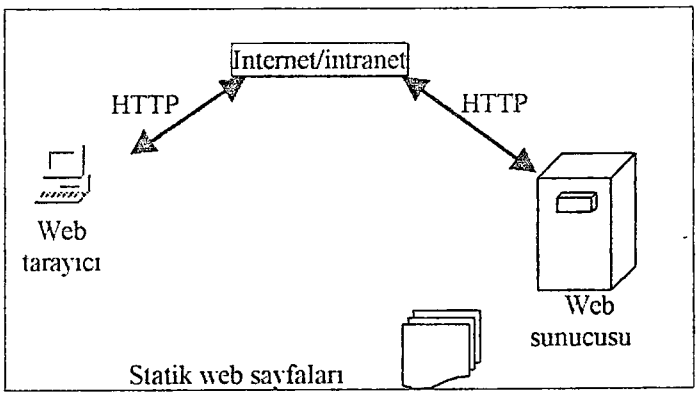
- Standart teknolojilerin kullanımı
- HTML' nin kolay öğrenilebilmesi
- PC, Unix gibi geçiş platformlarına uyum
- CGI programlarıyla uyum
- Hızlı kullanıcı arayüzü geliştirme

1.4 Web Uygulamalarının Sınıflandırılması

1.4.1 Statik Web Uygulamaları

Statik bir web dokümanının yayımlanması sadece bir web tarayıcısı, Internet ya da intranet ve bir web sunucusu gerektirmektedir. Web sunucusu, istemci web tarayıcısından bir istek almaktadır. Sunucu, bu istekleri işlemekte ve cevabı tarayıcıya HTTP (HyperText Transfer Protocol) protokolünü kullanarak geri göndermektedir. Bu

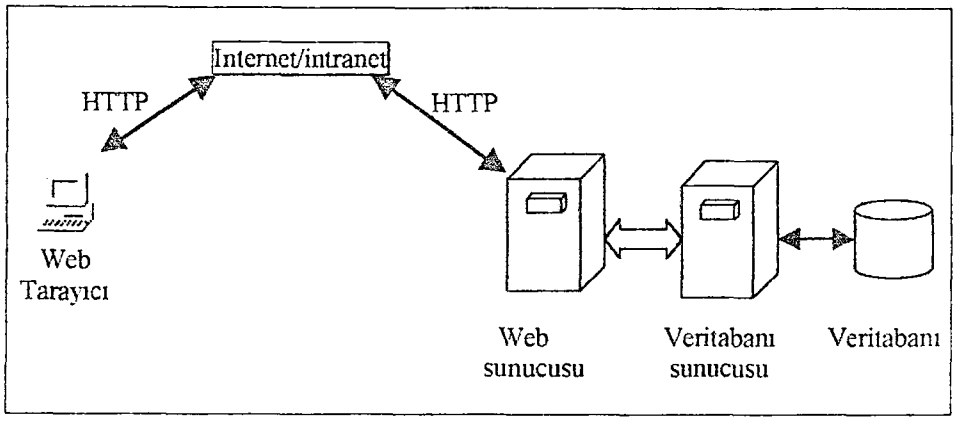
tip bir uygulama, grafikler ve çoklu ortam öğelerinden meydana gelmektedir. (Swank ve Kittel, 1996)



Şekil 1-1: Statik web uygulaması mimarisi

1.4.2 Dinamik Web Uygulamaları

Dinamik bir web uygulaması, istemci ve sunucu arasında etkileşimi sağlayacak formlar ve sorgulama amaçlı bir veritabanı bağlantısı içermektedir. Bu tip bir uygulama statik uygulamalara ek olarak bir veritabanı sunucusu içermektedirler. İstemciye ait tarayıcı yine HTTP protokolünü kullanarak sunucuya bağlanır. Bununla birlikte sunucu, veritabanı sunucusu ile farklı bir protokol ile iletişim kurmaktadır. (Swank ve Kittel, 1996)

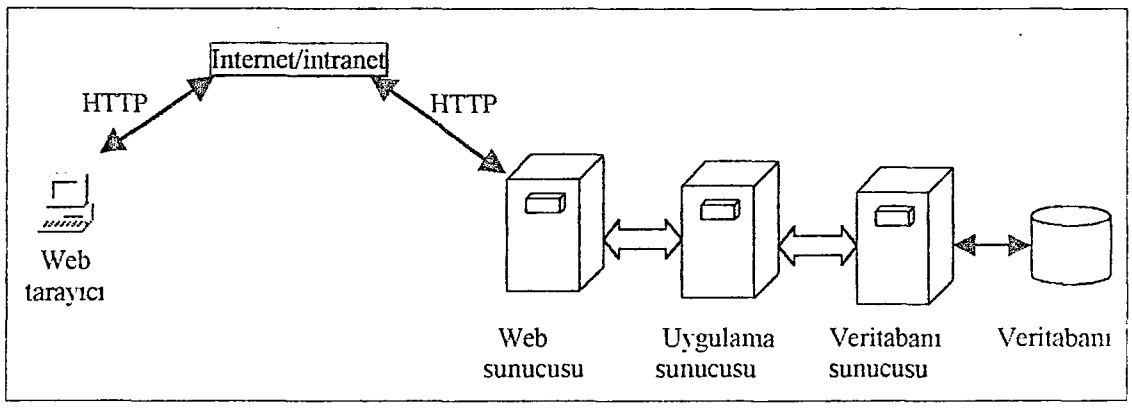


Şekil 1-2: Dinamik web uygulaması mimarisi

1.4.3 Kompleks Web Veritabanı Uygulamaları

Web veritabanı uygulamaları, uygulama sunucusu adı verilen bir ek bileşene ihtiyaç duymaktadırlar. Uygulama sunucusu katmanı, geniş ve karmaşık web veritabanı uygulamalarının ihtiyaç duyduğu yüksek hızlı ve ölçeklendirilebilir bir mimari sağlamaktadır.

Uygulama sunucusu, uygulama işlem mantığının çoğunu yerine getirmekte ve gerekli olduğunda veritabanına bağlanmaktadır. Buna ek olarak, uygulama sunucusu, durum idaresi ve oturum kontrol mekanizmasını da sağlamaktadır. Bunlar, etkileşimli işlemler için gerekli olmaktadır. Örneğin, bir kullanıcı tek bir işlem için birden fazla web sayfasına girmek durumunda ise uygulama sunucusu bilgiyi bir web sayfasından diğerine aktarılacak şekilde kurabilmektedir. (Fournier, 1999)



Şekil 1-3: Kompleks web uygulaması mimarisi

1.5 Web Tabanlı Veri tabanı Sistemleri

1.5.1 Veri tabanının tanımı ve türleri

Genel olarak veritabanları, çok büyük miktarda bilgiyi depolayabilen, düzenleyebilen ve istendiğinde bu bilgileri sunabilen yazılımlardır. Veritabanları, verileri saklama ve kullanıcıya sunma biçimleri açısından farklı yapılarda olabilirler. Yaygın olarak kullanılan veritabanı tipleri şunlardır.

- **İlişkisel (Relatioanal) Veritabanları:** En yaygın olarak kullanılan tiptir. Microsoft Access gibi pek çok PC tabanlı veritabanı yazılımı, ilişkisel veritabanı türünü kullanırlar. Ayrıca Oracle ve Sybase gibi güçlü yazılımlarda bu türü kullanırlar. Bu yazılımlar daha çok networkler için tasarlanmışlardır. Farklı sistemlerde çalışabilen paketler olarak sunulmaktadırlar. Bu yüzden farklı ortamlarda çalışan organizasyonlar tarafından tercih edilirler.

- **Hiyerarşik (Hierarchical) Veritabanları:** Bu tip veritabanları, ana bilgisayar ortamlarında çalışan yazılımlar tarafından kullanılmaktadır. Örnek olarak IBM' in

IMS' i verilebilir. Uzun bir geçmişi olmasına rağmen, PC ortamına uyarlanan hiyerarşik veritabanları bulunmamaktadır. Hiyerarşik veritabanları, bilgileri bir ağaç yapısı şeklinde saklamaktadırlar. Bir kök ve bu köke bağlı dallar veritabanının yapısını oluşturmaktadır.

- **Nesneye Yönelik (Object Oriented) Veritabanları:** 90' ların en popüler kavramlarından olan "nesneye yönelik programlama" içerisinde yer alan bu veritabanı türü henüz tam bir kullanım alanına sahip değildir. Ancak ilişkisel veritabanları ile karşılaştırıldığında bazı avantajları olduğu görülmektedir. Örnek olarak arama işlemlerindeki hızı gösterilebilir.

Veritabanları. Web üzerinde kullanılan sistemler dahil birçok bilgi sisteminin kalbidir. Bir veritabanı sistemi kullanılmak isteniyorsa, düşünülmesi gereken temel noktalar, kullanılacak veritabanının tipi ve bu veritabanından verileri saklama ve okuma yöntemleri olmalıdır.

İlişkisel veritabanları, öznitelikler arasındaki ilişkileri kullanarak verileri indeksler ve getirirler. Geniş ölçekli bir web yayımcılık ortamında dahil edilebilecek ilişkisel veritabanı olanağını ilk sunanlar, Oracle, Sybase ve Informix' tir. İlişkisel veritabanlarının en önemli özelliği, verileri başka bir veritabanına aktarmayı sağlayan SQL dilini kullanmasıdır.

Nesne tabanlı veritabanları. nesneler arasındaki ilişkileri kullanarak verileri indeksler ve getirirler. Şu anda birçok nesne tabanlı veritabanı, uygun doküman yayımlama ortamlarının bir parçasıdır. Bu veritabanı sistemleri, web yayımcılığında kullanılmak üzere geliştirilebilir. Nesne tabanlı veritabanları, güçlü özellikleri ile büyük bir potansiyel olarak sunulurlar. Fakat verileri bir sistemden farklı diğer bir sisteme aktaracak yapıdan yoksundurlar.

Veritabanı sistemleri, bir veri elemanına, genellikle *öznitelik* olarak adlandırılan, özel bilgileri atama avantajı sağlar. Öznitelikler, kullanıcının yayımcılık ortamını daha iyi yönetmesini sağlamaktadırlar. Bir veri elemanının özniteliği, müşterinin posta kodu, adresi veya ürün başvurusu gibi yayımcılık ortamına yararlı herhangi bir başvuru olabilmektedir. Bu öznitelik aynı zamanda, web yayını olan ve rehberin yazıldığı dilin formatına, yazarına, düzeltme numarasına, kaba bir görünüm mü, yoksa son görünüm mü olduğuna ilişkin başvurular içeren özniteliklere sahip başvuru rehberi gibi büyük bir dokümanın anahtar elemanlarına başvuru olabilir. Doküman ve onun kişisel öğeleri veya organizasyon ve yapısı hakkındaki öznitelik bilgilerinin koleksiyonu, veritabanı tarafından ortaya sürülen bir yayımcılık sisteminin *veri modelini* oluşturur. (Stanek, 1997)

1.6 WEB Veritabanları ve Kullanım Amaçları

Herhangi bir veritabanı yönetim sistemi gibi web veritabanı da bir kaynak deposudur ki buna herhangi bir API kullanan dillerle girilebilir. Normal veritabanı sistemlerinin tersine, web veritabanları tamamen kolay bir şekilde ulaşıma sahip değildir ve arabirimler ve komut satırlarıyla giriş sağlanamamaktadır. Yerel biçimde dizayn edilmiş platformlar da kullanılmamaktadır.

Web veritabanları diğer web uygulamaları yardımı ile giriş yapılabilen veritabanlarıdır ve özellikle de standart html leri kullanarak oluşturulmuş formları içermektedir. HTML deki bazı kolaylıkları kullanarak web server üzerindeki uygulama programlarına sunucu içerikli CGI programları yardımı ile giriş yapılabilir. Html form arabirimleri, kullanıcıları veritabanlarının fonksiyonelliği ile birleştirir aynı zamanda organizasyonel veri depolarına girişi sağlayabilmektedir. Yine veritabanında araştırma yapmak ve spesifik bilgiye erişim açısından birtakım uygulamalar yaratılabilir. Örneğin, önceki yıllarda organizasyon içerisinde çok iyi iş yapan pazarlamacınızın profilini

hazırlayabilirisiniz. Bu uygulama aynı zamanda içeriği geniş veritabanlarını destekleyecek platformlardan bilgi çekmenizi de sağlayabilecektir. Örneğin, satış istatistikleri belli bir veritabanından çekilebilir ve çeşitli satışların ve promosyonların gelirleri nasıl etkilediği yönündeki istatistiksel analizler yapabilir. Web tarayıcısını kullanan kullanıcılar tarafından bu tür veritabanlarına girebilme olasılığı sağlandığı sürece bu da bir web veritabanını oluşturmuş olacaktır.

Birçok organizasyon çeşitli özel veritabanları kullanmaktır ki bunlarda temel altyapı ihtiyaçlarını karşılamaktadır. Bunlara örnek olarak da insan kaynakları ve temel bilgi sistemleri verilebilir. Organizasyonların bu veritabanlarını web uygulamalarında kullanmaları açısından bazı sebepler öne çıkmaktadır. Aslında birçok web tabanlı uygulamalarda bu veritabanları bilgi sistemleri sunmada temel blokları inşa açısından kullanılabilir. (Swank ve Kittel, 1996)

1.7 Performans Kriterleri

Bu bölümde, web uygulamalarında eğer önceden göz önünde bulundurulmazsa performansı etkileyebilecek bazı konular incelenecektir. Bununla birlikte web sayfasının tasarımında dikkat edilmeyen bazı noktalar performansı olumsuz yönde etkileyebilmektedir. Aynı zamanda estetik ve kullanılabilirlik konuları bu bölümde ele alınmaktadır. (Fournier, 1999)

Tablo 1-1: Web sayfası dizaynı kusurları

Kategori	Kullanıcıların Ortak Şikayetleri
Organizasyon (Organization)	• Web sitesi ya da uygulamasının içeriğini gösteren herhangi bir sayfa bulunmaması. • Web sitesi ya da uygulamasının yapısının çok düzensiz ya da karmaşık olması. • Bir sayfadan diğerine geçişte herhangi bir tutarlılık gözlenmemesi.

Yanıtlama zamanı (Response Time)	• Kullanıcılar sayfaların yüklenmesi için çok uzun bir süre beklemektedirler. Çünkü, bazı sayfalar çok fazla çoklu ortam öğesi içerebilmektedir.
Dolaşım (Navigation)	• Kullanıcıların web sitesinde sörf yapmaları konusunda onlara yardımcı olacak herhangi bir uygulama geliştirilmemiştir. • Kullanıcıların ekranda sürekli olarak aşağı-yukarı bütün bilgileri görebilmek için gezmeleridir. Bu da sayfaların çok büyük olmasından kaynaklanmaktadır. • "Dead End" şeklinde adlandırılan çıkmaz sayfalar kullanıcıların sürekli geriye doğru giderek sayfaları takip etmelerine sebep olmaktadır. • Kopuk linkler, web siteleri ve sayfalar arasında kopukluklar oluşturmaktadırlar.
Okunaklılık (Legibility)	• Web tarayıcısına ait pencere bir çok framelere bölünmüştür. Buradaki grafik arayüzler karışıklıklara sebep olmaktadır ve kullanıcıları farklı yerlere yönlendirebilmektedir. • Fontlar okunamayacak kadar küçüktür. Bu nedenle ekrandan çok uzak gözükmektedirler. • Web sayfalarının metin içeriklerini okumak çok daha zordur. Bu da arka planın kontrast ayarından kaynaklanmaktır.
Estetik (Aesthetics)	• Kullanıcılar sürekli yanıp sönen metinlerden sıkılmaktadırlar. • Metin ve grafikler birbirlerine iyi entegre olmamışlardır. • Web sayfaları çok fazla görsel öğe ile doludur.

1.7.1 Cevap Süresi

Bir web sayfasının kullanıcı tarafından incelenmesi biraz süre gerektirmektedir. Bu zamanlama seçilen web tarayıcısının türüne ve versiyonuna göre değişmektedir. Belli web uygulamaları spesifik bazı tarayıcılarla çalışırken, optimum bir süre ortaya koymaktadır.

Aynı zamanda bir Internet tabanlı uygulamanın ortaya konması kararı, ethernet ağları tarafından desteklenen iç kullanıcılar için 10 Mbits/sec veya 100 Mbits/sec gibi bir veri aktarımını ortaya koyarlar ki bu da web sayfalarının dizaynı ve içeriğini etkilemektedir. Böyle bir bant genişliği kapasitesi ile birçok grafiksel öğenin kullanımı kolaylaşmaktadır. Bunlar web işlemlerini engellemeyecek bir hızda kullanılabilir.

Öte yandan, eğer web uygulamasına karşıdaki bir kullanıcı tarafından modem aracılığı ile giriliyorsa grafiklerle ilgili en küçük bir bozulma bile web sayfasını görüntülemek için geçecek zamanı kötü yönde etkileyebilmektedir. Bu durumda mevcut bant genişliğini değerlendirmek gerekmektedir. Sadece ofislerdeki dahili kullanıcılar açısından düşünmemek gerekmektedir. Burada amaç tüm kullanıcılara ulaşmak olmalıdır.

Grafiklerin yanında videolar, ses efektleri ve resimlerle dolu, büyük web sayfalarını kullanmak için harcanan zaman bazı sorunlar yaratabilmektedir.

Grafik ara yüzünün gösterilmesinden farklı diğer faktörler de web veritabanı uygulaması işleminin cevap süresini geciktirebilmektedir. Örneğin, bu faktörlerden birisi de bilgiyi cevaplama süresidir. Bilgi cevaplama süresi, uygulamanın veritabanı sunucusuna girişi ve istenen bilginin kullanıcı tarafından çekilmesini içermektedir. Buna rağmen basit bir modele sokulabilirse basit formlar kullanmak bu süreyi kısaltacaktır. Çok daha uygun arayüzler web veritabanı işlemleri için daha etkili olabilmektedir. (Fournier, 1999)

1.7.2 Okunabilirlik

Web veritabanı uygulaması geliştiriciler, çok süslü web sayfaları yapma konusundaki iştahlılıklarını biraz olsun engellemek zorundadırlar. Birçok geleneksel istemci-sunucu uygulaması, çok farklı fon çeşitlerinin kullanılması, bütün bilginin web

sitesi kullanıcı ara yüzü kullanılarak ortaya konan tüm bilgilerin okunmasını azaltacaktır. Sayfa içerisinde yanıp sönen fonlar belki hoş gözükebilir ama kullanıcı onları bir seferliğine alacaktır. Ama aynı efekt, kullanıcının bu sayfaya her girdiğinde daha da sıkıcı ve rahatsız edici bir his verebilmektedir. Aynı zamanda terminoloji de basit olmalı ve kullanıcıların aişna olduğu bir şekilde kullanılmalıdır. (Fournier, 1999)

1.7.3 Estetiklik

Web veritabanlarının estetikliğine bakıldığında değerlendirmesi çok zor bir durum ortaya çıkmaktadır. Çünkü bu yorum bireysellik içermektedir. Web uygulamasının arayüzünün kullanıcıyı ne derece tatmin edebileceğini bilmek zordur. Ama buna rağmen bazı statik unsurlar çok abartılmadan kullanılabilir.

Örneğin web arayüzü sürekli tekrarlanan bir sayfadan diğerine geçişi yavaşlatan unsurlar içeriyor mu? Sürekli ortaya konulan görsel elemanlar diğer sayfalarda da kendisini tekrar ediyor mu? Renk kullanılmışsa bunlar birbirleri ile uyumlumu? şeklinde değerlendirme yapılabilmektedir. (Fournier, 1999)

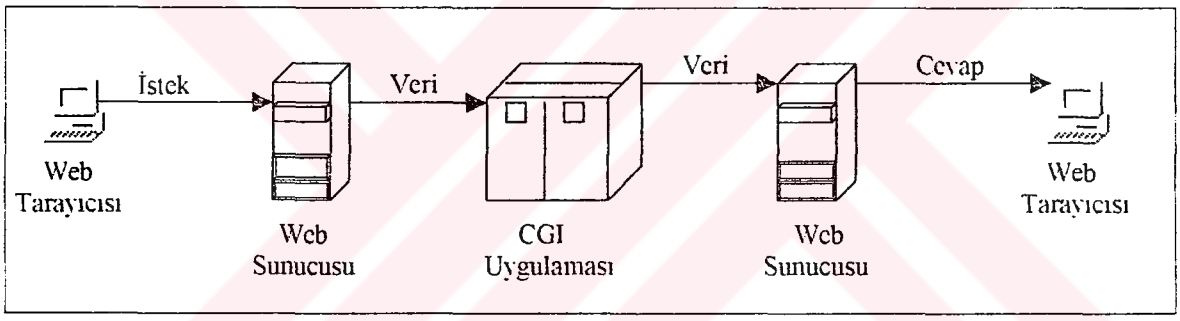
1.7.4 Kullanılabilirlik

Kullanılabilirlik daha önce de anlatıldığı gibi istemci-sunucu web uygulamaları açısından önem içermektedir. Farklı kullanıcı senaryoları, web veritabanı uygulamalarında, özellikle ticari işlemlerde farklı kullanımlar gösterebilmektedir. Herşeyden önce kullanıcılar web grafik arayüzlerini test etmelidirler ve geliştiriciler de bunların uygulanabilirliğini tartışmalıdırlar. (Fournier, 1999)

2 CGI (COMMON GATEWAY INTERFACE)

2.1 CGI' ın Tanımı ve Çalışma Prensibi

En genel tanımı ile CGI, Web sunucuları ile iletişimi sağlayan dış kaynaklı programlara verilen bir isimdir. CGI iki şeyi yapabilmektedir: web tarayıcısından, web sunucusuna gönderilecek olan bilgiyi toplamak ve isteklerden dış bir programda geçerli olabilecek bilgiyi meydana getirmektedir. Böylece CGI web tarayıcısına program çıktısını gönderebilmektedir. Aşağıdaki şekilde CGI' ın çalışma şekli gösterilmektedir: (Colburn, 1998)



Şekil 2-1: CGI' ın çalışma prensibi

CGI programları kullanılarak okuyucu ile gerçek bir etkileşim içerisinde profesyonel web yayıncılığı yapılabilmektedir. CGI programları sayesinde, okuyucunun yapacağı girişler bir indekse veri olarak işlenebilmekte, veritabanlarında sorgulamalar yapılabilmekte ya da etkileşimli dokümanlar yaratılabilinmektedir. Program işlerken arka planda gerçekleşen olaylar şu şekildedir: (Stanek, 1997)

1. İstemci girişi sunucuya geçirir.
2. Sunucu girişe uygun olacak şekilde ortam değişkenlerini ayarlar.

3. Sunucu, girişi CGI programcığının adı ile bir değişken gibi geçirir.
4. Sunucu, eğer varsa, komut satırı girişini ya da standart giriş katarını CGI programına geçirir.
5. Programcık girişi işler.
6. Programcık, çıkışı sunucuya döndürür. Bu çıkış daima belli bir başlığı taşımaktadır. Eğer ek bilgi varsa bir gövdesi bulunmaktadır.
7. Sunucu, ortam değişkenlerini çıkışa uygun olarak hazırlar.
8. Sunucu, çıkışı istemciye geçirir.

2.2 CGI' ın Avantajları

2.2.1 Platform bağımsızlığı

CGI birçok web sunucusu için oluşturulabilmektedir. Bunların hangi platformda çalışacağı bir sorun teşkil etmemektedir. Windows NT, Apache, Netscape ve Microsoft IIS gibi en popüler web sunucular CGI' ı desteklemektedir.

Bunun yanında birçok platformu desteklemesi CGI programlarının bir ortam için yazılıp, diğer ortamlar için de aynen kullanılabileceği anlamına gelmemektedir. Örneğin, sunucu taraflı Java Appletleri ve bazı Perl scriptleri ufak değişiklikler ile ya da değişikliğe uğramaksızın, C programları da tekrar derlenerek farklı platformlarda kullanılabilmektedir. (Colburn, 1998)

2.2.2 Dil bağımsızlığı

CGI programları birçok dil ile yazılabilmektedir. Bu da geliştiricilerin en rahat kullanabildikleri dil ile CGI programı yazabilecekleri anlamına gelmektedir. Bu konuda Perl, C, C++, Visual Basic dilleri oldukça yagın olarak kullanılmaktadır.

2.2.3 Ölçeklendirilebilme

CGI arayüzünün sadeliği, son derece ölçeklendirilebilir olması anlamına gelmektedir. Bir CGI programı, veritabanı Oracle gibi veritabanlarına bağlantılar yaratmak ya da bir anket içeriğini e-mail adreslerine göndermek gibi birçok işlemde kullanılabilinmektedir.

2.3 CGI Programlarının Yaratılması

2.3.1 Programlama Dilinin Seçimi

CGI genel olarak birçok programlama dilini desteklemektedir. Standart girişten veri okuyabilen ve standart çıkışa veri yazabilen her dil CGI programı yazmak amacıyla kullanılabilmektedir. CGI programları Perl gibi yorumlanan diller ile yazılabildiği gibi, C gibi derlenen diller ile de oluşturulabilir. (Colburn, 1998)

Programcıkları yazmak için seçilecek dil, web sunucusunun ihtiyaçlarına cevap verebilecek ve bu sunucu üzerinde kullanılabilecek bir dil olmalıdır. En çok kullanılan diller şunlardır: (Stanek, 1997)

- Bourne Shell
- C Shell
- C/C++
- Perl
- Python
- Tel
- Visual Basic
- Java Script
- VB Script

Yukarıda da belirtildiği gibi bir CGI programcığı herhangi bir dil ile yazılabilmektedir. Fakat bu dil aşağıdaki gereksinimleri karşılamalıdır:

- Kullanılan dil sunucu üzerinde standart girdi ve çıktıya imkan sağlayabilecek mekanizmaya sahip olmalıdır. Bu durum, kullanıcı girdisinin HTML formlarından elde edilmesini sağlamaktadır. Buna ek olarak standart çıktıların yazılmasını, alt formların geliştirilmesini, veritabanı sorgu sonuçlarının elde edilmesini ve bunların istemciye geri dönüşünü sağlayabilmelidir.
- Kullanılan dil, CGI programlarının çevre değişkenlerine giriş sağlayabilmelidir. Bu da yine HTML formlarından kullanıcı girdisinin elde edilmesi ile sağlanmaktadır. Buna ek olarak, CGI uygulamalarının kullanıcının web tarayıcısının cinsi, yol bilgisi, IP adresleri gibi çeşitli bilgilere erişebilmesine olanak sağlanmalıdır. Son olarak, çevre değişkenleri kullanıcı girdilerinin bağlantılarının oluşumunu sağlayacaktır.
- Kullanılan dil birtakım mekanizmalar ortaya koymalıdır ya da veritabanı sistemlerine ulaşım gibi uygulamalara bağlantı için arabirim olabilecek araçlar sunmalıdır. (Swank ve Kittel, 1996)

2.3.2 Derlenen Dillere Karşı Yorumlanan Diller

Visual Basic, C/C++ gibi derlenen diller ve TCL, Java Script gibi yorumlanan diller programlama dillerinin iki temel kategorisidir. Derlenen dillerde ilk adım, derleyicinin kaynak kodu makine diline çevirmesidir. İkinci adım da ise program işletilmekte ve çıktı meydana getirilmektedir. Yorumlayıcı dillerde ise derleme ve yorumlama işlemleri aynı anda gerçekleştirilmektedir.

Derlenen ve yorumlanan diller arasındaki önemli bir fark, derlenen dillerin derleme ve yorumlama işlemlerini farklı zamanlarda gerçekleştirmesidir. Program yazıldıktan sonra derleme işlemine tabi tutulur, daha sonra yorumlanır. Yorumlanan dillerde ise bu iki işlem aynı anda gerçekleştirilmektedir.

Bazı derlenen diller arasında da bu yönden farklılıklar göze çarpmaktadır. Örneğin C dili programı bir kez derlemekte ve herhangi bir değişiklik yapılmadığı sürece programı bu şekilde çalıştırmaktadır. Perl' de ise program her çalıştırıldığında yeniden derlenmektedir.

Bu tür dillerin en önemli avantajı hızlı olmalarıdır. Çünkü derlenen diller ikili sisteme göre depolanmaktadır. Diğer bir önemli avantaj ise daha güçlü bir güvenirliğe sahip olmalarıdır. Böylece program makine dili kodlarına dönüştürülmekte ve bunları açmak ve içlerinde değişiklik yapmak herhangi bir yolu bulunmamaktadır. Tüm bu avantajların yanında dezavantajlarından birisi hata ayıklamanın zor olması olarak verilebilir.

Bunun yanında Java dilinin ilginç bir performans problemi bulunmaktadır. Java byte code' lar şeklinde JVM (Java Virtual Machine) içerisinde derlenmekte ve işletilmeye başlanmaktadır. Yine çalıştırma işlemi JVM içerisinde gerçekleşmekte ve bu da performansı engellemektedir.

Yorumlanan dillerde ise derleme ve yorumlama işlemlerinin ayrı zamanlarda yapılmasından dolayı performansları daha düşük olmaktadır. Bununla birlikte genel olarak hata ayıklama işlemi daha kolay olmaktadır. (Colburn, 1998)

2.3.3 CGI Programcıklarında Girdi

Kullanıcı bir CGI programcığına ilişkin bir bağlantıyı etkinleştirdiğinde giriş sunucuya gönderilir. Sunucu bu verileri ortam değişkenleri olarak biçimlendirir. Ayrıca

ek bir veri varsa komut satırı ya da standart bir giriş kataı yolu ile gönderilip gönderilmediğini kontrol etmektedir.

2.3.4 Ortam Değişkenleri

CGI programcıklarında giriş ortam değişkenleri şeklinde olmaktadır. Bu değişkenler tarayıcının sunucudan istemiş olduğu bilgi ile ilgilidir. Büyük-küçük harfe duyarlıdırlar. (Stanek, 1997)

Sunucu bu değişkenleri üç değişik kaynaktan oluşturmaktadır:

- Gelen HTTP istek paketi: Bu istekte bulunan bilgisayarın IP (Internet Protocol) adresini sağlamaktadır.
- Web sunucusunun kendisi: Örneğin, SERVER_SOFTWARE değişkeni web sunucusu hakkında bilgi sağlar.
- HTTP istek başlığı web sunucusuna ve istemciden bilgi sağlayan değişkenler: İstek başlığından bilgi alan QUERY_STRING. REQUEST_METHOD gibi ortam değişkenlerine istek başlığından bilgi sağlanabilmektedir.

Aşağıda ortam değişkenlerinin kullanımına dair bazı örnekler görülmektedir:

- GATEWAY _INTERFACE = CGI/1.1
- REMOTE_USER = william
- REMOTE_HOST = www.tvp.com
- QUERY_STRING = /usr/cgi-bin/formparse.pl?sozcuk1+sozcuk2+sozcuk3

Bazı ortam değişkenleri sisteme özgü olmakla birlikte çoğu standarttır. Standart ortam değişkenleri ve işlevleri tablo 2-1' de verilmektedir. (Stanek, 1997)

Tablo 2-1: CGI ortam değişkenleri

Değişken	Açıklama
AUTH_TYPE	Doğrulama yöntemini belirler ve kullanıcı girişlerini geçerli kılmak için kullanılır.
CONTENT_LENGTH	Veri uzunluğunu sayısal bir değer olarak izlemek için kullanılır.
CONTENT_TYPE	MIME tipi verilere işaret eder.
HTTP_ACCEPT	Sunucu tarafından CGI programına geçirilirken inceleyicinin kabul edebileceği MIME tiplerin işaret eder.
HTTP_USER_AGENT	Sunucu tarafından CGI programcığına geçirilirken isteklerin gönderilmesi için kullanılan inceleyicinin tipine işaret eder.
GATEWAY_INTERFACE	Sunucunun kullandığı CGI standartının sürümüne işaret eder.
PATH_INFO	CGI programının tanınmasından sonra URL içerisinde bulunan ek bilgiye işaret eder.
PATH_TRANSLATED	PATH_INFO değişkenine dayanan sürücü tarafından ayarlanan bir değişkendir. Sunucu PATH_INFO değişkenine göre bu değişkeni ayarlar.
QUERY_STRING	URL bir sorgulama taşıyorsa sorgu katarını ayarlar.
REMOTE_ADDR	İstekte bulunan bilgisayarın IP adresini tanımlar.
REMOTE_HOST	İstekte bulunan bilgisayrın adını belirler.
REMOTE_IDENT	İstekte bulunan makineyi belirler.
REMOTE_USER	Kullanıcı adını belirler.
SCRIPT_NAME	Çalıştırılacak programcığın adresini tanımlar.
SERVER_NAME	Sunucunun ev sahibi adını, takma adını ya da IP adresini tanımlar.
SERVER_PORT	İsteklerin kabul edileceği sunucu port numarasını tanımlar.
SERVER_PROTOCOL	Sunucuya gönderilen isteklerin protokolünü tanımlar.
SERVER_SOFTWARE	Web sunucu yazılımını tanımlar.

2.3.5 Standart CGI Girdisi

Web sunucusuna gönderilen çoğu girdi ortam değişkenlerini ayarlamak için kullanılmaktadır. Bir kullanıcı bir CGI programcığı tarafından işlenmek üzere gerçek veriyi gönderdiğinde, bu veri bir URL kodlu arama katarı olarak ya da standart bir giriş katarı yoluyla alınır. Sunucu da gerçek verinin veriyi gönderme yöntemi aracılığı ile nasıl işleneceğini bilmektedir.

Standart bir giriş olarak veri göndermek veri iletiminin en doğrudan yoludur. Kısaca; sunucu, CGI' a standart girişten kaç byte okuyacağını söylemektedir. Programcık standart giriş akımını başlatmakta ve belirtilen miktarda veriyi okumaktadır. Uzun URL kodlu arama katarları kesilip kısaltılmasına rağmen standart giriş akımı üzerinden gönderilen veriler kesilmemektedir. Sonuç olarak, standart giriş akımı veri aktarmak için tercih edilmektedir. (Stanek, 1997)

2.3.6 CGI Programcıklarında Çıktı

Programcık girişi işlemeyi tamamladıktan sonra çıktıyı sunucuya döndürmelidir. Daha sonraki adımda ise sunucu cevabı istemciye gönderecektir. Bu çıktı, boş bir satır ve mesaj gövdesinin takip ettiği başlığı içeren bir HTTP cevabı şeklinde olacaktır. CGI başlık çıktısının formatı tam olarak belli olmamakla birlikte mesaj gövdesinin formatı başlıkta belirtildiği durumlara uygun olarak gerçekleşmektedir. Örneğin mesaj gövdesi istemciye gösterilecek bir HTML dokümanı taşıyabilmektedir.

2.4 CGI Başlıkları

CGI başlıkları sunucuya ilişkin bazı talimatlar içermektedir. Tek bir başlık sunucu talimatlarından birini ya da tümünü taşıyabilmektedir. CGI programcığı bu talimatları sunucuya aktarmaktadır. Normalde, başlığı mesaj gövdesinden ayıran boş bir satır takip eder. Fakat, buradaki çıkışın mesaj gövdesi yoktur. Üç geçerli sunucu talimatı mevcuttur:

1. **Content_Type (İçerik Tipi):** İstemciye geri gönderilecek verinin MIME (Multipurpose Internet Mail Extension) tipini tanımlamaktadır. Genellikle programcığın veri çıktısı bir HTML dokümanı gibi tam formatlı bir

dokümandır. Bu format, başlık içerisinde Content - Type : [tip]/[alt tip] şeklinde belirlenebilmektedir. Sık kullanılan MIME tipleri aşağıdaki tabloda görülmektedir.

Tablo 2-2:Sık kullanılan MIME tipleri

Tip/Alt tip	Tanımı
application/msword	Microsoft Word dokümanı.
audio/basic	Wav formatında ses verisi.
image/gif	Gif formatında görüntü.
image/jpeg	Jpeg formatında görüntü.
multipart/alternative	Alternatif formatlarda veri.
Text/html	HTML formatlı metin.
Text/plain	Düz metin.
Video/mpeg	Mpeg formatında video.
Video/quicktime	QuickTime formatında video.

2. **Location (Konum):** CGI programcığının çıktısı programcık içerisinde yaratılmış olan bir doküman olmak zorunda değildir. Location alanı kullanılarak web üzerinde yer alan herhangi bir dosyaya erişmek mümkündür. Sunucu, dosyanın konumuna göre doğrudan ya da dolaylı olarak konum referanslarını işlemektedir. Dosya yerel olarak bulunabilirse istemciye aktarılmaktadır. Aksi halde sunucu istemcinin URL' sini yeniden yönlendirmekte ve istemci dosyayı almaktadır. Location, programcık içerisinde aşağıdaki şekilde belirlenebilir:

Location: http://www.tvpress.com

3. **Status (Durum):** Bu alan, istemciyi izlemek üzere sunucuya bir durum satırı aktarmak amacıyla kullanılmaktadır. Durum kodları genellikle durumun ne olduğunu açıklayan bir bir katarla ve bunu takip eden üç rakamlı bir kod kullanılarak ifade edilmektedir. Bu rakamlar ve karşılıkları aşağıda listelenmektedir:

1xx	Henüz yerleştirilmedi
2xx	Başarılı
3xx	Yeniden yönlendirme
4xx	İstemci hatası
5xx	Sunucu hatası

2.4.1 Verilerin CGI Programcığına Aktarılması

Bir web sayfasından CGI programcığına veri aktarırken iki tür yöntem kullanılabilmektedir. Bunlar Get ve Post' dur. Bu işlem için öngörülen yöntem Get' dir. Get yöntemi kullanıldığında postalanan veri URL programcığının sonuna eklenmektedir. URL ve veri sunucuya tek bir giriş olarak geçmektedir. CGI programcığı içerisinde, URL SCRIPT_NAME, veri ise QUERY_STRING ortam değişkenleri içerisine yerleştirilmektedir.

Post yöntemi kullanıldığında ise veri sunucu aracılığı ile ayrı bir akım olarak CGI programcığına gönderilmektedir. CONTENT_LENGTH ortam değişkeninin değeri CGI programcığının standart giriş akımından ne kadar veri okuyacağını bildirir. Bu yöntem kullanıldığında aktarılacak veri için bir sınırlama söz konusu olmamaktadır. (Stanek, 1997)

Get ve Post kullanımı konusunda dikkat edilmesi gerekli olan özel durumlar söz konusudur. Örneğin, verinin içeriği telefon numarası, şifre kredi kartı bilgisi gibi özel bilgileri içeriyorsa her zaman Post metodu kullanılmalıdır. Çünkü, Get metodunun veriye karşı duyarlılığı zayıftır. Bu yöntemle gönderilen veri güvenli ve özel olma durumunu kaybetmektedir.

Kullanım sırasında dikkat edilmesi gerekli diğer önemli noktalar ise şöyledir:

- Eğer kullanıcılar CGI script' i tarafından geri döndürülecek bir bookmark oluşturmak istiyorlarsa GET metodu kullanılması gerekecektir. Böylece, bu sayfaya ulaşmak için forma girilen veriler bookmark içerisinde URL' nin bir parçası olarak kaydedilecektir.
- Benzer bir şekilde, form yerine CGI programına girmek için bir link kullanılıyorsa GET metodu en iyi seçim olacaktır.
- Eğer form bir çok alan içeriyorsa POST metodu kullanılmalıdır. Çünkü GET metodu, sunulacak çok fazla form verisi olması durumunda çok uzun URL' ler oluşturacaktır. (Colburn, 1998)

2.5 CGI ' ın Alternatifleri

2.5.1 CGI Alternatiflerinin Ortaya Çıkma Sebepleri

CGI, çok önemli bir gelişme gibi durmasının yanında bir takım kendine has kısıtlamaları da vardır. En önemlilerinden birisi performans konusundadır. Kullanıcı her seferinde CGI scriptini istemekte ve server da bu durumda CGI programını sunmaktadır. Ama bu CGI programının Perl gibi yorumlanmış bir dille yazılması durumunda program bütün Perl yorumlayıcısını çalıştırmakta ve programı bu şekilde işletmektedir. Bu da bir miktar vakit almaktadır. Bu her Web sitesi için geçerli bir durum değildir. Çünkü, bunların performans probleminden etkilenecek kadar kullanıcıları yoktur. Ama çok kullanıcılı ve karmaşık uygulamalar gerektiren sitelerde CGI' ın kısıtlamalarını aşmak çok zor olmaktadır.

Bir kullanıcının her seferinde CGI scriptinden bir talebi olduğunda server script i çalıştırır ve çıktısını elinde tutar. Eğer CGI script i veritabanından veri toplamaya göre dizayn edilmişse scriptin her çalışması durumunda veritabanına bağlanması ve girmesi

gerekmektedir. Bu da Web sitesinin performansı açısından kötü bir durum oluşturmakta ve veritabanını da zayıflatmaktadır.

CGI ile ilgili diğer bir sınırlandırma ise, statik Html sayfaları yazarken ve CGI scriptleri kullanarak dinamik sayfalar oluşturma durumunda çok iyi bir araç olmamasıdır. Server tarafının içeriği de Web sayfalarının içine bir takım yönergeler yerleştirmeyi gerektirebilir. Ama bu yönergeler de çok güçlü ve esnek değildir. Birçok CGI uygulamaları Web sayfalarına kod girilmesine sebep olacaktır ki bu da çoğu CGI scriptlerinin kullanım alanına girmektedir.

2.5.2 CGI Alternatiflerinin Çeşitleri

CGI seçeneklerini iki kategoriye yerleştirebiliriz: HTML destekli CGI seçenekleri, örneğin Microsoft' un Active Server Pages' leri, Allaire' s Cold Fusion ve PHP/FI gibi programlar Html içine özel uygulamalar girilmesini sağlayacaktır ki bunlar da CGI' a benzer fonksiyonellikleri taşıyabilmesi açısından önemlidir. Öte yandan mod_Perl, NSAPI ve ISAPI gibi kullanımlarda, server yazılımı ile birlikte direkt olarak iletişime girilmesini sağlayacak uygulamaları gösterir. Bu CGI ile yaşanan problemleri bir bakıma ortadan kaldırmaktadır.

Tahmin edilebileceği gibi web uygulamaları çerçevesini, daha çok geliştiricinin ne yapmak istediği ve en rahat hangi arabirimi kullandığı belirlemektedir. Html uzantısı olan uygulamalar web sayfasına çeşitli yönergeleri eklemekte ve onları daha dinamik hale getirmekte, aynı zamanda da dışarıdan veri alıp sayfaya koymada da faydalı olacaktır.

1- Active Server Pages

Active server pages (ASP), Html, scriptler ve ActiveX server bileşenleri kullanarak web sayfaları oluşturmaya yardımcı olur. Html sayfalarına script ler

ekleyerek ASP dökümanları oluşturulabilir. Eğer kullanıcı ASP istiyorsa server sayfa içerisinde bulunan scriptleri çalıştırır ve böylece scriptlerin çıktısı Html' nin bir parçası olarak görünür ve herhangi bir browser bu sayfayı gösterebilir.

ASP, VBScript ve Java Script gibi dilleri destekler ve Perl gibi diğer dillere de uyum sağlar. Kullanıcı ASP yi ilk defa download ederken sayfadaki yazılar tekrar düzenlenir ve server bu düzenlenmiş kaynakları kod değişene kadar hafızaya alır ve bu işlem aynen tekrarlanana kadar orada tutar. Bu metodu kullanarak ASP, CGI programları ile ilgili olan sorunları gidermede önemli ve etkili bir rol oynar. Çünkü scriptler her isteğe yanıt verecek şekilde düzenlenmeyebilir.

ASP' nin altyapısını oluşturan unsur ActiveX bileşenidir. Birçok kişi ActiveX kontrollerine aşinadır ve bunlar Microsoft' un Internet Explorer web tarayıcıları tarafından download edilebilir ve çalıştırılabilir. Ayrıca ActiveX kontrolleri Web server larına da fonksiyonellik sağlarlar. ASP dökümanları ActiveX bileşenleri ile çalışabilirler.

Örneğin en çok kullanılan bileşenlerden biri Microsoft' un Active Data Object idir (ADO). ADO, ASP ile bununla ilgili veritabanı arasında orta rolü oynamaktadır. Yine buna örnek olarak Ms Access ile hazırlanan veritabanından kayıtlar almak istiyorsanız scriptler ADO ya çağrı gönderecek ve böylece Access uygulamasından veritabanı yüklenecektir.

ASP unsurları Microsoft' un Component Object Model' ini (COM) desteklemektedir ve C++, VisualBasic, Java ve Cobol gibi birçok programlama dilinde de kullanılabilir. ASP' nin fonksiyonelliğini artırmak için yapılması gereken tek şey yeni unsurlar yazmaktır. Hatta üç partili unsurlarda kullanılabilir.

2- Netscpe LiveWire

Netscape LiveWire ve LiveWire Pro paketleri web sayfalarının kullanılması ve bunların desteklediği uygulamaları kullanmada önemli rol oynamaktadır. Live Wire, birçok veritabanı yazılımına SQL bağlantıları sağlayan kütüphaneler içerir. Ayrıca server side içerisinde javascript uygulamaları yazmanızı sağlayacak javascript compiler' ları içerir.

3- Allaire Cold Fusion

Cold Fusion, Windows NT server' ları bünyesinde kullanılan bir uygulamadır. Herhangi bir Windows NT web server' ı içerisinde geçerlidir ki bu Netscape Enterprise ya da Fast Track server' ları içermesi veya Microsoft ' s Internet Informatıon Server ya da O' Reilly web sitesi içermeside olasıdır. Cold Fusion ASP' nin işlemesine benzerlik gösterir. Dinamik sayfalar oluşturmak için sadece özel Cold Fusion etiketlerini web sayfasına girmek ve Cold Fusion uygulaması bu etiketleri yorumlayarak belge kullanıcıya gönderilmeden önce standart html gibi yerlerine yerleştirilir.

Cold Fusion, NSAPI ve ISAPI gibi teknolojileri kullanarak web server' lara direkt erişimi sağlamaktadır. Aynı zamanda CGI gibi yazılımlara da kurulabilir ki böylece basit bir arabirim içermeyen web server' larada da kullanılabilir.

4- PHP/FI

PHP/FI, web sayfası içerisine yerleştirilen script dillerin en etkililerindendir. Birçok komut içermektedir ve bu komutlarla birlikte veritabanlarına ve komut zincirlerine (if...else gibi) web sayfası içinden kolaylıkla ulaşılabilmektedir.

PHP/FI, herhangi bir Unix web server' ını çalıştırabilir. Yalnız bunun CGI' ı desteklemesi gerekir. Bunlar içerisinde CERN veya NCSA HTTPD server' ları olabilir.

Eğer Apache server kullanılıyorsa Apache modül kullanılabilir. Phtml uzantılı dosyalar PHP/FI modülü tarafından paylaştırılacaktır. Diğer server' lar üzerinde PHP/FI CGI scripti program sonuna ekstra olarak path ile belirtilerek kullanılabilir. PHP/FI ücretsiz olarak kullanıma sunulmuştur.

Diğer bir önemli özellik de Oracle, Sybase, Postgres, MySQL ve Adabas gibi birçok veritabanını desteklemesidir. ODBC' yi de desteklemektedir ki bu da sorgulamalrda kullanılan ayrı bir veritabanı yöntemidir.

5- ePERL

ePERL, web sayfası içinde perl scriptleri ile kullanılabilecek bedava bir yazılımdır. ASP' nin fonksiyonlarına benzer özellikler taşımaktadır ve aralarında çok büyük bir yakınlık vardır. Html sayfaları içinde kullanılan tüm kodlara sahiptir.

6- Server Side Java

Java, applet tasarlamak ve kullanmak için geliştirmiş faydalı bir dil olarak büyük bir önem kazandı. Appletler, web tarayıcısı üzerinde çalışan küçük uygulamalardır. Java' nın popülerliği arttıkça birçok yazılımın yazılmasında kullanılmaya başlandı. Java' ya ulaşmak için hazırlanan platformlardan biri de *Servlet* teknolojisidir. Servlet' ler, web server larına ek fonksiyonellik sağlamak için kullanılan uygulamalardır.

Daha önce de belirtildiği gibi CGI ile ilgili en önemli sorunlardan birisi kullanıcıyı belli bir web server ına kilitlemeleridir. Eğer Netscape Server API için program yazılmışsa tüm kodu başka bir dile ya da API ye ayarlamak gerekmektedir. Aynı şey Apache, Microsoft ASP ve bu bölümde tartışılan diğer teknolojiler için de söylenebilir.

- Sun, Java Servlet Development Kit' ini bu problemi halletmek için geliştirmiştir. Java' nın en önemli taraflarından biri uyumluluktur. Sun, Java Virtual Machine adında bir teknoloji geliştirmiştir ki bu da java uygulamalarını yürütmede birçok değişik platform üzerinde kullanılmaktadır. Eğer bir bilgisayarda JVM kuruluysa, uygulama derlemeye gerek kalmadan çalıştırılır. Sun buna pazarlama dilinde "Bir defa yaz, her yerde kullan" anlamında bir isim vermiş ve birçok popüler server da çalışabilen servlet ler yaratabilmişlerdir.

7- Netscape Server API

Netscape Server API, hemen hemen bütün netscape server ları içinde kullanılabilir. Bunların içinde; Enterprise, Fast Track, Commerce ve Communications Servers sayılabilir. Serverların içerisine dahili arabirim sağlar. Bu da kullanıcılara server a bir fonksiyonellik kazandırmaya ya da mevcut bölümleri daha üst versiyonlarla değiştirir.

NSAPI modüllerini, mevcut CGI scriptleri ile değiştirerek farklı şeyler yaratılabilir. Burda web server lara login sistemleri oluşturulabilir. HTTP nin sağlanması ve oluşturulması ile ilgili tüm gereksinimler NSAPI a maruz kalan bir süreçtir. Böylece bu sürecin herhangi bir bölümünü sürdüren bir kod yaratılabilir.

8- Microsoft' un Internet Server API' si

Internet Server API' si NSAPI' ye benzer bir fonksiyonellik sağlar. Tek farkı Microsoft' un Internet Information Server ı ile beraber çalışır. Daha önde de tekrarlandığı gibi CGI ile ilgili en önemli problemlerden biri CGI uygulamasının girildiği yerlerde program yeni süreç başlatır ki bu da server ı yoğun trafiği olan siteler de yavaşlatır. ISAPI bu problemi, uygulamasını dinamik olarak bağlı olduğu kütüphanelere uygulamalar sağlayarak halleder. Bu da server ın işletimi içerisinde çalışan ve dinamik olarak istenildiğinde bağlanılan bir sistemdir.

ISAPI iki şekilde kullanılabilir: uygulama ve filtre. Uygulamalar, CGI scriptleri olarak düşünülen ISAPI versiyonlarıdır. CGI uygulamalarını ISAPI ye çevirdiğimizde bunlar exe dosyaları olmaktan çıkarak server dışında çalıştırılabilirler ve böylece exe dosyasında, dll dosyasına çevrilirler. Dll' ler, dinamik olarak server a bağlanabilir ve server işletimi içerisinde çalışabilir. (Colburn, 1998)

3 VISUAL BASIC PROGRAMLAMA DİLİ

3.1 Visual Basic' e Giriş

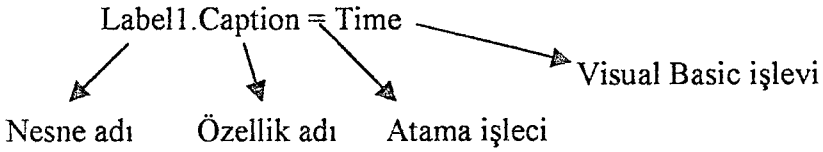
Microsoft Visual Basic 5.0, Microsoft Windows ve Windows NT için hızlı ve etkili uygulamalar oluşturmada kullanılabilecek güçlü bir programlama sistemidir. İlk Basic dili Dartmouth College' de 1963 yılında John G. Kemeny ve Thomas E. Kurtz tarafından yaratılmıştır. Daha sonra üniversite ve okullarda öğretim dili olarak yaygınlık kazanmış ve 1970' lerin ortalarında Bill Gates tarafından kişisel bilgisayarlarda kullanılmak üzere uyarlanmıştır.

Visual Basic' in tümüyle grafiksel olan geliştirme ortamı ve programlama dili ilk Basic yorumcularıyla pek benzerlik göstermese de, özgün Basic' in sahip olduğu şıklık ve yalınlık dilin içinde büyük ölçüde korunmuştur. Visual Basic' in gücü ve kullanım kolaylığı, Excel gibi Windows uygulamalarında programlama dili olarak seçilmiş olmasındaki temel etmendir. (Halvorson, 1999)

Visual Basic birçok işlemi yapmak için uygun bir dildir. Bunlar arasında; Internet, veritabanlarına giriş, resim işleme sayılabilmektedir. Visual Basic 4 nesne tabanlı programlama dilini ortaya koymuştur. Aynı zamanda OLE otomasyonunu da desteklemektedir. Bu da Visual Basic ile tekrar kullanılabilen uygulamalar yaratma açısından yeterli kılmış ve OLE unsurları ile birlikte etkileşimi sağlamıştır.

Visual Basic 5 versiyonu ActiveX kontrolleri yaratmada bir çok yönlülük ortaya koymaktadır. Bu da web sayfaları içerisine konulabilecek uygulamaların arttırmaktadır.

CGI programlarının bir uygulama olduğu düşünüldüğünde Visual Basic web sunucuları ile etkileşim kurmada kullanılabilmektedir. Bu da çok az bilinen Win-CGI çözümünü ortaya koymaktadır. (LaOr, 1998)



Bir programlama bildirisi hazırlarken uyulması gereken yapı kurallarına bildirinin söz dizimi denir. Visual Basic kullanıcıya, bir programı yazarken zaman kazanma ve elde edilen programları başka uygulamalarda kullanabilme olanağı vermektedir. (Halvorson, 1999)

3.3 **Değişkenler ve Veri Türleri**

Bir değişken programda verilerin geçici olarak depolandığı yerdir. Bu değişkenler , sözcük, sayı, tarih ya da özellikleri saklayabilmektedirler. Bir değişkeni açık olarak tanıtmak için, değişkeni kullanmadan önce Dim bildirisinden sonra değişkenin adının yazılması gerekmektedir. Değişken tanımlandıktan sonra türü de belirtilebilmektedir. Örneğin aşağıdaki bildiri Soyad isimli değişken için hafızada bir yer ayırmaktadır:

```
Dim Soyad as String
```

Bir değişken tanımlandıktan sonra artık program kodunda bu değişkene bilgi atanabilir. Bu atamadan sonra program içerisinde "Çuhadar" yerine artık Soyad değişkeni kullanılabilecektir. (Halvorson, 1999)

```
Soyad = "Çuhadar"
```

Yukarıda verilen son örnek aynı zamanda Dim bildirisi kullanılmadan yapılabilecek bir diğer değişken tanımlanma şeklidir. Bu tür tanımlamaya örtülü tanıtma adı verilmektedir.

3.3.1 Değişken Adlandırma Standartları

Visual Basic' de değişkenleri adlandırırken aşağıdaki kurallara dikkat edilmelidir: (Halvorson, 1999)

- Değişken adları bir harf ile başlamalıdır. Sayı ve harfler bir arada kullanılabilir ancak nokta karakteri kullanılmamalıdır.
- Değişken adları 256 karakterden uzun olmamalıdır.
- Visual Basic anahtar sözcükleri, nesne ve özellikleri değişken adı olarak kullanılamaz.

3.3.2 Temel Veri Türleri

Aşağıdaki tabloda Visual Basic' de kullanılan temel veri türleri gösterilmektedir: (Halvorson, 1999)

Tablo 3-1: Visual Basic' de temel veri türleri

Veri Türü	Boyut	Aralık	Örnek Kullanım
Integer	2 bayt	-32.768' den 32.767' ye	Dim Kuslar% Kuslar%=33
Long integer	4 bayt	-2.147.483.648' den 2.147.483.647' ye	Dim Ikraz& Ikraz&=460.000
Single precision floating point	4 bayt	-3.402823E38' den 3.402823E38' e	Dim Fiyat! Fiyat!=899.99

Double precision floating point	8 bayt	-1.79769313486232D308' den 1.79769313486232D308' e	Dim Pi# Pi#=3,1415926535
Currency	8 bayt	-9223371203685477.5808' den 9223371203685477.5808' e	Dim Kredi@ Kredi@=7600300,50
String	Karakter başına 1 bayt	0' dan 65.535 karaktere	Dim Kopek$ Kopek$="Leydi"
Boolean	2 bayt	True ya da False	Dim Cevap as Boolean Cevap=Ture
Date	8 bayt	1 ocak 100' den 31 Aralık 9999' a	Dim DogumGunu as Date DogumGunu=#23-4-1969#
Variant	Sayılarda 16 bayt, dizilimlerde karakter başına 1 bayt +22 bayt	Bütün veri türü aralıkları	Dim Toplam Toplam=289.13

3.3.3 Kullanıcı Tanımlı Veri Türleri

Visual Basic programcının kendi veri türlerini yaratmasına olanak vermektedir. Bu özellik, doğal olarak birbiriyle uyumlu ancak farklı veri türlerine ait veri öğelerini tanımlanmasında kullanılmaktadır. Type bildirisi kullanılarak kullanıcı tanımlı bir veri türü yaratılabilir ve Dim bildirisi ile de bu yeni türe ilişkin değişkenler tanımlanabilmektedir. Type bildirisi ile veri türü tanımlanması standart bir modülün Declarations bölümünde olmalıdır. Aşağıda bu tür bir tanımlamanın örneği verilmektedir: (Halvorson, 1997)

```
    Type Çalışan
        Ad as String
        DoğumTarihi as Date
        BaşlamaTarihi as Date
    End Type
Dim ÜrünMüdürü as Çalışan
ÜrünMüdürü.Ad = "Alper Tandoğan"
```

3.4 Temel Visual Basic Komutları

3.4.1 Karar İfadeleri

If yapısı

If şart yapısı bütün programlama dillerinde olan, bazı şartların gerçekleşmesi ya da gerçekleşmemesi durumunda ayrı ayrı kodların çalıştırılmasına imkan veren yapıdır.

```
If [şart] Then
        [Komutlar]
Else
        [Komutlar]
End If
```

Şartın gerçekleşmesi halinde Then deyiminden sonraki satır işletilir. Gerçekleşmemesi durumunda ise Else deyiminden sonraki satırlar işletilmektedir. Else bloğunun kullanımı isteğe bağlıdır. Ayrıca tek satırda şart yazılıyorsa End If kullanılmaz. Aşağıda If yapısının kullanımına ait bir örnek verilmektedir. (Karagülle ve Pala, 1997)

```
dim vize, final, sonuc as integer
vize = val(InputBox("Vize notu girişi")
final = val(InputBox("Final notu girişi")
sonuc = (vize+final)/2
if (sonuc>49) Then
        MsgBox("Başarılı")
Else
        MsgBox("Başarısız")
End If
```

Select Case Yapısı

Bir değişkenin aldığı bir çok değere göre ayrı komutların çalıştırılması gereken durumlar için Select Case yapısı kullanılabilir.

```
Select Case [Değişken]
        case [Durum1]: [Komutlar]
        case [Durum2]: [Komutlar]
        .....
        case [DurumN]: [Komutlar]
        case Else [Komutlar]
End Select
```

Burada, değişken parametresi ile belirlenen değişkenin aldığı duruma göre geçerli olan durum işleme girmektedir. Eğer değişkenin değeri hiçbir duruma uymuyorsa Case Else kısmındaki komutlar çalışacaktır. (Karagülle ve Pala, 1997)

```
dim vize, final, sonuc as integer
vize = val(InputBox("Vize notu girişi")
final = val(InputBox("Final notu girişi")
sonuc = (vize+final)/2
Select Case sonuc
        case 0 To 49: MsgBox("Başarısız")
        case 50 To 100: MsgBox("Başarılı")
        case Else: MsgBox("Not girişinde problem var")
End Select
```

IIF yapısı

Bir değişkenin değeri sadece iki durumdan birine göre değer alıyorsa IIF yapısı kullanılabilir. (Karagülle ve Pala, 1997)

IIF ([şart];[doğru ise];[yanlış ise])

Bu yapıya örnek olarak şöyle bir program parçası verilebilir:

```
Dim X as Integer
X= 1
IIF (X;Print "X sıfırdan büyüktür"; Print "X sıfırdan küçüktür.")
```

Choose Deyimi

Değişkenin aldığı değer bir sayı ise Choose yapısını kullanmak uygun olacaktır. Bu deyimin kullanım şekli aşağıdaki gibidir:

Sonuc = Choose([Sayı], [değer1], [değer2], [değer3],..... [değerN])

Burada sonuç sayıya bağlı olarak değerlerden biri ile eşleşmektedir. Örneğin, sayının 3 olması durumunda değer 3 olacaktır.

```
GunNo = 4
Gun = Choose(GunNo, "Pazar", "P.tesi", "Salı", "Çarşamba", "Perşembe", "Cuma", "C.tesi")
Print Gun
```

Choose yapısı kullanılarak oluşturulmuş olan bu örnek program parçası işletildiğinde, sonuç olarak form üzerine GunNo ile belirtilen 4. Değer, yani Çarşamba değeri elde edilecek ve forma yazılacaktır. (Karagülle ve Pala, 1997)

3.4.2 Döngü Deyimleri

For - Next Döngüsü

For – Next döngüsü bir sayacın başlangıç değerinden bitiş değerine kadar sayacı birer birer ya da istenilen aralıklarla arttırarak blok içerisindeki komutları çalıştırmaktadır. Sayacın azalarak işletilmesi için step değeri negatif alınmalıdır. Sayaç değişkeni maksimum değerine ulaşmadan döngünün sona erdirilmesi isteniyorsa Exit For deyimi kullanılmaktadır. (Karagülle ve Pala, 1997)

```
For [sayaç] = [Başlangıç değeri] to [Bitiş değeri] [step artım]
        Komutlar
Next sayaç
```

Bu deyimin kullanımına örnek olarak aşağıda 1' den 100' e kadar olan tek sayıları ekrana yazdıran bir program parçası verilmektedir.

```
Dim sayac as Integer
For sayac = 1 to 100 step 2
Print sayac
Next sayac
```

While – Wend ve Do – While Döngüsü

Bir şart gerçekleştiği sürece çalışması istenen program bloklarında kullanılmaktadır.

Tablo 3-2: While-Wend ve Do While döngülerinin kullanımı

Deyim	Kullanım şekli	Örnek
While – Wend deyiminin kullanımı	While [Şart] Komutlar Wend	Dim sayac as Integer Sayac= 1 While Sayac > 100 Print Sayac Sayac = Sayac + 2 Wend
Do – While Deyiminin kullanımı	Do While [Şart] Komutlar Loop	Dim sayac as Integer Sayac= 1 Do While Sayac > 100 Print Sayac Sayac = Sayac + 2 Loop

Do – Loop While ve Do – Loop Until Döngüsü

Bu tür döngülerde şart döngüye girerken kontrol edilmektedir. Dolayısıyla kod en az bir kez çalıştırılmaktadır. (Karagülle ve Pala, 1997)

Tablo 3-3: Do Loop Until ve Do Loop While döngülerinin kullanımı

Deyim	Kullanım şekli	Örnek
Do – Loop Until Deyiminin kullanımı	Do Komutlar Loop Until [Şart]	Dim sayac as Integer Sayac= -1 Do Sayac = Sayac + 2 Print Sayac Loop Until (Sayac>100)
Do – Loop While Deyiminin kullanımı	Do Komutlar Loop While [Şart]	Dim sayac as Integer Sayac= -1 Do Sayac = Sayac + 2 Print Sayac Loop While (Sayac<100)

Do – Until Loop Döngüsü

Bu döngü yapısı istenilen şart gerçekleşene kadar program bloğunun çalışmasını sağlamak amacıyla kullanılabilir. (Karagülle ve Pala, 1997)

```
Do Until [Şart]
        [Komutlar]
Loop
```

Aşağıda bu yapı için örnek bir program parçası verilmektedir.

```
Dim Sayac as Integer
Sayac = 1
Do Until (Sayac > 100)
        Print Sayac
        Sayac = Sayac + 1
Loop
```

3.5 Visual Basic ve Veri Tabanları

3.5.1 Klasik Dosya Sistemi

Klasik dosya sisteminin temel özelliği, saklanacak bilgiler ile bunları işleme sokacak olan bilgisayar programlarının birbirine bağımlı olmasıdır. Program, kullanacağı dosyanın yapısı ve bu dosyalara erişim biçemleri hakkında gerekli bilgiye sahiptir. Başka türlü bir ifade ile; dosya yapısı ve erişim şekilleri bilgisayar programının içine katılmış vaziyettedir. (Uysal, 1998)

Visual Basic üç tür klasik dosya sistemi kullanmaktadır:

1. Sıralı Erişimli Dosyalar (Sequential Acces Files): Bu tür bir dosyalama işleminde herhangi bir bilgiye erişmek ya da onu belleğe getirmek için dosyanın 1. kaydından itibaren istenen bilgiye ulaşana kadar tüm kayıtlar sırayla gözden geçirilmektedir.
2. Rasgele Erişimli Dosyalar (Random Acces Files): Bu tür dosyalarda istenen kayda, o kayda ait sıra numarası ile erişilir.
3. Index Örgütlü Dosyalar (Indexed Files): Bu tür dosyalarda, her kayda, birbirinden farklı anahtarlar yardımı ile erişilir. (Özel, 1993)

Klasik dosya sistemlerinin kullanımı birtakım sakıncaları da yanında getirmektedir: (Uysal, 1998)

1. Veri tekrarları
2. Çoklu güncelleme
3. Bellek hacminin israfı
4. Erişim dili

Visual Basic, veri tabanı yönetim sistemleri ile geliştirilen (Database Management Systems) veri tabanları ile mükemmel bir etkileşim göstermektedir. Geliştirilen uygulamalarda kayıtlar üzerinde iki değişik yöntemle işlem yapılabilmektedir:

1. Visual Basic komutları kullanılarak yazılacak kodlar ile doğrudan
2. Visual Basic içerisinde SQL (Structured Query Language-Yapısal sorgulama dili) kullanılarak dolaylı yoldan

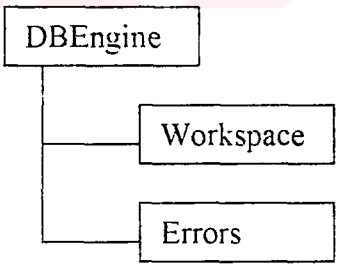
Visual Basic ile aşağıda belirtilen veri tabanı formatlarını kullanmak mümkündür:

- Microsoft Access
- DBase
- Microsoft FoxPro
- Excel
- Lotus

Ayrıca ODBC (Open Database Connectivity-Açık Veri Tabanı Bağlantısı) sürücüsüne bağlanarak, diğer türdeki veri tabanları ile de çalışmak da mümkündür. (Uysal, 1998)

3.6 Veri Tabanı Motoru Nesnesi (DBEngine Object)

DBEngine adlı veri tabanı motoru nesnesi, Veriye Erişim Nesne Modelinin (DAO – DataAccessObject) en üst düzeyindeki nesnesidir. Kullanıcının ya da sistemin veri tabanlarına veri kaydederken ve bu veri tabanlarından veri elde eden bir veri tabanı yönetim sistemidir. (Şekil 3-2)



Şekil 3-2: DBEngine ve alt düzey nesneleri

DBEngine nesnesi, DAO nesneleri ile hiyerarşi içerisinde, birçok nesne içermekte ve kontrol etmektedir. DBEngine, herhangi bir nesne gurubunun elemanı değil bağımsız tek bir nesnedir. Ayrıca ilave DBEngine nesneleri yaratmak mümkün değildir.

Bir mdb uzantılı Access dosyasına erişirken ya da ODBC yolu ile arka yüzeydeki (back-end) bir veri tabanı sunucusuna erişirken Jet Database Engine kullanılmaktadır. Microsoft Jet, farklı veritabanlarına erişim söz konusu olduğunda istekleri hedef veri tabanının anlayabileceği formatlara çeviren rutinler içermektedir. Çeviri rutinleri olarak nitelenebilecek bu rutinler Access dışındaki farklı veri tabanlarına erişim yapılırken kullanılmaktadır.

Uzak sistemlerde saklanan verilere ulaşabilmek amacıyla ise RDO (Remote Data Object) nesneleri kullanılmaktadır. RDO nesneleri de Microsoft Jet data-access nesnelerine (DAO) benzer şekilde programlanabilir nesnelerdir. Farkları, uzak sistemlerde saklanan verilere erişim sağlayan ilave özellikler ve metotlara sahip olmalarıdır. DAO ve RDO nesnelerinin içermekte olduğu nesne topluluklarının bir listesi aşağıda ki tabloda gösterilmektedir. (Uysal, 1998)

Tablo 3-4: DAO ve RDO veri erişim nesneleri

DAO Nesne Topluluğu	RDO Nesne Topluluğu
DBEngine nesnesi	RdoEngine nesnesi
Workspace nesnesi	RdoEnviroment nesnesi
Database nesnesi	RdoConnection nesnesi
TableDef nesnesi	RdoTable nesnesi
Field nesnesi	RdoColumns nesnesi
Index nesnesi	
Relation nesnesi	
Connection nesnesi	
Recordset nesnesi	

3.6.1 Veri Erişim Nesneleri (DAO – Data Access Object)

DAO. Microsoft Access ve diğer farklı veri tabanlarında tutulan verilere erişmek için kullanılan nesne modelidir. DAO, istemci-sunucu uygulamalarının zorunlu bir

parçası olan Microsoft SQL Server ve Oracle veri tabanları gibi uzak veri tabanı sunucularına da erişim sağlamaktadır.

Microsoft Jet, bir gurup veri erişim nesnesi (Data Access Object) içermektedir. Bu nesneler topluluk (collections) olarak isimlendirilen birliktelikler oluşturmakta ve kendilerine özgü özellik ve metotlar içermektedirler. Özellikler (properties), nesneleri tanımlayan karakteristikleri; nesneler ise bir nesne üzerinde yürütülebilecek yordamları temsil etmektedir. (Uysal, 1998)

3.6.2 Uzak Veri Erişim Nesnesi (RDO-Remote Data Object)

RDO nesneleri, DAO-Jet Data Access nesnelerine benzer bir şekilde programlanabilir nesnelerden oluşmaktadır. Bu tür nesneler de hiyerarşik bir yapı göstermektedirler. RdoEngine nesnesi bu yapının en üstünde yer almaktadır.

Uzaktaki veriye erişim bağlantı nesneleri aracılığı ile gerçekleşmektedir. Her bir RdoConnection nesnesi bir ya da daha çok veri satırlarından meydana gelen RdoResultSet nesnesi oluşturabilmektedir. RdoConnection nesnesinden RdoQuery ya da RdoTable nesneleri meydana getirilebilir. RdoQuery nesnesi aynı zamanda sorgulama işlemi esnasında gönderilen parametreleri yönetmek için kullanılan RdoParameter nesnesine sahiptir. (Uysal. 1998)

3.7 Visual Basic ve İlişkisel Veri Tabanları

Visual Basic' de veri tabanlarına doğrudan erişim için, dbEngine nesnesinin sağladığı görsel bir arakesit bulunmamaktadır. Yani form üzerine, tasarım esnasında yerleştirmekte olan kontrol nesnelerine benzer görsel nesneler, veri tabanı nesnesi ile

tasarımda mevcut değildir. Bu nedenle Visual Basic' de ilişkisel veri tabanlarına erişim amacıyla Microsft Jet Engine adı verilen bir veri tabanı motoru kullanılmaktadır.

Microsoft firmasınca geliştirilmiş olan, Microsft Jet-Engine 3.5 yazılımı, bir ilişkisel veri tabanı yönetimi (Relational Database Management) motorudur. Jet sözcüğü, Joint Engine Technology (Ortak Motor Teknolojisi) kelimelerinden oluşturulmuştur.

Microsoft Jet Engine, veri tabanı yönetiminin tüm işlevlerine sahip olmakla birlikte tek bir ürün değildir. Aksine, icra zamanı kütüphanelerinin (DLL – Dynamic Link Libraries) bir koleksiyonundan oluşmaktadır. Microsoft Jet Engine, kendi sahip olduğu Access formatının yanında, XBASE (dBase, FoxPro), Paradox, Btrieve, HTML ve ODBC ile anlaşabilecek arakesitlere de sahiptir. (Uysal, 1998)

3.8 Visual Basic ile Örnek Bir İlişkisel Veri Tabanı Uygulaması

3.8.1 Çalışma Ortamının Belirlenmesi

Mevcut bir veri tabanı açılmadan ya da yeni bir veri tabanı yaratılmadan önce çalışma ortamının (Workspace) belirlenmesi gerekmektedir. Visual Basic' de bu durum 3 ayrı yöntemle gerçekleştirilebilir:

1. Yöntem	**2. Yöntem**	**3. Yöntem**
Dim co As Workspace Set co=DBEngine.Workspaces(0)	Dim co As Workspace Set co=DBEngine(0)	Dim co As Workspace Set co=Workspaces(0)

Bazı uygulamalarda çalışma ortamının kullanıcı adı ve şifre ile açılması gerekebilir. Bu durumda aşağıda verilen kod satırları kullanılmaktadır: (Uysal, 1998)

```
Dim co As Workspaces
Set co=DBEngine.CreateWorkspace("co", "[kullanıcı adı]", "[şifre]")
```

3.8.2 Veri Tabanının Açılması

Çalışma ortamının tanımlanmasından sonraki aşama veri tabanının açılması olacaktır. Kullanılacak olan veri tabanını, Visual Basic içerisinden doğrudan açabilmek için şu komut satırı kullanılmaktadır:

```
Set vt=co.OpenDatabase("[Veri Tabanı Adı]", [Hususiyet], [Sadece Okunabilir])
```

Yukarıdaki komut satırında kullanılan parametrelerin açıklamaları ve alacakları değerler aşağıda açıklanmaktadır: (Visual Basic 5.0 ile İleri Uygulamalar)

Tablo 3-5: VB' de OpenDatabase deyiminin parametreleri

Parametre	Değeri	Anlamı
Veri Tabanı Adı	Dosya Adı	Açılacak olan veri tabanının yol ve dosya adı bilgileri
Hususiyet	True ya da False	Genellikle Update işlemi için kullanılan bu özellik veri tabanının diğer kullanıcılara kapalı olarak açılması için kullanılmaktadır.
Sadece Okunabilir	True ya da False	Append ve Edit metotları için kullanılan bu parametre false değeri aldığında veri tabanı sadece veri okunması amacıyla açılmaktadır.

Bu komutun kullanımına ilişkin bir örnek şu şekilde verilebilir:

```
On Error Goto Hata
Dim co As Workspace
Dim Veritab As DATABASE
Set co=DBEngine.Workspace(0)
Set Veritab=co.OpenDatabase("c:\ornek.mdb", False, False)
.....
Hata:
```

3.8.3 Tablolar ile Çalışmak

İlişkisel veri tabanlarında bilgilerin yüklü olduğu nesneler tablolardır. Tablolara erişim işlemini gerçekleştirmek amacıyla Visual Basic' de Recordset adı verilen DAO (Data Access Object – Veriye Erişim Nesnesi) kullanılmaktadır.

Recordset nesnesine ait metot ve özelliklerle tablolarla ilişkili işlemler gerçekleştirilebilir. Bu nesnenin Visual Basic içinde 3 tür kullanım şekli vardır. Bunlar sırasıyla:

1. Dynaset
2. Table
3. Snapshot

Recordset nesnesi ile çalışabilmek için öncelikle veri tabanı üzerine OpenRecordset metodu uygulanmalıdır. Bu nesnenin kullanımı ile ilgili örnek bir program parçası şu şekilde verilebilir:

```
Dim vt As DATABASE
Dim recset As Recordset
Set vt=co.OpenDatabase("c:\ornek.mdb")
Set recset=vt.OpenRecordSet("SELECT * FROM personel")
.....
```

Yukarıda verilen örnek program parçasında, ornek.mdb adlı veri tabanı içerisinden personel tablosundaki bilgilere erişebilmek amacıyla bir Recordset nesnesi oluşturulmuştur. OpenRecordSet nesnesinin genel yazım şekli şu şekildedir:

```
[Nesne].OpenRecordSet([Kaynak], [Tip], [Seçenekler], [Kısıtlar])
```

Bu nesne içerisinde kullanılan parametrelerin açıklamaları ise şu şekildedir:

Kaynak: Oluşturulacak Recordset' in kayıtları nereden alacağını bildiren ifadedir.

Tip: Bu parametrede, dbOpenTable, dbOpenDynaset veya dbOpenSnapshot değerlerinden birisi kullanılabilmektedir. Bu değerler ile ilgili açıklama ilerleyen bölümlerde yapılacaktır.

Seçenekler: Seçenekler parametresi için aşağıdaki tabloda görülmekte olan değerlerden birisi kullanılabilmektedir.

Tablo 3-6: OpenRecordSet nesnesi için parametreler

Sabit	Anlamı
DbDenyWrite	Çok kullanıcılı ortamlarda kayıtların değiştirilmesini engeller.
DbDenyRead	Çok kullanıcılı ortamlarda kayıtların okunmasını engeller.
DbAppendOnly	Kayıt ekleme işlemine müsaade eder.
dbIncontsistent	Dynaset tipi RecordSet' lerde tutarsızlık oluşturacak değişiklikleri müsaade eder.
dbConsistent	Dynaset tipi RecordSet' lerde tutarsızlık oluşturacak değişiklikleri engeller.
dbSQLPassThrough	Sorgulamanın MS Jet Engine' de değil, sunucu tarafındaki sorgu işlemcisinde gerçekleşmesini sağlar.
dbSeeChanges	Düzenlenen verilerin başka kullanıcılar tarafından değiştirilmesi durumunda oluşan hatanın bildirilmesinde kullanılmaktadır.

Kısıtlar: OpenRecordSet metodunun kısıtlar parametresi, çok kullanıcılı ortamlar için geçerlidir ve aşağıdaki değerleri alabilmektedir. (Uysal, 1998)

Tablo 3-7: Kısıtlar parametresinin alacağı değerler

Sabit	Anlamı
DbReadOnly	Kayıtlar sadece okunabilir türdendir.
DbPessimistic	Tablo üzerine konan kısıtlar Edit çağrısından sonra geçerlidir.
dbOptimistic	Tablo üzerine konan kısıtlar Update (güncelleme) işlemi süresince geçerlidir.

3.8.4 RecordSet Türleri (Dynaset, Table, Snapshot)

1. **Dynaset:** Kayıt ekleme, silme, değiştirme gibi işlemlere izin veren veri tabanı tabloları oluşturmak için kullanılmaktadır. Dynaset, bir SQL ifadesinin sonucu ya da bir tablo olabilmektedir.

   Örnek:

   ```
   Dim co As Workspace
   Dim dyn As Recordset
   Dim vt As Database
   Set co=DBEngine.Workspaces(0)
   Set vt=co.OpenDataBase("c:\ornek.mdb")
   Set dyn=vt.OpenRecordSet("SELECT ad FROM personel", _dbOpenDynaset)
   ```

2. **Table:** Bir tablo içerisinde, bir Index tanımlayarak hızlı bir arama yapmak gerekiyorsa Table tipinde bir Recordset oluşturmak uygun olacaktır. Table nesnesi, hiç bir zaman bir sorgunun sonucu olamaz. Sadece mevcut tam bir tabloyu içerebilir.

Örnek:

```
Dim co As Workspace
Dim tb As Recordset
Dim vt As Database
Set co=DBEngine.Workspaces(0)
Set vt=co.OpenDataBase("c:\ornek.mdb")
Set tb=vt.OpenRecordSet("personel", _dbOpenTable)
```

3. **Snapshot:** Statik yapıda bir tablo nesnesidir. Bu nedenle içerdiği bilgiler üzerinde değişiklik yapılamamaktadır. Sadece verilerin görüntülenmesi amacıyla kullanılabilmektedir. Bu nesne ile yapılacak sorgularda, sorgu sonucu bellekte tutulacağı için veri miktarının fazla olduğu tablolarda önemli bir hız problemi

meydana gelebilecektir. Bu nedenle Snapshot oluşturulurken yalnızca ihtiyaç duyulan alanlar seçilmelidir. (Uysal, 1998)

Örnek:

```
Dim co As Workspace
Dim sn As Recordset
Dim vt As Database
Set co=DBEngine.Workspace(0)
Set vt=co.OpenDataBase("c:\ornek.mdb")
Set sn=vt.OpenRecordSet("SELECT ad FROM personel", _dbOpenSnapshot)
```

3.8.5 Tablolara Uygulanan İşlemler

1- Kayıt Ekleme

Visual Basic' de, veri tabanı içerisindeki bir tabloya yeni bir kayıt eklemek için öncelikle AddNew metodu kullanılarak tabloya yeni bir boş kayıt eklenir. Daha sonraki işlem kaydı oluşturan alanlar içerisine atama yolu ile yeni kayıtların yerleştirilmesidir. Son olarak, Update (güncelleme) metodu çağırılarak kayıt ekleme işlemi sonuçlandırılmaktadır. (Uysal, 1998)

Örnek:

```
Dim co As Workspace
Dim t As Recordset
Dim vt As Database
Set co=DBEngine.Workspaces(0)
Set vt=co.OpenDataBase("c:\ornek.mdb")
Set t=vt.OpenRecordSet("personel", _dbOpenTable)
t.AddNew
t!sicil="14326"
t!ad="Hasan"
t!soyad="Caner"
t.Update
```

2. Kayıt Silme

Veri tabanı içerisindeki bir kaydın silinmesi için öncelikle o kaydın aktif hale getirilmesi gerekmektedir. Bu işlem gerçekleştirilirken hata oluşmaması için tablonun boş olup olmadığı kontrol edilmelidir. Uygulanacak ikinci ve son işlem ise Recordset nesnesi üzerine uygulanacak olan Delete metodu ile kaydın silinmesidir. (Uysal, 1998)

Örnek:

```
On Error Goto Hata
if Not t.Updateable Then
        MsgBox "Kayıt değiştirilebilir özellikte değildir!",32,"HATA"
        Exit Sub
End If
t.Delete
Exit Sub
Hata: MsgBox Error,32,"HATA"
        Exit Sub
        Resume Next
```

3. Kayıtların Listelenmesi

Recordset nesnesindeki kayıtların sıralanmasında Snapshot ve Dynaset tiplerini bir gurup, Table tipini ise ikinci bir gurup olarak ele almak uygun olacaktır.

Dynaset ve Snapshot tipindeki Recordset' lerde listeleme işlemini gerçekleştirebilmek için Sort özelliğinden ya da SQL ifadesindeki ORDER BY komutundan yararlanılmaktadır.

Örnek: (Sort metodu kullanılarak)

```
Dim co As Workspace
Dim t As Recordset
Dim vt As Database
Set co=DBEngine.Workspaces(0)
```

```
Set vt=co.OpenDataBase("c:\ornek.mdb")
Set t=vt.OpenRecordSet("SELECT * FROM personel")
t.Sort="[ad] Asc"
Set t=vt.OpenRecordSet()
```

Örnek: (ORDER BY ifadesi kullanılarak)

```
Dim co As Workspace
Dim t As Recordset
Dim vt As Database
Set co=DBEngine.Workspaces(0)
Set vt=co.OpenDataBase("c:\ornek.mdb")
Set t=vt.OpenRecordSet("SELECT * FROM personel ORDER BY ad")
```

Table tipindeki Recordset' lerin sıralanması için ise Index özelliği kullanılmaktadır. Fakat bu özelliğin çağırılabilmesi için ise daha önceden oluşturulmuş bir index dosyası mevcut olmalıdır. (Uysal, 1998)

Örnek: (Index dosyasının oluşturulması)

```
Dim Ind As Index
Set Ind=t.CreateIndex("personelindeks")
Set Fd=Ind.CreateField("ad")
Ind.Fields.Append Fd
t.Indexes.Append Ind
```

Örnek: (Kayıtların listelenmesi)

```
Dim co As Workspace
Dim t As Recordset
Dim vt As Database
Set co=DBEngine.Workspaces(0)
Set vt=co.OpenDataBase("VerTab1")
Set t=vt.OpenRecordSet("personel", dbOpenTable)
t.Index="personelindeks"
```

4. Kayıtların Aranması

Sıralama işleminde olduğu gibi, arama işleminde de Recordset nesnesinin farklı iki tipi arasında bir ayrım yapmak gerekmektedir. Dynaset tipi kullanılarak yapılacak bir arama uygun bir SQL-WHERE ifadesi kullanılarak gerçekleştirilebilir.

Örnek:

```
Dim co As Workspace
Dim t As Recordset
Dim vt As Database
Set co=DBEngine.Workspaces(0)
Set vt=co.OpenDataBase("VerTab1")
Set t=vt.OpenRecordSet("SELECT * FROM personel")
t.FindFirst "ad='Ahmet'"
if Not t.NoMatch Then Print "Kayıt bulunmuştur."
```

Yukarıdaki örnek program parçasında koşulu sağlayan ilk kayda konumlanmak amacıyla FindFirst metodu kullanılmıştır. Kayıtlar arasında hareket etmeyi sağlayan metodların listesi ve arama işlemi sırasında kullanılabilecek mukayese sembolleri aşağıdaki tablolarda verilmektedir:

Tablo 3-8: Kayıtlar arasında hareket etmeyi sağlayan metodların listesi

Metod	Anlamı
FindFirst	Koşula uygun ilk kaydı bulur.
FindNext	Koşula uygun bir sonraki kayda gidişi sağlar.
FindLast	Koşula uygun son kaydı bulur.
FindPrevious	Koşula uygun bir önceki kayda gidişi sağlar.

Tablo 3-9: Mukayese sembolleri

Mukayese Sembolü	Anlamı
<	Küçüktür
<=	Küçük eşittir
>=	Büyük eşittir
>	Büyüktür
<>	Eşit değildir
=	Eşittir

- Table tipindeki bir Recordset' de ise arama işlemi Seek metodu ile gerçekleştirilmektedir. Ancak bu metot uygulanmadan önce yine bir Index dosyası oluşturulmalıdır. (Uysal, 1998)

Örnek:

```
Dim x As String
Dim co As Workspace
Dim t1 As Recordset
Dim vt1 As Database
Set vt1= Workspaces(0).OpenDataBase("VerTab1")
Set t1=vt1.OpenRecordSet("personel", dbOpenTable)
t1.Index="personelsicilindeks"
x=InputBox("Aranacak sicili giriniz.")
t1.Seek "=", x
If Not t1.NoMatch Then
        MsgBox "Aranan kayıt bulunmuştur", 16 ,"Sonuç"
        Print t1!sicil, t1!ad, t1!soyad, t1!maas
Else
        MsgBox "Aranan kayıt bulunamıştır!", 48 ,"Sonuç"
        End
End If
```

4 JAVA VERİTABANI BAĞLANIRLIĞI (JDBC) VE SERVLET API

4.1 Java Programlama Diline Giriş

Internet, ortaya çıkışından itibaren çok büyük bir hızla gelişmiş ve yaygınlaşmıştır. Özellikle World Wide Web geniş bir kullanıcı kitlesine hizmet vermektedir. HyperText adı ile bilinen elektronik ortamda yayıncılık, kullanım kolaylığı ve sunduğu imkanlar bakımından bu gelişmeden en çok pay alan Internet hizmeti olarak göze çarpmaktadır.

Bu yüksek ivmeli gelişim içerisinde güçlü Internet uygulamaları oluşturma gereksinimi zamanla kendini göstermeye başlamıştır. Üretici firmalar ağ üzerinde geniş imkanlar sunabilecek yazılım ve bileşenler geliştirmeye başlamışlardır. Bunlar içerisinde Sun Microsystems, Java adını verdiği yeni programlama dili ile yazılım dünyasını çok önemli bir değişime uğratmıştır.

Java programlama dili C++ baz alınarak oluşturulmuştur. Bununla beraber Internet programcılığında kullanılan diğer dillere göre çok daha fazla avantaja sahiptir. Java' yı bu denli güçlü yapan nedenlerin bazılarını şu şekilde sıralayabiliriz: (Sun MS, 2000)

- Nesneye dayalı programlar geliştirebilme
- Taşınabilirlik
- Kanal yönetimi
- Ağ desteği
- Veritabanı bağlanırlığı (JDBC)
- Etkileşimli web uygulamaları geliştirebilme (Servletler)

4.2 Nesneye Dayalı Program Geliştirme

4.2.1 Nesnenin Tanımı

Nesneye dayalı programlama son yıllarda yazılım dünyasının en popüler kavramlarından biri haline gelmiştir. Bunun başlıca sebebi, geliştirilen programların esnek ve genişletilebilir olmasıdır. Bu önemli avantajının yanında nesneye dayalı yazılım geliştirme tekniğinin tam anlamıyla öğrenilmesi uzun zaman almaktadır.

Gerçek hayatta ki nesneler ile yazılım geliştirme sürecinde kullanılanlar arasından mantıksal açıdan bir ilişki kurulabilir. Örneğin, gündelik yaşama önemli ölçüde girmiş olan bilgisayarları ele alalım. Her bilgisayar markası, modeli, çevre birimleri gibi durumlara sahiptir. Yine her bilgisayarın açık ya da kapalı olması, klavye üzerinde bir tuşa basılıp basılmaması gibi çeşitli davranışları olacaktır. Bu durumda nesne, bir ya da daha fazla duruma sahip olan ve bunlarla ilişkili çeşitli davranışlar sergileyebilen kavramdır. Birbiriyle ilişki kurabilen farklı nesneleri uygun bir şekilde birleştirerek programın bütününe ulaşılmaktadır.

Yazılım nesneleri de gerçek hayattaki nesnelerdeki gibi bir durum ve davranışa sahiptir şeklinde modellenebilir. Bir yazılım nesnesi durumunu değişkenler ile ve davranışını da metotlar ile sürdürmektedir. (Sun MS, 2000)

4.2.2 Nesneler ile İlgili Kavramlar

Değişkenler

Program içerisinde çeşitli işlemler yapabilmek ve bu işlem sonuçlarını bellekte saklayabilmek amacıyla veri tipleri ve değişkenler kullanılmaktadır. Değişkenler harflerden, sayılardan ve $, _ gibi bazı özel karakterlerden oluşmaktadır.

Java' da kullanılan değişkenler kullanım amaçlarına göre şu üç kategoriden birine dahildir:

1. **Örnek (Instance) değişkenler:** Program içerisinde tüm nesne, yöntem ve bunlara ait sıfatlar tarafından kullanılması istenen değişkenlerdir.
2. **Sınıf (Class) değişkenleri:** Örnek değişkenler gibidirler. Fakat sınıf değişkenleri diğerinin aksine sadece sınıf içerisindeki nesneler için geçerli olmaktadır.
3. **Yerel (Local) değişkenler:** Yalnızca içinde yer aldığı fonksiyon ya da yöntem içinde geçerli olup, tanımlı oldukları blok bittiğinde geçerlilikleri sona ermektedir. (Grup Java, 1997)

Aşağıda bu üç tür değişkeni kullanan bir örnek program parçası verilmektedir.

```
class Hafta {
   // Sınıf değişkeni tanımlanıyor...
   String gunler;
      public static void main(String args[]) {
         // Yerel değişken tanımlanıyor...
         String mesaj ="Bu gun ";
         Hafta tarih = new Hafta();
         // Örnek değişken tanımlanıyor...
         tarih.gunler="pazar";
         System.out.println (mesaj + tarih.gunler);
      }
   }
```

Değişkenler kullanılmadan önce mutlaka uygun veri tipi ile deklare edilmelidirler. Java' da kullanılan veri tipleri aşağıdaki tabloda gösterilmektedir.

Tablo 4-1: Java' da kullanılan veri tipleri

Veri Tipi	Büyüklüğü	Alabileceği Min. ve Max. Değerler
Byte	8 bit	-128 +127
Short	16 bit	-32768 +32767
İnt	32 bit	-2147483648 +2147483647
Long	64 bit	-(2 exp 64) +(2 exp 64)
Float	32 bit	3.4e-38 3.4e+38
Double	64 bit	1.7e-308 1.7e+308
Char	16 bit	0 65536
Boolean	Mantıksal değer	Yalnızca "true" veya "false"

Sınıflara Ait Genel Özellikler

Nesne yönelimli programlama ile ilgili en önemli kavramlardan biri sınıflardır. Sınıflar, birbiriyle benzer özellikleri olan nesnelerden meydana gelmektedirler.

Çalıştırılabilir Java kodunun her satırı mutlaka bir sınıf tanımına aittir. Java, temsil edebileceği her şey (düğme, applet, URL, dosya ve hatta sınıfların kendisi) için sınıfları kullanır. (Heller, Seymour, vd., 1998)

Sınıf kullanılmadan önce mutlaka örneklenmelidir. Bir sınıf örneği yaratıldığında o tipte bir nesne yaratılmış olur ve örnek değişkenin sınıf ile deklare edilmesiyle sistem hafızasında yer tahsis edilir. (Sun MS, 2000)

Program geliştiriciler tarafından oluşturulan sınıflar dışında Java standart bir sınıf kütüphanesine sahiptir. Yazılım sektöründeki hızlı gelişmeye ayak uydurabilmek amacıyla Sun Microsystems, Symantec gibi üreticiler bu kütüphaneleri sürekli güncellemektedirler. Aşağıda standart olarak sunulan bu paketler kısaca açıklanmaktadır.

1. **java.lang Paketi:** Bu paket, Java' da kullanılan tüm temel elemanları içermektedir. Değişkenler, hata ayıklama, güvenlik gibi genelde her programda standart olarak kullanılan elemanlar burada tanımlıdır.

2. **java.io Paketi:** Verilerin sistemdeki giriş/çıkış aygıtlarında veya dosyalarda işlenmesine yönelik bir pakettir.

3. **java.util Paketi:** Program içerisinde kullanılacak olan bazı kapsamlı veri yapıları, tarih, zaman v.s. kullanımını sağlayan sınıfları içermektedir.

4. **java.net Paketi:** Ağ uygulamalarında kullanılacak olan URL, connect gibi nesne ve arabirimleri içerir.

5. **java.awt Paketi:** *AWT (Abstract Window Toolkit) Soyut Pencere Aracı* kullanıcı arabirimini oluşturmak üzere tüm grafiksel öğeleri içermektedir. Menüler, fontlar, renkler ve diğer tüm grafiksel kullanıcı arabirim elemanları bu paket içerisinde bulunmaktadır.

6. **java.applet Paketi:** Bir tür Java uygulaması olan ve Web sayfaları üzerinde çalıştırılabilen appletleri oluşturmak için gerekli sınıflar bu paket içerisinde yer alır. (Grup Java, 1997)

Sınıfların Oluşturulması

Oluşturulan yeni bir sınıf içerisinde temel olarak metotlar ve değişken tanımlamaları olacaktır. Bir sınıf oluşturmak için önce onu deklare etmek

gerekmektedir. Eğer sınıf kütüphanelerinden bir ya da birkaçı program içerisinde kullanılacak ise bunlar da henüz başta tanımlanmalıdır. (Sun MS, 2000)

Yeni bir sınıf meydana getirmek için class ifadesi ve sınıf ismi kullanılır:

```
[Erişim Denetimi] class [Sinif İsmi]
{
        ["Metot ve değişkenler...."]
}
```

Oluşturulan sınıf bir üst sınıfa ait alt sınıf ise deklarasyon şöyle olacaktır:

```
class [Alt Sınıf] extends [Üst Sınıf]
{
        ["Metot ve değişkenler...."]
{
```

Java' da ki standart sınıflardan oluşan paketleri kullanabilmek için ise import ifadesi kullanılmalıdır. Örneğin:

```
import java.awt.Font;

public class [Alt Sınıf İsmi] extends [Üstsınıf İsmi]
{
        ["Metot ve değişkenler...."]
}
```

Metot ve Nesnelerin Oluşturulması

Metotlar nesnenin davranışını tanımlar. Nesne oluşturulduğunda ne olacağı, nesnenin kullanımı sırasında neler olması gerektiği hep metotlar sayesinde saptanır. (Grup Java, 1997)

Java' da metot tanımlaması şu şekilde yapılmaktadır:

```
[Metot Tipi] [Metot İsmi] [(Parametreler)]
{
        [Değişkenler ve program kodları...]
}
```

Bu tanımlama içerisinde Metot Tipi Java' da kullanılan int, boolean gibi herhangi bir tip olabilir. Tip, metodun geri dönüş değerine bağlı olarak seçilmektedir. Eğer işlem sonucunda dönen bir değer olmayacak ise burada void ifadesi kullanılmaktadır.

Program içerisinde başka metotlar ile iletişim söz konusu ise parametreler kullanılır. Bu parametreler yine Java' ya ait veri tipleri içerisinden seçilmektedir. Birden fazla parametre kullanılacak ise bunlar virgül kullanılarak birbirinden ayrılmalıdır.

Java' da bir metot oluşturulduktan sonra artık nesneler bunların içine dahil edilebilmektedir. Nesne oluşturmak için ise 2 adet bileşen kullanılmaktadır; nesnenin ait olduğu sınıf ve new ifadesi:

```
[Nesnenin Tipi] [Nesne İsmi] = new [Sınıf]
```

Oluşturulan yeni nesne için Java' nın diğer dillerden farklı bir özelliğinden yararlanılmaktadır. Programcıyı önemli bir yükten kurtaran bu özellik dinamik hafıza kullanımıdır. Bu özellik, nesne meydana geldiğinde hafıza da onun için otomatik olarak bir yer ayırır. İşlem sona erdiğinde ise hafıza da ayrılan bu yer serbest bırakılmaktadır.

4.3 Taşınabilirlik

Java, her ortamda çalışabilen bir teknolojiyi yazılım dünyasına kazandırmıştır. Sun Microsystems bu savını şu şekilde sunmaktadır. "Düşünce basittir. Java teknolojisi tabanlı yazılımlar küçük araçlardan süper bilgisayarlara kadar her yerde çalışır. Java teknoloji bileşenleri, bilgisayarın türünü, telefon, TV veya çalıştığı işletim sistemini umursamaz. Sadece Java' nın desteklediği Java uyumlu tüm cihazlarda çalışır." (Sun MS, 2000)

Esasen Java ile oluşturulan kaynak dosya ASCII karakterlerden meydana gelen bir metin dosyasıdır. Dolayısıyla herhangi bir metin editörü ile yazılabilmektedir. Fakat bununla birlikte görsel araçlar ile donatılmış çeşitli editörler de mevcuttur. Borland, Sun, Microsoft gibi çeşitli üreticilerin ürünleri yaygın olarak kullanılmaktadır. Microsoft Visual Java++, Java Development Kit ve Java Builder buna örnek olarak gösterilebilir.

Java programları yazıldıktan sonra *bytecode* denilen ve işlemciye özel olmayan kodlar olarak derlenir. Böylelikle uygulama çalışmak için yalnızca JVM (Java Virtual Machine) adı verilen bir yorumlayıcıya ihtiyaç duyacaktır. JVM' nin platforma bağımlı olmaksızın Unix, Windows gibi her işletim sisteminde çalışabilme yeteneği ise Java programlarına *taşınabilirlik* özelliğini vermektedir.

Taşınabilirlik ya da *platformdan bağımsızlık* Java programcıları ve kullanıcıları için çok önemli bir avantajıdır. C++ gibi çok büyük benzerlikler gösterdiği güçlü bir dil

bile bu açıdan yetersiz kalmaktadır. Örneğin IBM tabanlı bir bilgisayarda derlenen C++ programları, Mac tabanlı diğer bir bilgisayarda çalışmamaktadır. Bunun için oluşturulan program mutlaka yazıldığı bilgisayarda derlenmelidir. Bytecode' lar şeklinde derlenmiş Java programlarının ise bu tür bir gereksinimi olmayacaktır. (Grup Java, 1997)

Platformdan bağımsız uygulamalar geliştirebilme özelliği programcılar açısından çok önemli bir avantaj olarak görülmektedir. Fakat yine de farklı konfigürasyonlara sahip makinelerde çalışabilecek yazılımlar geliştirebilmek için birkaç önemli noktaya dikkat edilmesi gerekmektedir. Sorunsuz ve tam olarak taşınabilir denilebilecek uygulamalar yaratabilmek için dikkat edilmesi gereken önemli noktalar şunlardır: (Heller, Seymour, vd., 1998)

- Veri gösterimi
- Zamanlama
- Görsel konular
- Güvenlik etkileri

4.3.1 Veri Gösterimi

Java' da kullanılan değişken ve veri tipleri küçük farklılıklar olsa da diğer dillerle hemen hemen aynıdır. Örneğin C++ dili ve türevlerinde kullanılan pointer türü Java' da yoktur. Veri gösterimi platformdan bağımsız, hatasız çalışması istenen uygulamalar için önem taşımaktadır. Derleyiciden ve platformdan bağımsız bir uygulama içerisinde veriyi oluşturan elemanlar aynı şekilde görünmeli ve davranmalıdırlar. 32 bit değerinde bir tam sayı uygulamanın çalışacağı her ortamda yine bu şekilde davranış göstermeli ve algılanabilmelidir.

Verilerin ağ üzerinde transferi ile de birtakım sorunların ortaya çıkması mümkündür. Farklı sistemler kodlama için kendilerine özgü yöntemlere sahiptir. Verinin taşınması esnasında bu farklı olabilecek yöntemler uyumsuzluklara sebep olabilmektedir. Bu durumda iki ayrı çözüm ortaya çıkar. İlki, uygulama oluşturulurken farklı ortamlarda çalışabileceğinden alınan ya da verilen veri buna göre kodlanmasıdır. Diğer çözüm ise uygulamanın Java' yı desteklemeyen bir sistemde çalıştığında verinin kontrolüne izin verilmesidir. (Heller, P., Seymour, P., vd., 1998)

4.3.2 Zamanlama

Bir uygulamanın mükemmel olması yürütme için kullanılan zamanın kısalmasını sağlayamamaktadır. Bu tamamen mikroişlemcinin ve donanımın yeteneklerine bağlı bir durumdur. Yavaş bir işlemcinin hesaplamayı zamanında tamamlayamaması uygulamada hataya sebebiyet verebilmektedir. Bu durumda algoritma da düzeltmeler yaparak problem giderilebilinir. Aksi halde donanım terfisi gerekecektir.

Karşılaşılan diğer bir sorun da kanal mekanizmasının kullanımından kaynaklanmaktadır. Örneğin, iki kanal üzerinde çalışması istenen bir programda işlemin sürdürülmesi hesaplamadan çok disk üzerindeki işlemin tamamlanmasına bağlı ise bir problem ortaya çıkacaktır. Eğer uygulama yavaş bir makinede çalışıyor ise bir sorun görülmeyebilir. Fakat hızlı bir işlemciye sahip makinelerde ilk kanal işini daha çabuk bitirecek. bu da programın başarısız olmasına sebep olacaktır. Heller, P., Seymour, P., vd., 1998)

4.3.3 Görsel Konular

Windows X, Macintosh gibi ortamlardan her birinin kendine özgü bir davranış şekli ve görünümü bulunmaktadır. Bir Java uygulamasının da bu tür değişik

platformlara uyumlu olması beklenmektedir. Aksi bir durum kullanıcı açısından alışılagelmeyen bir durum olarak kabul edilebilir ve programın kullanılması istenmeyebilmektedir. (Heller, P., Seymour, P., vd., 1998)

Java birçok platform ile ortak bir standart sağlayarak bir GUI (Graphic User Interface - grafiksel kullanıcı arayüzü) oluşturmaya imkan veren bir araç takımı içermektedir. AWT (Abstract Windowing Toolkit - Soyut Pencereleme Aracı) adı verilen bu araç takımı programcılara şu imkanları sunmaktadır: (Gurup Java, 1997)

- Menüler, butonlar, text alanları vs.
- Renk ve font seçimi
- Yerleşim elemanları
- Kullanıcı olaylarının yönetimi

4.3.4 Güvenlik Etkileri

Güvenlik Java' da iki kısım olarak ele alınmaktadır. Bunlardan birincisi, bir applet in kullanıcının yerel kaynaklarına erişme yeteneğinin olamamasıdır. İşlemci, hafıza ve diğer kaynakların kontrolü istemci tarafında olacaktır. Diğer bir güvenlik sorunu ise ağ üzerinde yetkili olmayan kişilerce yapılmak istenen veri değişiklikleridir. Java bu her iki sorunu da kullanıcının rahatça çalışabileceği bir güvenlik mekanizması sağlayarak gidermektedir. (Heller, P., Seymour, P., vd., 1998)

4.4 Kanal Yönetimi

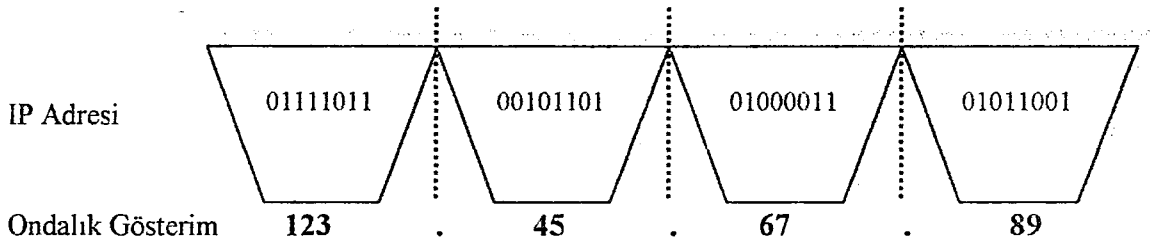
Java programlama dilini güçlü bir yazılım geliştirme aracı durumuna getiren önemli özelliklerinden birisi de çok kanallı olmasıdır. Çok kanallılık, mikroişlemcinin birden fazla görevi aynı anda yapabilmesidir. Çok kanallı bir yapılanma, görevlerin ancak uygun bir zamanlama ile çalışmasının sağlanması şeklinde gerçekleşebilmektedir.

Tek kanallı olarak tasarlanan bir uygulama da bir işlem başlamakta ve sonuç alınana kadar yürütme devam etmektedir. Çok kanallı bir ortamda ise mikroişlemci aynı anda bir çok görevi yerine getirmek üzere görevlendirilmektedir. Örneğin, bir kanalda klavyeden bilgi girişi için bekleniyorsa, diğer program bölünmüş diğer kanal ya da kanallar vasıtası ile bir başka işlemi yerine getirme şansına sahiptir.

Bir işlemci üzerinde kanallardan hangisinin o anda yürütülmekte olduğunun gösterilmesi Kanal Planlama Modeli olarak adlandırılmaktadır. Bu model öncelikli ve zaman paylaşımlı olarak iki guruba ayrılmaktadır. Öncelikli modelde, yürütmede öncelikli kanal ilk önce ele alınmaktadır. Zaman paylaşımlı model de ise, işlem önceliğe bakılmaksızın kanallar arasında uygun bir zamanlamayla paylaştırılmaktadır. Java bu modellerden herhangi birini tam olarak desteklememektedir. Bunun yanında zaman paylaşımlı modelin kullanılması beklenmektedir. Kanal planlamasının doğru yapılmadığı uygulamalar da verimin düşmesi ve platforma bağımlılık söz konusu olabilmektedir. (Heller, P., Seymour, P., vd., 1998)

4.5 Ağ Desteği

Java Internet' in temelini oluşturan TCP/IP (Transmission Control Protocol/Internet Protocol) protoklünü desteklemektedir. Bu protokolün işleyişi, gelen ve giden verinin birer akım olarak değerlendirilmesine dayanmaktadır. Akımlardan biri bir bilgisayardan bir yönde veri taşırken diğeri tersi yönde bir hareket gerçekleştirmektedir. Bu iletişimi gerçekleştirmek için onluk düzende bir aritmetiksel adres meydana getirilmektedir.



"helpful.com"
bölgesi için
ad-sunucu
veritabanındaki kayıt

barney=123.45.67.89

adrese karşılık
uygun bir ad
sağlanır

barney.helpful.com

Şekil 4-1: TCP/IP adresleme formatı

Bu adresleme tekniği Internet tarafından DNS (Domain Name System) olarak adlandırılmaktadır. DNS, adreslerin akılda tutulması daha güç olan sayılar yerine alfabetiksel olarak gösterimi sağlayan bir mekanizmadır. Bu adresleme tekniği içerisinde düğümü oluşturmakta olan her sayı, temsil ettiği organizasyonun türü, adı, bulunduğu ülke gibi bilgileri içermektedir.

Java ile bir istemci/sunucu bağlantısı yaratmak diğer dillere göre daha basit gerçekleştirilmektedir. Ancak bağlantı kurulduktan sonra haberleşme kontrol altında tutulmalıdır. Ağ üzerinde taşınacak olan veri dört değişik formatta olabilmektedir:

1. ASCII: Karakter yazıldıktan hemen sonra monitör gibi bir uç birimde görüntülenebilmektedir. Sistemde hata ayıklama daha kolay olmaktadır.

2. UNICODE: 16-bit gösterimdir. Hizmeti kullanan çeşitli diller mevcut ise iyi bir seçenek olarak görülebilmektedir.

3. UTF: Unicode setindeki tüm karakterleri gösterebilmektedir. Aynı zamanda ASCII olmayan karakterleri de desteklemektedir.

4. BINARY: Sayısal verinin en doğal iletim şeklidir. Büyük miktarda veri iletiminde verimlilik açısından önerilir.

Java, ağ kullanımı ve güçlü istemci/sunucu uygulamaları geliştirebilmek üzereprogramcılara standart bir paket sunmaktadır. Bu paket URL desteği sağlamakta ve aynı zamanda soket bağlantılarına da izin vermektedir. Ancak bu paketin desteklemediği durumlarda mevcuttur. Buna örnek olarak, bant-dışı veya acil isteklerin işlenmesi verilebilmektedir. (Heller, P., Seymour, P., vd., 1998)

4.6 Java ile Program Geliştirme

Java ile uygulamalar ve applet' ler şeklinde iki tür program geliştirilebilir. Uygulamalar genel amaçlıdır ve tek başına yürütülebilen kodlardır. Applet' ler ise yalnızca Web tarayıcıları üzerinde çalışabilen ve daha çok dinamik sayılabilecek sayfalar yaratmada kullanılan programlardır.

4.6.1 Uygulamalar

Uygulamalar temel olarak sınıflardan ve main() fonksiyonundan oluşmaktadır. Aşağıda örnek teşkil etmesi açısından başlangıç seviyesinde bir Java uygulaması verilmiştir.

```
class Merhaba {
        public static void main(String args[]) {
                System.out.println("Java ile ilk program."); }}
```

Örnek uygulamada da görüldüğü gibi öncelikle bir sınıf tanımı yapılmaktadır. Esas program ise main() fonksiyonu içerisinde oluşturulmuştur. Programlar yorumlayıcı tarafından çalıştırılabilmeleri için bu ana fonksiyona ihtiyaç duyarlar. Uygulama oluşturulduktan sonra .java uzantısı ile kaydedilmeli ve derlenmelidir. Bu işlemi gerçekleştirmek ve programı çalıştırmak üzere aşağıdaki adımlar takip edilmelidir.

1. Adım: c:\jdk1.1.6\bin\>javac Merhaba.java
2. Adım: c:\jdk1.1.6\bin\>java Merhaba
Sonuç: Java ile ilk program.

1. Adımda derleyici *Merhaba.class* sınıf dosyasını oluşturmuştur. Yorumlayıcı ise oluşturulan bu bytecode dosyasını 2. Adımda JVM' de çalıştırarak yukarıda görülen sonucu vermektedir.

4.6.2 Appletler

Appletler de durum biraz daha farklıdır. Bu türden Java programları uygulamalarda olduğu gibi aynı yol ile yazılır ve derlenirler. Fakat bazı yapısal farklılıklar göstermektedirler. Örneğin bir appletin main() metodu yoktur. Bunun yerine değişik işlevleri olan beş adet farklı metoda sahiptirler:

1- init (): Applet ilk kez yüklendiğinde değişkenlere ilk değer, font, yazı tipi ayarı gibi bir takım başlangıç değerlerini atamak için kullanılır.

2- start(): Gerekli atama ve ayarlar yapıldıktan sonra başlatma için bu metot kullanılır.

3- paint(): Web sayfasına herhangi bir şey basmak istendiğinde kullanılır.

Graphics sınıfına ait bir argümana sahiptir. Program içerisinde diğer metotların aksine bu argümanla *public void paint (Graphics g)* şeklinde kullanılır.

4- stop(): start metodunun tam tersidir. Aktivitenin sona ermesi istendiğinde kullanılır.

5- destroy(): Tarayıcının appletin bulunduğu sayfayı terk etmesi sonucunda applet hafızadan bu metod ile temizlenmektedir.

Appletleri çalışma şekillerinde de önemli bir farklılık görülmektedir. Bunları gerekli kod parçalarını kullanarak Web sayfalarına dahil etmek ve daha sonra bu sayfaları Internet Explorer 5 gibi Java destekli bir tarayıcıda görüntülemek gerekmektedir. Aşağıda daha önce uygulama olarak oluşturduğumuz örnek programın applet şeklindeki hali ve ekran çıktısı görülmektedir. Tarayıcıdan bu çıktıyı alabilmek için öncelilkle applet kaynak koduna referans sağlayan bir web sayfası oluşturulmalıdır. (Grup Java, 1997)

Web sayfasının içeriği:

```
<html>
<head>
<title>İlk Applet</title>
<applet code="Merhaba.class" width=500 height=50></applet>
</head>
</html>
```

4.6.3 Uygulamalar ve Appletler Arasındaki Farklar

Java' da uygulamalar ve appletler arasında çok önemli farklar bulunmaktadır:

- Uygulamalar güvenilir, appletler ise güvenilmeyen kodlar olarak nitelendirilirler.
- Appletler istemci tarafındaki tarayıcı tarafından önemli ölçüde kısıtlanmaktadır.
- Uygulamalar yerel disklerdeki dosya sistemlerine erişebilmektedir.
- Uygulamalar tek başına yürütülebilen kodlardır. Appletler ise çalışmak için Java destekli bir tarayıcıya ihtiyaç duyarlar.

4.7 Java ile Programlamanın Temelleri

4.7.1 Diziler

Aynı özellikte değişkenler tarafından oluşturulan grup dizi olarak adlandırılmaktadır. Java' da diziler nesne özelliği göstermektedirler. Dizi oluşturmada öncelikle *tip* belirtilir. İkinci adım ise *new* operatörü kullanılarak dizinin yaratılmasıdır. Dizinin büyüklüğü de yine bu son adımda belirtilir. Örnek bir tanımlama aşağıdaki gibi olacaktır:

```
String[] mesaj;
String mesaj[]=new String[3];
```

Diziler tanımlandıktan sonra değer ataması yapılabilir. Bu atama yapılmadığı sürece dizi elemanlarının varsayılan değerleri; *String* veri tipi için *null*, sayısal tipler

için ise 0 (sıfır) olacaktır. Yukarıda yaratılan üç elemanlı diziye değer ataması aşağıdaki şekilde yapılmaktadır:

```
mesaj[0]="Java";
mesaj[1]="C++";
mesaj[2]="Visual Basic";
```

Java' da çok boyutlu dizi tanımlaması yapmakta mümkündür. Fakat bu tanımlama direkt olmamaktadır. Ancak dizi içerisinde dizi oluşturarak çok boyutlu bir dizi yaratılabilir.

4.7.2 Döngüler

Java içerisinde üç tür döngü yapısı görülmektedir. Bunların kullanımı programcının amacına ve döngünün özelliklerine göre değişmektedir.

Eğer başlangıç değeri belli ise ve bu değer belli bir artım miktarına göre mantıksal bir ifadeyi test ediyor ise *for* döngüsü kullanmak yerinde olacaktır. Bu tür bir döngüye ait kullanım şekli şöyledir:

```
int x;
for(x=0;x<=10;x=x+2){
    System.out.println("Şu an ki x değeri = "+x);
}
```

Örnek olarak verilen program parçasında x değişkeni, +2 artım miktarı ile 10 değerine ulaşana kadar çevrime tabi tutulacaktır. Döngü her defasında blok içerisinde kullanılan komut ve/veya komut satırlarını işleyecektir.

İkinci döngü türü *while* yapısıdır. Burada yalnızca mantıksal bir değer vardır. Blok içerisinde kullanılacak olan komut satırları bu değer *true* olduğu sürece işletilecektir.

```
int x=0;
while (x<10){
    System.out.println("Şu an ki x değeri = "+x);
    x=x+1;
}
```

Son yapı ise *do.....while*' döngü türüdür. Bir önceki while döngüsünden tek farkı en az bir kez çalıştırılabilir oluşudur. Bu da test ifadesinin döngünün sonuna konmasıyla gerçekleştirilmektedir.

```
int x=0;
do {
    System.out.println("Şu an ki x değeri = "+x);
    x=x+1;
} while (x<10);
```

4.7.3 Şart İfadeleri

Java' da şart ifadeleri olarak *if* ve *switch case* yapıları görülmektedir. Hemen hemen aynı görevi gerçekleştiren bu iki yapı arasında bazı farklar bulunmaktadır.

If şartında mantıksal bir ifadeye bağlı olarak programın gidişatına karar verilmektedir. Örneğin, tanımlanmış bir x değişkeninin değeri 10' dan küçük veya 10' a eşit ise aşağıda gösterildiği gibi ekrana bir mesaj yazılacaktır:

```
if(x<=10) {
        System.out.println("x değeri 10 dan küçüktür.");
        }
```

Oluşacak harekete bağlı olarak birden fazla durum söz konusu ise *else* deyimi kullanılabilir:

```
if(hareket= =true) {
  System.out.println("Araba hareket halindedir.");
 }
else {
 System.out.println("Araba durmaktadır.");
 }
```

Switch ifadesi ise sadece sonucu nümerik olacak değerlerin karşılaştırılmasında kullanılabilir. If ile arasındaki en önemli fark da budur. Kullanım şekli şöyledir:

```
switch(x) {
        case 1:
                System.out.println("x değeri 1 dir");
                break;
        case 2:
                System.out.println("x değeri 2 dir");
                break;
        default:
                System.out.println("x değeri 1 veya 2 değildir.");
                break;}
```

4.8 JDBC (Java Database Connectivty)

Güçlü, platformdan bağımsız uygulamalar oluşturma yeteneği ve Web tabanlı applet' ler geliştiricileri Java' yı kullanarak ön-uç bağlanırlık çözümleri geliştirmek konusunda harekete geçirdi. Başlangıçta üçüncü parti yazılım geliştiricileri uygun çözümler sağlayarak, istemci tarafındaki kütüphaneleri birleştirmek için doğal yöntemler kullanarak veya yeni bir protokol ve bir üçüncü kat oluşturarak ihtiyacı karşılamıştır. Böylece Sun' ın JavaSoft Bölümü de Java veritabanı spesifikasyonunu 1996 da piyasaya sürmüştür. (Heller, Seymour, vd., 1998)

JDBC uygulama geliştiricilerine, uniform ve veritabanından bağımsız bir single API sağlar. API, kod yazmak için bir standart ve tüm değişik uygulama tasarımlarını hesaba katan bir standart sağlamaktadır. İşlemin özünde, bir sürücü tarafından gerçekleştirilen bir Java arabirimler seti yatmaktadır. Sürücü standart JDBC çağrılarının, sürücünün desteklediği veritabanı tarafından ihtiyaç duyulan özel çağrılara dönüştürülmesini ele alır. Bunun devamında, uygulama bir kez yazılır ve değişik sürücülere taşınır. Uygulama aynı olarak kalır; sürücüler değişir. (Heller, Seymour, vd., 1998)

4.8.1 JDBC Sürücüleri

İçinde Oracle ve IBM' in de bulunmakta olduğu birçok önemli veritabanı üreticisi JDBC sürücülerini veritabanı ürünleri ile birlikte pazarlamaktadır. Bu sürücüler kişisel RDBMS ler içerisinde birtakım özel karakteristikleri içermekle birlikte jenerik spektrumlarda bulunan karşı platform ilişkisini sağlamaktadır. Örneğin, Oracle 8 ürünü JDBC yi Oracle bağlantılarını güçlendirmek amacıyla kullanabilir. (Joch, 1998)

JDBC sürücüleri yapılarına göre dört kategoride toplanmaktadır:

1- JDBC-ODBC köprüsü sürücüsü

Java istemcisinin ODBC veritabanı servisine bağlanması için bir köprü kullanırlar. Sun' ın JDBC-ODBC sürücüsü tip 1 sürücülerin en yaygınlarındandır. Bu sürücüler doğal kodlar kullanılarak tanımlanmaktadırlar. (Hunter, Crawford, 1998)

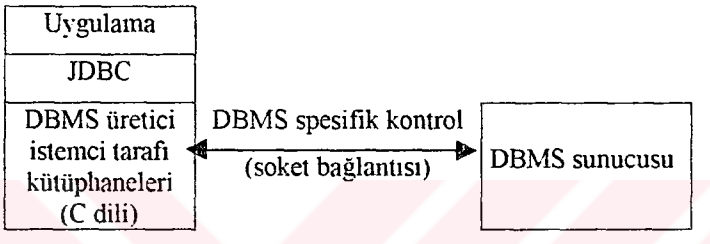
Bir JDBC-ODBC köprüsü, ODBC ikili kodu ve bazı durumlarda da bir istemci kütüphanesi ile gerçekleştirilmektedir. Köprü sürücüsü üç kısımdan oluşmaktadır: JDBC' yi ODBC sürücü yöneticisine bağlayan C kütüphaneleri takımı; ODBC sürücü yöneticisi ve ODBC sürücüsü.

Şekil 4-3: JDBC-ODBC köprüsü

Geliştiricinin perspektifinden bakıldığında, bir JDBC-ODBC köprüsünün kullanılması kolay bir seçimdir. Uygulamalar yine doğrudan JDBC sınıfları ile haberleşecektir. Ancak bir JDBC-ODBC köprüsü uygulaması, geliştiricinin uygulamayı çalıştırmak için neye ihtiyaç duyulduğunu bilmesini gerektirmektedir. Ayrıca ODBC çağrıları ikili C çağrıları kullanılarak yapıldığından, istemci ODBC sürücüsünün, ODBC sürücü yöneticisinin ve istemci tarafındaki yerel kütüphanelerin yerel bir kopyasına sahip olmalıdır. (Heller, Seymour, vd., 1998)

2- Native-API (Doğal kütüphaneden) Kısmi-Java sürücüsü

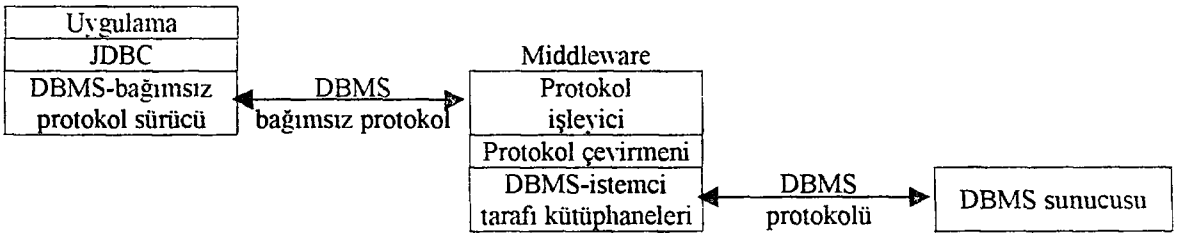
Java' nın ince bir tabaka ile çevrilmiş yüzeyini kullanarak veritabanı bağlantılı yerel kod kütüphanelerini sararlar. Bu tip sürücüler yerel kodları kullanarak uygulandıkları için bazı durumlarda tam bir Java uygulamasından daha iyi bir performans ortaya koyabilmektedirler. Bu beraberinde bir risk getirmektedir. Sürücü kodundaki herhangi bir arıza sunucuyu çökertebilmektedir. (Hunter, Crawford, 1998)



Şekil 4-4: Doğal kütüphaneden Java sürücüsü

3- Net-Protocol All-Java (DBMS-bağımsız ağ protokol) sürücüsü

Ağ protokolü sürücüsüdür. JDBC çağrıları bu sürücü tarafından DBMS' den bağımsız bir protokole dönüştürülmekte ve bir soket üzerinden orta-kat (middleware) sunucusuna gönderilmektedir. Orta kat kodu, istemci adına değişik veritabanları ile bağlantı kurmaktadır. Bu yaklaşım ayrıca, firewall' lar aracılığı ile veri gönderilmesini içeren ağ güvenliği ile ilgili konuları da ele almaktadır. (Heller, Seymour, vd., 1998)

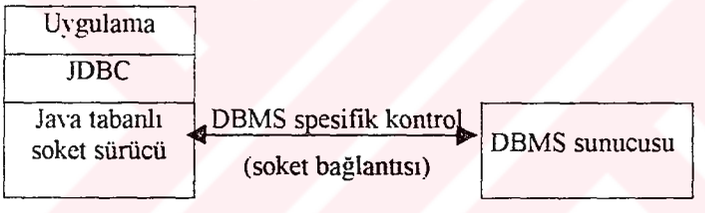


Şekil 4-5: DBMS-bağımsız ağ protokol sürücüsü

4- Native-Protocol All-Java sürücüsü

Tamamen Java ile yazılmış sürücülerdir. Bununla birlikte veritabanı kaynaklı ağ protokollerini anlayabilmektedirler ve herhangi bir ek yazılım olmadan direkt olarak veritabanına girebilmektedirler. (Hunter, Crawford, 1998)

Bu tip sürücülerde, JDBC çağrıları doğrudan DBMS sunucusu tarafından kullanılmakta olan ağ protokolüne dönüştürülmektedir. Bu sürücü senaryosunda, veritabanı sağlayıcısı bir ağ soketini desteklemekte ve JDBC bir soket bağlantısı üzerinden doğrudan veritabanı sunucusu ile haberleşmektedir. İstemci tarafındaki kod Java ile yazılabilmektedir. Bu çözüm uygulanması en basit olanıdır. Intranet kullanımı içinde oldukça pratiktir. (Heller, Seymour, vd., 1998)



Şekil 4-6: Doğal protokol Java sürücüsü

Üçüncü parti sürücü satıcıları bu dört sürücü çeşidinden birini kurarlar. Bu spektrumda bahsedilmeyen birtakım farklılıkları da göz önüne alarak onlara eklerler. Böyle bir basit açıklama da sürücü satıcıları kendilerini JDBC nin modüllerine daha uygun olarak kullanılabilecek sürücüler yazarak diğerlerinden ayırırlar. JDBC uygunluk testlerini uygularlar. (Joch, 1998)

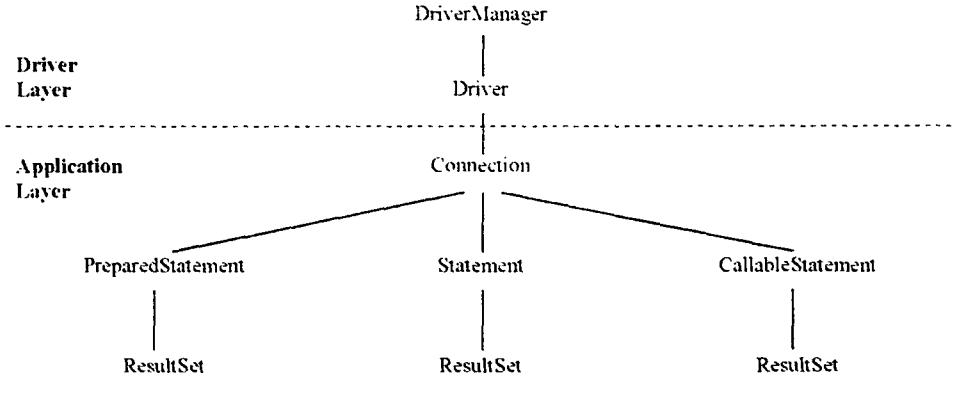
Sürücü üreticileri açısından gelecek 3. ve 4. tip de yatmaktadır ki bunlar uygulama geliştiriciler için bir takım faydalar sunmaktadırlar. Bu geliştiriciler de daha iyi işleyen RDBMS uygulamaları yaratmak isteyen kişilerdir. 3. tip sürücüler ki bunlar bir uygulama sunucusunda yer almaktadır ve tüm Java istemcileri ile

haberleşebilmektedir. Bu da Java uygulamalarına RDBMS ye ağ üzerinde bilgi okuma kullanma özelliği sağlar. 3. tip sürücüler bütün Java istemcileri ile uyumlu çalıştığı için yüklenmesi gereken C kütüphaneleri yoktur. Bu da 3. tipi tarayıcı tabanlı uygulamalar da en etkin kılan durumu oluşturur. Ticari olarak adlandırılan 3. tip sürücüleri değerlendirirken jenerik spektrumunun nereye kadar uzantı sağladığına dikkat etmek gerekmektedir. Örneğin bazı sürücüler IIOP yi desteklemektedir. Burada sorgulamalar yapmayı ve veritabanı bağlantılarını toplamayı desteklemektedirler.

4. tip sürücüler ise 2. tip gibi aynı uygulamaları sağlamakla birlikte Tamamen Java olma avantajını sağlarlar. (Joch, 1998)

4.9 JDBC API' sini Oluşturan Bileşenler

Bir SQL-Level API olan JDBC iki ana kısımdan meydana gelmektedir. Bunlar uygulama ve sürücü katmanlarıdır. API' nin kendisi arabirimini ayarlamakta ve sınıfları buna göre dizayn etmektedir. Şekil 4-7' de bu katmanlar gösterilmektedir: (Hunter, Crawford, 1998)



Şekil4-7: JDBC API bileşenleri

4.9.1 Sürücü Katmanını meydana getiren nesneler

DriverManager sınıfı

DriverManager sınıfı, sürücüleri yüklemek, geri yüklemekten ve sürücüler vasıtasıyla bağlantılar yapmaktan sorumludur. Bu sınıf ayrıca logging ve veritabanı oturum aşım süreleri ile ilgili özellikleri sağlar. (Heller, Seymour, vd., 1998)

Bir JDBC sürücüsü kullanılarak veritabanına bağlantı kurmanın ilk adımı geçerli sürücü sınıfının uygulama içinden JVM' ye yüklenmesini gerektirmektedir. Bu daha sonra bir bağlantı açıldığında sürücünün ulaşılabilir olmasını sağlamaktadır. Sürücü sınıfının yüklenmesinin en kolay yolu Class.forName() metodunun kullanılmasıdır:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

Sürücü hafızaya yüklendiğinde kendi kendisini java.sql.DriverManager sınıfına geçerli bir veritabanı sürücüsü gibi kaydeder. Hunter, Crawford, 1998)

Daha sonraki adım Drivermanager sınıfını kullanılarak veritabanına bağlantı açmaktadır. Bu bağlantıyı açmak için kullanılan metot DriverManager.getConnection() metodudur.

```
Connection con=DriverManager.getConnection("jdbc:odbc:somedb", "user", "passwd");
```

JDBC de tüm veritabanı bağlantıları bir TCP/IP ağı üzerinde meydana gelmektedir. Java' da birşeyleri yönetmenin en kolay yolu tüm veritabanı bağlantılarının URL ler şeklinde tanımlanmasıdır. (Hoobs, 1997)

JavaSoft sürücü URL' leri için aşağıdaki amaçları tanımlamaktadır:

- Sürücü-erişim URL adı kullanılmakta olan veritabanının tipini tanımlamalıdır.
- Kullanıcı (uygulama geliştiricisi) bir veritabanı oluşturulmasının yönetiminden sorumlu olmamalıdır; bu nedenle herhangi bir veritabanı bağlantı bilgisi (makine, port, veritabanı ismi, kullanıcı erişimi ve şifreler) URL içinde kodlanmalıdır.
- Kullanıcının özellikle doğru makine adını ve veritabanının port numarasını kodlamak zorunda kalmasını önlemek için bir ağ adlandırma sistemi kullanılabilir. (Heller, Seymour, vd., 1998)

Yukarıdaki açıklamalara göre bir JDBC URL si aşağıdaki gibi bir yapıya sahip olacaktır:

jdbc:<subprotocol>:<subname>

Bu yapıya göre <subprotocol> deyimi aşağıdaki bilgilerden birini içermektedir:

Bileşen	**Açıklaması**
IP Adresi	Uygulama sunucusuna ait IP adresi
Port	Uygulama sunucusunun iletişim kuracağı port numarası
ODBC motoru	Kullanılan ODBC arayüzü.
<subname>	Kullanılacak olan veritabanının ismini belirtir

Bu açıklamalara göre örnek bir URL tanımlamaları şu şekilde olacaktır:

- jdbc://localhost:8080/tez.mdb
- jdbc:oracle:products
- jdbc:odbc:tez.mdb

4.9.2 Uygulama Katmanı

Uygulama katmanı sürücü katmanında gerçekleştirilen, ancak uygulama geliştiricisi tarafından kullanılan üç arabirim içerir. Java' da arabirim, özel bir nesneyi göstermek için genel bir ad kullanılmasına imkan sağlar. Bu genel ad, özel nesne sınıfları tarafından gerçekleştirilmesi gereken yöntemleri tanımlar.

Bu üç arabirim Connection, Statement ve ResultSet' dir. Bir Connection nesnesi. DriverManager.getConenction() yöntem çağrısı aracılığı ile sürücü uygulamasından elde edilir. Bu nesne bir kez gönderildiğinde, uygulama geliştiricisi veritabanına göndermek için bir Statement nesnesi oluşturabilir. Statement' in sonucu ise belirli bir deyimin sonuçlarını içeren bir ResultSet nesnesidir. (Heller, Seymour, vd., 1998)

Connection arabirimi

JDBC Connection arabirimi, to handle processing, SQL ifadelerinin ve depolanmış işlemlerin icra edeceği metotları sağlamaktadır. Ayrıca bazı hata yakalama metotlarını da içermektedir.

Connection arabirimi direkt olarak kod ile yaratılan bir eleman değildir. Bu arabirim Driver.connect() metodunun yaratılması ile meydana gelmektedir. Bir connect() metodu çağrıldığında geriye aşağıdaki örnek program parçasında da görüldüğü gibi bir Connection nesnesi döner: (Hoobs, 1997)

```
try
{
        String driverName="sun.jdbc.odbc.JdbcOdbcDriver";
        Driver driver=(Driver)Class.forName(driverName).newInstance();
        String url="jdbc:odbc:tez.mdb";
        Connection
        con=DriverManager.getConnection(url,"cuhadar","cuhadar");
```

Statement Arabirimi

Deyimler (Statement), veritabanına SQL sorgulamaları göndermek ve bir sonuç setine erişmek için kullanılan araçtır. Deyimler, SQL güncelleştirmeleri (update), eklemeler (insert), silme işlemleri (delete) ve sorgulamalar (select) ile olmaktadır. (Heller, Seymour, vd., 1998)

Stetement nesnesi de Connection nesnesi gibi direkt olarak yaratılmamaktadır. Bir statement nesnesi, diğer bir nesneye ait bir metodun geri dönen değeri atanarak meydana getirilmektedir. createStatement() metodu Statement nesnesine bir değer geri döndürmektedir. Örnek bir ifade aşağıda gösterilmektedir:

```
Statement stmt=con.createStatement();
```

Bir Statement nesnesi yalnızca statik SQL ifadelerini işletmek ve bu SQL sorgulamalarından yanıtları elde etmek amacıyla kullanılmaktadır. Ekleme, silme ve güncelleme gibi statik SQL ifadelerinin işletilmesi sonucu herhangi bir değer geri dönmez. Dinamik bir SQL ifadesinin işletilmesi için ise PreparedStatement adı verilen bir deyim kullanılmaktadır.

PreparedStatement Arabirimi

Dinamik SQL İfadeleri

PreparedStatement nesnesi içerisinde dinamik bir SQL ifadesi meydana getirmek için kullanılacak olan dinamik parametre yerine ? (soru işareti) gibi bir yer tutucu kullanılmaktadır. Soru işareti (?) ifade içerisinde kod içerisindeki tayin edilecek değerin yerini alacaktır. Aşağıdaki örnekte, tEmployee tablosu içerisinde bulunan soyadı ne olursa olsun tüm isimleri bir SQL select ifadesi işleterek geri döndürecek bir sözdizimi verilmektedir: (Hoobs, 1997)

```
Select FirstName  from tEmployee Where LastName = ?
```

Depolanmış Prosedürler

PreparedStatement olarak verilen bu nesne düzenli bir Statement nesnesi gibidir. SQL ifadelerini uygulayabilmek için kullanılabilinmektedir. Önemli farklarından biri, PreparedStatement içerisinde yer alan bir SQL ifadesinin daha hızlı bir şekilde uygulamaya girebilmesi için veritabanı tarafından önceden hazırlanmaktadır. PreparedStatement bir kez oluşturulduğunda önceden tanımlanmış parametreler eklenerek düzenlenmektedir. Hazırlanmış bu ifadeler aynı genel SQL komutlarının tekrar tekrar çalıştırılabilmesi açısından faydalıdırlar. Aşağıda PreparedStatement ifadesinin kullanımını gösteren bir program parçası gösterilmektedir. (Hunter, Crawford, 1998)

```
PreparedStatement pstmt = con. preparedStatement (
                "INSERT INTO ORDERS (ORDER_ID, COSTUMER_ID,
                TOTAL) VALUES (?, ?, ?)");
pstmt.clearParameters();
pstmt.setInt(1, 2);
```

```
pstmt.setInt(2, 4);
pstmt.setDouble(3, 53, 43);
pstmt.executeUpdate();
```

Parametreler, ya yeni bir setType yöntemi çağırılıncaya ya da PreparedStatement nesnesi için clearParameters() yöntemi çağırılıncaya dek o andaki değerlerini tutarlar. Statement arabiriminden miras alınan execute() yöntemlerine ek olarak, PreparedStatement aşağıda verilen setType yöntemlerini bildirmektedir. (Heller, Seymour, vd., 1998)

Tablo 4-2: Java için SQL tipleri

Yöntem İmzası	Java Tipi	Veritabanından Gönderilen SQL Tipi
void setByte (int index, byte b)	byte	TINYINT
void setShort (int index, short x)	short	SMALLINT
void setInt (int index, int i)	int	INTEGER
void setLong (int index, long l)	long	BIGINT
void setFloat (int index, float f)	float	FLOAT
void setDouble (int index, double d)	double	DOUBLE
void setBigdecimal (int index, bigdecimal x)	java.math.BigDecimal	NUMERIC
void setString (int index, string s)	java.lang.String	VARCHAR ya da LONGVARCHAR
void setBytes (int index, byte x[])	byte dizisi	VARBINARY ya da LONGVARBINARY
void setDate (int index, Date d)	java.sql.Date	DATE
void setTime (int index, Time t)	java.sql.Time	TIME
void setTimeStamp (int index, TimeStamp ts)	java.sql.Timestamp	TIMESTAMP
void setNull (int index, sqlType)	-	0
void setBoolean (int index, boolean b)	Boolean	BIT

ResultSet Arabirimi

ResultSet arabirimi, bir deyimin yürütülmesi sonucu olarak üretilen veri tablolarına erişmek için yöntemler tanımlamaktadır. ResultSet sütun değerlerine

herhangi bir sıra ile erişebilmektedir. Bunlar indekslenmiştir ve sütunun adı veya numarası ile seçilebilmektedir. (Heller, Seymour, vd., 1998)

Bu özelliğinden dolayı ResultSet metodu çok boyutlu dizilere benzetilebilir. Sütunlardan meydana gelen belli sayıda kayıt ya da satır vardır. Her bir sütun karakter, tamsayı gibi belli veriler içermektedir.

Veritabanına bağlantı yapıldıktan sonra cevaplara ulaşabilmek amacıyla SQL ifadeleri meydana getirilir. Aşağıdaki örnek ifade, tEmployee tablosundan FirstName (ad) ve LastName (soyad) alanlarındaki bilgiyi alacaktır:

```
Select FirstName, LastName from tEmployee
```

Daha sonraki adım bir bu SQL ifadesini işletecek bir Statement nesnesi yaratılmasıdır:

```
Statement sqlStatement = connection.createStatement();
```

Sonuç olarak ise SQL ifadesindeki kriterler ile eşleşen tüm kayıtlar ResultSet metodunda geri dönecektir:

```
String sql = "select FirstName, LastName from tEmployee";
ResultSet  employees = sqlStatement.executeQuery(sql);
```

Veritabanı sorgulaması sona erdiğinde yanıtların tamamı ResultSet nesnesine geri dönmektedir. Bu yanıtları tek tek değerlendirebilmek için ise next() ve getNNN() metotları kullanılmaktadır. next() metodu, ResultSet nesnesine ait pointer' ı (işaretçi) veritabanındaki bir sonraki kayda hareket ettirmektedir. Pointer' ın göstermiş olduğu kayıt geçerli ise değeri ture, diğer durumlarda ise false olacaktır. (Hoobs, 1997)

4.10 Java Servletleri

Sunucu taraflı Java uygulamalarının yükselişi Java ile programlamanın en son ve en heyecan verici gelişimidir. Java dili orijinal olarak küçük ve içerisine eklemeler yapılabilecek bir dil olarak ortaya çıkmıştır. Başlarda appletler şeklinde istemci-taraflı web içerikleri işlenebilen bir dildi. Şimdi ise Java sunucu-taraflı program geliştirme de ideal bir hale gelmiştir.

Java servletleri sunucu taraflı Java program geliştirmenin anahtar bir bileşenidir. Bir servlet sunucunun fonksiyonelliğini arttıran, sunucuya uzantısı olan küçük bir eklentidir. Servletler geliştiricilere web sunucusu, mail sunucusu, uygulama sunucuları, custom server gibi Java destekli sunuculara uzantı yaratmaya ve seçenekli kılmaya imkan tanır. Böylece kullanım, taşınabilirlik ve esneklik açısından kolaylıklar sağlanır. (Hunter, Crawford, 1998)

4.10.1 Servletlerin Gelişimi

HTML ve World Wide Web ilk keşfedildiğinde, görüntülenen her sayfanın içeriği esas olarak statikti. Daha fazla etkileşim ihtimaline ve dolayısıyla belli bir isteğe uyarlanmış sayfalara imkan vermek için CGI mekanizması ortaya konmuştur.

CGI bir URL' nin bir HTML sayfasına değil. ancak bir programa temel bir başvuru içermesine izin vermektedir. Bu temel başvuruya ek olarak, yürütmeyi kontrol etmek için kullanılan parametreler de tarayıcıdan CGI programına gönderilebilir.

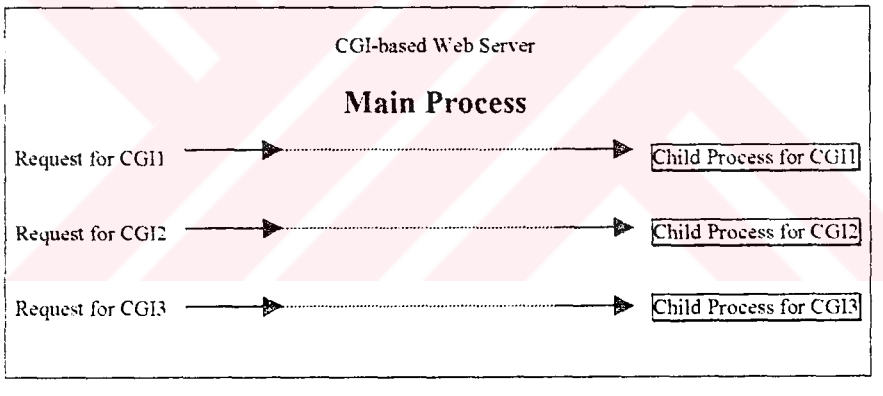
CGI kullanımı basit olmasına rağmen hem kullanıcı, hem de sistem yöneticisi açısından birkaç zayıf yönü bulunmaktadır:

1. CGI programları scriptler ve yorumlayıcı diller kullanarak yazılmaktadır. Derlenen dilerin kullanımı hızı arttırmakta fakat platform bağımlılığını da berberinde getirmektedir.

2. CGI programları ayrı süreçler olarak çalıştırılırlar. Bu da genellikle uzun başlangıç zamanları anlamına gelmektedir.

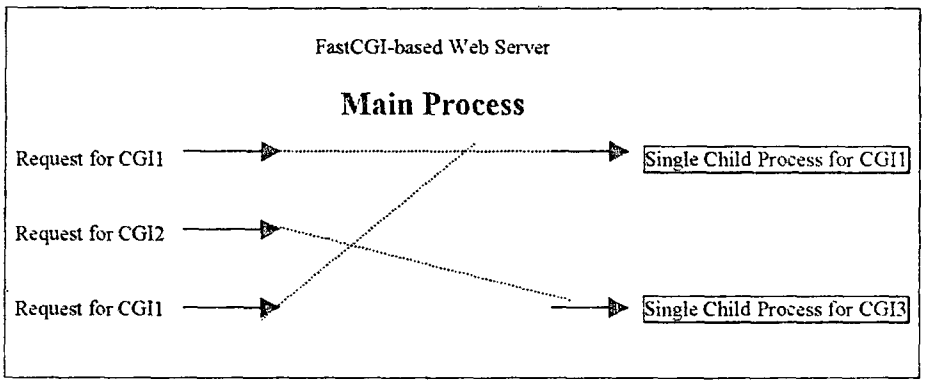
3. Her CGI programı ayrı süreçler olarak başlatıldığından çağrılar arası haberleşme dosyalar üzerinden yapılmaktadır. Bu durumda da iletişim yavaşlamaktadır. Aynı sunucu üzerindeki farklı CGI programları arasındaki iletişimde yine oldukça yavaştır.

4. Yeni standartlar ve PERL gibi daha güvenli diller kullanılsa dahi sistem temel bir güvenlik yapısına sahip değildir. (Heller, Seymour, vd., 1998)



Şekil 4-8: CGI yaşam çizgisi

Bu kısıtlamalardan dolayı yeni alternatifler aranmıştır. İsmi Open Market olan bir şirket standart CGI' a FastCGI adı verilen bir alternatif geliştirmiştir. Birçok yönden FastCGI, CGI gibi çalışmaktadır. Önemli fark, FastCGI' ın her bir FastCGI programı için tek bir işlem ortaya koymasıdır. (Hunter, Crawford, 1998)



Şekil 4-9: CGI yaşam çizgisi

FastCGI kalıcı süreçleri kullanarak süreç-başlangıç yükünü önler, ancak FastCGI kullanılırken de hala süreç içi haberleşme yavaştır. Bazı C dili API' leri programların sunucu içerisinde çalışmasını göz önüne alır. Fakat bu API' ler platforma bağlıdır ve güvenliği sağlama zordur. Ayrıca bu API' ler oldukça karmaşık olabilirler. (Heller, Seymour, vd., 1998)

Web tabanlı güçlü bir Internet uygulamasının aşağıdaki gereksinimleri yerine getirmesi beklenir: (Lubling, Malave, 1998)

1. PC, Unix veya Machintosh gibi farklı sistemler tarafından kullanılacağı düşünülerek istemci açısından platformdan bağımsız olması.
2. Windows NT ya da Unix gibi farklı sunucular üerinde yer alabileceği düşünülerek sunucu açısından da platformdan bağımsız olması.
3. Potansiyel veri kaynakları Oracle, Informix ve SQL sunucuları içermektedir. Bu durumda çeşitli veritabanlarını bağlayabilmelidir.
4. Çeşitli web sunucuları üzerinde çalışabilmelidir.
5. Ağ, web sunucusu, veritabanı birlikteliği için birkaç saniyeden fazla zamana ihtiyacı olmamalıdır.
6. Kolay kullanımlı ve düşük maliyetli olmalıdır.
7. Modüler olmalıdır.

Son olarak tüm bu gereksinimleri sağlayan en iyi seçeneğin servletler olduğu görülmektedir. Java servletleri sunucu açısından JDBC' yi; istemci açısından da HTML ve Java scriptleri içermektedir. Servletler ve JDBC geliştiricilere platformdan bağımsız çalışabilme özelliği vermektedir. Herhangi bir web sunucusunda çalışabilir ve esnek veri tabanı birleştiriciliğini sağlamaktadır. (Lubling, Malave, 1998)

4.10.2 Servletlerin Özellikleri

Servletler, geleneksel servisleri kullanmaya imkan veren java destekli bir web sunucusu içerisine yerleştirilmiş Java bileşenleridir. İstek/cevap (Request/Response) işlemine göre çalışmak üzere dizayn edilmişlerdir. Bu modelde istemci sunucuya bir istek göndermekte; sunucu ise bu isteğe bir cevap ile karşılık vermektedir. İstek, HTTP. URL, FTP ya da geleneksel diğer bir formda olabilmektedir. (Sun MS -Fundementals of Java Servlets: Course notes-)

Diğer dinamik içerikli web teknolojilerine alternatif olarak gösterilen servletler çok önemli özelliklere sahiptir. Servletlerin taşınabilirlik, güç, entegrasyon, emniyet, dayanıklılık gibi özellikler sayesinde diğer yaklaşımlardan daha iyi bir seçenek olduğu düşünülmektedir.

Java ile yazıldığı için servletler birçok işletim sistemi ve sunucu üzerinde taşınabilirler. Ağ kurma, URL girişi, çoklu kanal oluşturma, veri sıkıştırması, resim manipilasyonu, veritabanı bağlantısı gibi Java API' lerinin oluşturduğu tüm güçleri kontrol altına alabilmektedir.

Servletler bir kez yüklendiklerinde sunucunun hafızasında tek nesne örneği olarak kalmaktadırlar. Bundan sonra da sunucu servleti basit bir yazım kullanarak isteği yerine getirmesi için uyarır. CGI' ın tersine burada ortaya bir yorumlayıcı çıkmamaktadır. Böylece servlet isteği hemen işleme koyar. Çoklu ve aynı anda gelen

istekler farklı ağlar tarafından ele alınır ve böylece de servletler yüksek derecede ölçeklendirilebilir.

Birçok aşamada servletler emniyetli uygulamaları desteklemektedirler. Çünkü servletler Java ile yazılmakta ve bu dilin en güçlü güvenlik çeşidini kullanmaktırlar. Java' da pointer' ların olmayışı servletlerin sorunlu pointer' lar, geçersiz pointer referansları ve hafıza eksikliği gibi sorunlar konusunda emniyetli olduğu durumunu ortaya çıkarmaktadır.

Servletler hataları da emniyetli bir biçimde düzeltebilmektedirler. Örneğin, eğer bir servlet sıfır ile bölünürse ya da uygun olmayan bir işlem yerine getirirse çok çabuk bir şekilde sorunlu kısım ortaya çıkarılarak kullanıcıya bilgi verilir. Eğer C++ ile oluşturulmuş sunucu uzantıları bu işleme maruz kalsa çok büyük bir olasılıkla sistem çökecektir.

Sunucu ile yakından entegrasyon servletlerin bir başka önemli özelliğidir. Bu entegrasyon sunucunun bir CGI programının yapamayacağı şekilde servletler ile iletişim halinde olmasını sağlamaktadır. Örneğin, bir servlet sunucuyu dosya dizinlerini çevirme, giriş sağlama, kimlik bilgilerini kontrol etme, MIME tiplerini haritalama ve veritabanına kullanıcı ekleme gibi işlemleri yapabilmektedir.

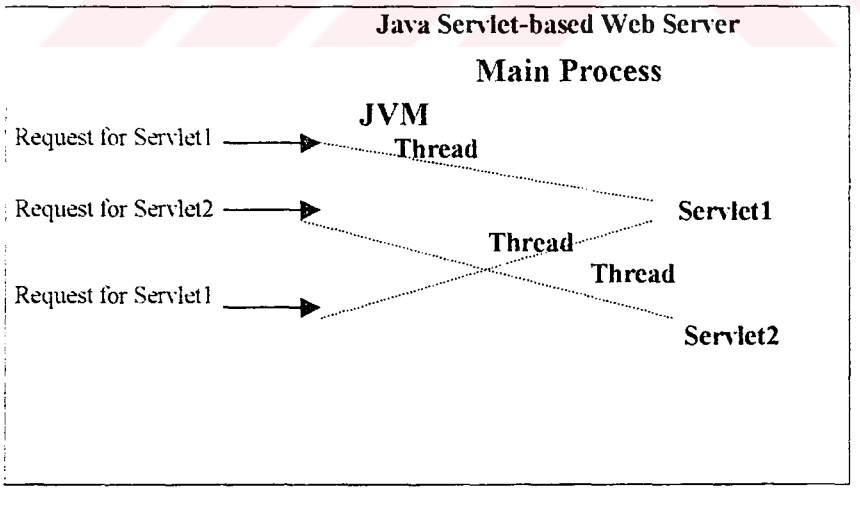
Servletler oldukça esnektirler. Bir HTTP servleti tamamen bir web sayfası oluşturmak için ya da diğer servletler ile birlikte bir servlet zinciri oluşturmak için kullanılabilmektedir. Buna ek olarak da Sun, servlet kodlarının statik HTML sayfaları içerisinde direkt olarak yazımını sağlayacak yöntem ya da parçaların kullanımını önermiştir. Bu da bir bakıma ASP (Active Server Pages) ile bir benzerlik göstermektedir. (Hunter, Crawford, 1998)

4.10.3 Servletlerin Yaşam Döngüsü

Servletler yaşam döngüleri boyunca CGI' ların hem performansına hem de kaynak sorunlarına cevap verebilmek amacıyla kullanılırlar. Bütün servletler JVM (Java Virtual Machine) içerisinde çalışabilmekte ve etkin olarak veri paylaşabilmektedirler. Öte yandan Java dili tarafından birbirlerinin özel bilgilerine girmeleri engellenmektedir. Servletler nesne örnekleri olarak istekler arasında süreklilik sağlayabilmektedirler. Böylece hafızada çok az bir yer kaplamaktadırlar.

Sunucu servletleri desteklemek konusunda belli bir çizgiye sahiptir. Buradaki en güç ve en hızlı kural servlet makinesinin aşağıdaki yaşam çizgisi bağlantılarına uyum sağlamasıdır: (Hunter, Crawford, 1998)

1. Servletlerin yaratılması ve harekete geçirilmesi
2. Sıfır ve üzeri sayıda istemciden gelen hizmet çağrılarını değerlendirebilmesi
3. Servletin ortadan kaldırılması



Şekil 4-10: Servlet yaşam döngüsü

Bir web sunucusu servletler ile javax.servlet.Servlet şeklinde gösterilen basit bir arabirim aracılığı ile iletişim kurar. Bu arabirim şu üç temel metodu içermektedir: (Sun MS -Fundementals of Java Servlets: Course notes-)

- init()
- service()
- destroy()

Yüzeysel olarak bakıldığında, servletler applet' lara oldukça benzemektedir. Bir uygulama gibi çalıştırılamaz ve veya statik bir main() yönteminden başlatılamazlar. Bunun yerine yüklenirler ve bir örnekleri oluşturulur. Örnek mevcut olduğunda, çağrılırken kullanılan parametreler gibi detayları belirleyebileceği bir ortam verilir.

Bir applet çağrıldığında davranış biçimi büyük oranda tarayıcı tarafından çağrılan birkaç yöntem ile belirlenir. Bu yöntemler; init(), destroy(), start(), stop() ve paint()' dir. Bir servlet içinde aynı temel yapı kullanılmaktadır. Fakat yöntemler biraz daha farklılaşmaktadır.

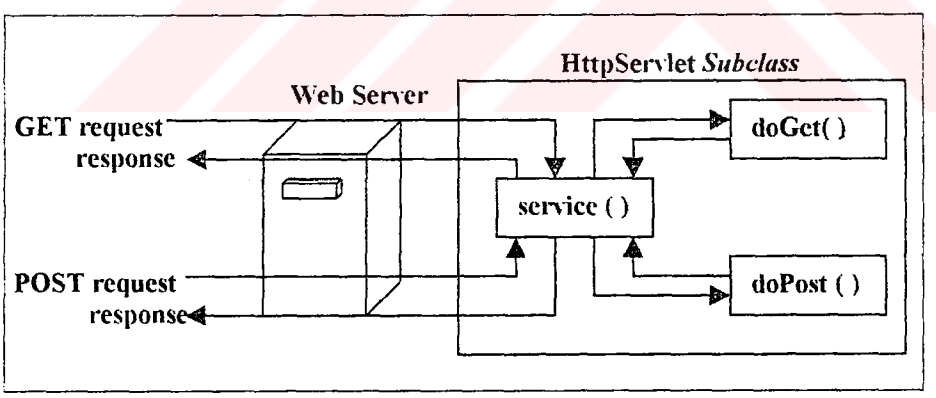
Bir servlet örneği oluşturulduğunda, bu örnek init() yöntemi ile çağrılmaktadır. Bu davranış bir applet' in yaşam çevrimi ile paralellik göstermektedir ve servlet' in kendisini başlangıç durumuna getirmesi amaçlanır. Heller, Seymour, vd., 1998)

init () Metodu isteklerin cevaplandırılmasında nesnelerin yüklenmesi ve yaratılması için kullanılmaktadır. Bir servletin kendi hakkında bilgi sağlayabilmesi açısından sunucu init () metodunu harekete geçirmekte ve ServletConfig adı verilen bir metodu çağırmaktadırlar. Bu nesne servleti -örneğin bir counter' ın (sayaç) hangi değerden başlayacağı gibi- başlangıç parametreleri hakkında bilgilendirmektedir. (Hunter, Crawford, 1998)

Bir sayfaya ya da sayfa dışına hareket edildiğinde ise servlet' in bir applet' in yaptığı gibi aktif veya pasif hale gelmesi gerekmez. Ayrıca bir servlet' in kullanıcı arayüzü (Graphical User Interface) yoktur. Bu nedenle start(), stop() ve paint() yöntemlerini de içermemektedirler. Servlet' in temel davranışı sunucudaki yeni bir bağlantıya cevap olarak gereklidir. Bu da servlet' in service() yöntemine yapılan bir çağrı ile sonuçlanmaktadır. (Heller, Seymour, vd., 1998)

Bir service() metodu, request (istek) ve response (cevap) şeklinde iki adet parametre kabul etmektedir. Bunlardan request nesnesi servlete isteği bildirirken, response nesnesi de bir cevabın geri döndürülmesinde kullanılmaktadır.

Bir HTTP servleti bazen service() metoduna çok fazla önem vermemektedir. Buna karşılık olarak, ihtiyaç duyulan isteğin türüne göre GET için doGet(), POST için ise doPost() metotları kullanılmaktadır. Aşağıdaki şekilde bir servlet için bu iki metodun kullanımı gösterilmektedir:



Şekil 4-11: Bir HTTP servletinin işlem tarzı

Son olarak ise servlet geri yüklenirken açık dosyalar ya da veritabanı bağlantıları gibi kaynakların temizlenmesi için destroy () metodu kullanılmaktadır. Eğer hafızada temizlenmesi gereken herhangi bir kaynak yoksa bu metodun içi boş olabilmektedir. (Sun MS -Fundementals of Java Servlets: Course notes-)

4.11 Servlet - Applet İlişkisi

Java' nın ortaya çıkışından beri, applet' ler web sayfalarının sadece uyarlanmış bilgilere değil, aynı zamanda gerçekten interaktif ve dinamik bir içeriğe sahip olması için de bir mekanizma sağlamaktadır. Bir applet ile süreç tarayıcının kendi makinesi üzerinde çalışmaktadır. Genelde bu konfigürasyon bir avantajdır. Çünkü, cevap süresini iyileştirmekte ve ağ-bant genişliği gereksinimlerini azaltmaktadır. Fakat, eğer program aşağıdaki gereksinimlerden herhangi birine sahipse, bu konfigürasyon sorunlara neden olabilmektedir:

- **Sunucu imkanlarına ayrıcalıklı erişim**
  Bir applet genellikle bir sunucu üzerindeki bilgi ve hizmetlere özel bir erişime sahip değildir. Hatta applet' i sağlayan sunucu bile güvenmek isteyeceği bir applet' den gelen istek ile bir başka isteği birbirinden ayıramamaktadır. Dolayısıyla, bir applet' in örneğin, sunucudaki bir veritabanı okuma yazma izni, bir HTTP isteğine bu veritabanına tam erişim hakkı verilmediği sürece kabul edilmemektedir.

- **Özel algoritmaların korunması**
  Birkaç yolla Java byte kodunun geriye analiz edilmesi diğer makine dillerinden daha kolay gerçekleşmektedir. Bu kısmen geçerlidir, çünkü karmaşık byte kodları üretmek zordur. Byte kodu doğrulayıcısının gereksinimleri, açık olmayan birçok kod biçimini geçerli olmadığı gerekçesiyle ret edecektir. Bu nedenden ötürü önemli değere sahip bir özel algoritma genel olarak bir applet' e emanet edilmemelidir.

Bu iki durumdan biri söz konusu ise bir servlet daha iyi bir seçim olacaktır. Servlet yaklaşımı ile, sunucu, veritabanları gibi yerel imkanlara tam erişime izin

verebilmektedir ve servlet' in dış kullanıcılara sağlanan erişimin boyutlarını kontrol edebileceğine güvenebilir. Böylelikle örnek olarak, isteklerin yapılacağı oran sınırlandırılabilmektedir. Ayrıca isteklerin kaynağı izlenebilmekte ve doğrulanabilmektedir. Bir servlet' e özel bir algoritma yerleştirilirse, kod asla sunucunun sınırlarının ötesine geçememektedir. Sadece kodun üretmiş olduğu sonuçlar bunu gerçekleştirebilmektedir.

Tüm bu sınırlılıkların yanında bir servlet ve bir applet hem servlet' lerin avantajlarından, hem de applet' lerin etkileşimli doğasından faydalanabilmek amacıyla birlikte kullanılabilmektedirler. Bu birleştirilmiş yapı yaklaşımı, gerçek sıkıştırma ve şifrelemeyi de içeren veri akımının optimizasyonunu sağlamak için de kullanılabilmektedir. (Heller, Seymour, vd., 1998)

4.12 HTTP Desteği

Servletler çok yaygın olan HTTP protokolünü kullanmaktadırlar. Bu protokol web tarayıcıları ve sunucular arasındaki iletişimi sağlamak üzere tanımlanmıştır. Servlet uygulamalarında kullanılan HTTP metotları GET, HEAD ve POST' dur.

4.12.1 GET Metodu

GET metodu bir web sunucusundan istek bilgisini almak için kullanılmaktadır. Bu bilgi bir dosya, sunucu üzerindeki bir cihazdan ya da bir programdan çıktı olabilmektedir. Aşağıda servlet içerisinde HTTP Get metodu kullanımı ve MyServlet servletine yapılan bir istek görülmektedir:

Metodun kullanımı:

```
public void doGet (HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
```

Yapılan istek:

```
GET /servlet/MyServlet?name=Scott&
    company=MageLang%20Institute  HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (
    compatible;
    MSIE 4.01;
    Windows NT)
Host: www.magelang.com
Accept: image/gif, image/x-xbitmap,
    image/jpeg, image/pjpeg
```

Bu örnekte isim ve şirket ismi şeklinde iki parametre bulunmaktadır. Parametreler, "name=değer" formatına göre düzenlenmektedir. Değerler (&) işareti ile birbirinden ayrılmakta ve (?) işareti ile de servlete gönderilmektedir. (Sun MS -Fundementals of Java Servlets: Course notes-)

Basit olmasına rağmen GET yönteminin bazı teknik sorunları bulunmaktadır. Örneğin çok uzun bir argüman listesine sahip bir istek tarafından web sunucusunun güvenliği tehlikeye sokulabilir. Bu olasılıktan dolayı diğer bir metot olan POST kullanılmaktadır. (Heller, Seymour, vd., 1998)

4.12.2 POST Metodu

Bir HTTP POST isteği de GET gibi bir istemciden sunucuya veri göndermek amacıyla kullanılmaktadır. Bu istek haber guruplarına (newsgroup) bilgi aktarımı, web sitesinin misafir defterine kayıt ekleme gibi amaçlar doğrultusunda yapılabilmektedir. Bazı web sunucuları isteğe ait URL' nin belli miktarının kabul edebileceğine dair bir limite sahiptir olmaktadırlar. POST metodu bu limiti aşarak tüm parametrelerdeki tüm veriyi bir giriş akımı şeklinde aktarabilmektedir. Tipik bir POST isteği ve servlet içersindeki kod satırları aşağıda görüldüğü gibi olacaktır: (Sun MS -Fundementals of Java Servlets: Course notes-)

```
public void doPost (HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
{
                doGet (req, res);
}
```

```
POST /servlet/MyServlet  HTTP/1.1
User-Agent: Mozilla/4.0 (
        compatible;
        MSIE 4.01;
        Windows NT)
Host: www.magelang.com
Accept: image/gif, image/x-xbitmap.
        image/jpeg, image/pjpeg
Content-type: application/x-www-form-urlencoded
```

Content-length: 39

name = Scott&company=MageLang%20Institute

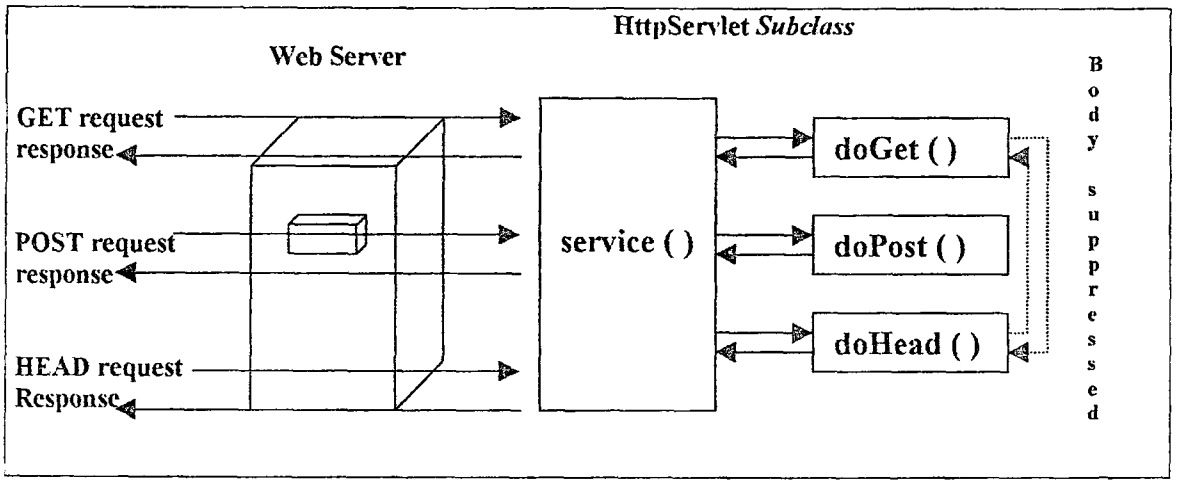
Post mekanizmasında URL, bağlantıyı başlatmak için kullanılır. Bundan sonra ise iki adım gerçekleşmektedir. Bunlardan birincisi, parametrelerin web sunucusuna iletilmesi; diğer ise isteğin sonuçlarını okumak için kullanılan giriş akımıdır. (Heller, Seymour, vd., 1998)

4.12.3 HEAD Metodu

Head metodu Get metoduna oldukça benzemektedir. İsteğin özellikleri Get metodu ile tamamen aynıdır. Fakat sunucu sadece başlık bilgilerini geri döndürmektedir. Head metodunun en sık kullanıldığı durumlar aşağıdaki gibidir: (Sun MS -Fundementals of Java Servlets: Course notes-)

- Bir dökümanın son değişitirilme tarihi.
- Bir dosyanın download (dosya indirme) edilmeden önceki boyutu.
- İstemcinin istekte bulunacağı sunucunun tipi.
- İstemcinin desteklediğinden emin olabilmek için istek dokümanının tipi.

Aşağıdaki şekilde GET, POST ve HEAD isteklerinin kullanım gösterilmektedir: (Hunter, Crawford, 1998)



Şekil 4-12: Temel HTTP isteklerinin kullanımı

4.13 Servletlere İstek (Request) Bilgilerinin Aktarımı

Başarılı bir web uygulaması yapabilmek için, uygulamanın çalıştığı sunucu ve istek gönderen istemciler hakkında da bilgi sahibi olmak gerekmektedir. Servletler bu bilgilere ulaşabilmek amacıyla bir takım teknikler kullanmaktadılrlar. Kullanılan metotları CGI programcıklarının bilgiye ulaşırken kullandığı tekniklerle karşılaştırdığımızda servletlerin bir takım avantajları olduğu görülmektedir:

- Daha güçlü kontrol mekanizması: CGI programcıkları ortam değişkenlerini çekmek için tek bir fonksiyon kullanmaktadırlar. Bu durumda çalışma zamanı (run-time) problemleri çıkana kadar fazla bir problem bulunmaz. Bunun için bir CGI programı ve servletin çalıştığı sunucuya ait port numarasını nasıl bulduğuna dair birer örnek aşağıda verilmektedir:

C dili ile yazılmış bir CGI scripti şöyledir:

```
char *port=getenv{"SERVER_PORT"};
```

Portun karakter zinciri ile temsil edildiği bu durumda rastlantısal hataların olması yüksekdir. Veri, ortam değişkeni ile gönderilen veri ile uyuşmayabilmektedir. Diğer yanda bu işlem bir servletde ise şöyle tanımlanabilir:

```
int port = req.getServerPort()
```

Servletlerde bu şekilde bir tanımlama bir çok rastlantısal hatayı önlemektedir. Çünkü derleyici herhangi bir yazım yanlışı olmadığını ve her dönüş tipinin doğruluğunu temin etmektedir.

- Gecikmiş heasplama: Sunucu bir CGI programı başlattığında her bir ortam değişkeni önceden hesaplanmakta ve işletilmektedir. Servleti başlatan bir sunucu bu hesaplamaları ihtiyaç olduğunda çalıştırmak suretiyle performansı arttırmaktadır.

- Sunucu ile daha fazla etkileşim: CGI programı işleme başladığında sunucudan ayrılmaktadır. Bu durumda program ile tek iletişim yolu standart çıktıdır. Servletlerde ise sunucu ile birlikte çalışabilmektedir. Bu bağlantı kullanılarak servlet sadece sunucunun sağlayabileceği hesaplanmış bilgi için ilgili istekler kullanmkatadır. (Hunter, Crawford, 1998)

Tablo 4-3: CGI ortam değişkenleri ve servlet metotlarının karşılaştırılması.

CGI Çevre Değişkeni	HTTP Servlet Metodu
SERVER_NAME	req.getServerName()
SERVER_SOFTWARE	getServletContext().getservletInfo()
SERVER_PROTOCOL	req.getProtocol()
SERVER_PORT	req.getServerPort()
REQUEST_METHOD	req.getMethod()
PATH_INFO	req.getPathInfo()
PATH_TRANSLATED	req.getPathTranslated()
SCRIPT_NAME	req.getServletPath()
DOCUMENT_ROOT	req.getRealPath("/")
QUERY_STRING	req.getQueryString()
REMOTE_HOST	req.getRemoteHost()
REMOTE_ADDR	req.getRemoteAddr()
AUTH_TYPE	req.getauthType()
REMOTE_USER	req.getRemoteUser()
CONTENT_TYPE	req.getContentType()
CONTENT_LENGTH	req.getContentLength()
HTTP_ACCEPT	req.getHeader("Accept")
HTTP_USER_AGENT	req.getHeader("User-Agent")
HTTP_REFERER	req.getHeader("Referer")

4.14 Servlet' ler Arası Haberleşme

Standart CGI' ın sorunlarından birisi de haberleşme ile ilgilidir. Geleneksel CGI mekanizmaları ile farklı CGI scriptleri arasındaki haberleşme oldukça yavaştır. Sunucu, bir servleti başlattığında, servlet sonraki tüm isteklere hizmet etmek için varlığını sürdürecektir. Ayrıca servletlerin tümü aynı sanal makine (Virtual Machine) içerisinde bulunmaktadır. Bu iki davranış biçimi farklı servletlerin etkin ve rahat biçimde haberleşmesini mümkün kılmaktadır. (Heller, Seymour, vd., 1998)

Genel olarak değişik tipteki programların haberleşmesinde kullanılan yöntemler Servlet' ler arasındaki haberleşme için de kullanılabilmektedir:

1- Birbirinin yöntemlerini çağırma: Eğer iki servlet aynı anda yazılmış ve derlendiklerinde birbirlerinin API' sini bilebiliyorlarsa birbirlerinin yöntemlerini çağırabilmektedirler.

2- Akım sınıflarının kullanılması: PipedInputStream ve PipedOutput sınıflarının kullanılması ile gerçekleşen bir yöntemdir. Böyle bir haberleşmede her servlet diğerinin akımları üzerinde bir işleyici elde edebilmelidir. Bu yaklaşım iki yönlü haberleşme için de uygun olabilmektedir.

3- Statik değişkenlerin kullanılması: Sınıflar arası haberleşmenin temeli olarak statik değişkenler kullanılabilmektedir. Örneğin, giriş-çıkış akımları kullanılarak, uygulama çıkış akımına erişim sağlamak amacıyla bir statik değişken kullanacak şekilde düzenlenebilmektedir.

4.15 Basit Bir Servlet Örneği

HTTP servlet' lerinin en temel şekli bir HTML sayfası şeklinde meydana getirilmeleridir. Buna örnek olarak aşağıda program kodu gösterilmekte olan servlet verilebilir:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

  public void doGet (HttpServletRequest req, HttpServletResponse res)
                    throws ServletException, IOException {

    res.SetContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Hello World</TITLE></HEAD>");
```

```
            out.println("<BODY>");
            out.println("HELLO WORLD");
            out.println("</BODY></HTML>");
        }
}
```

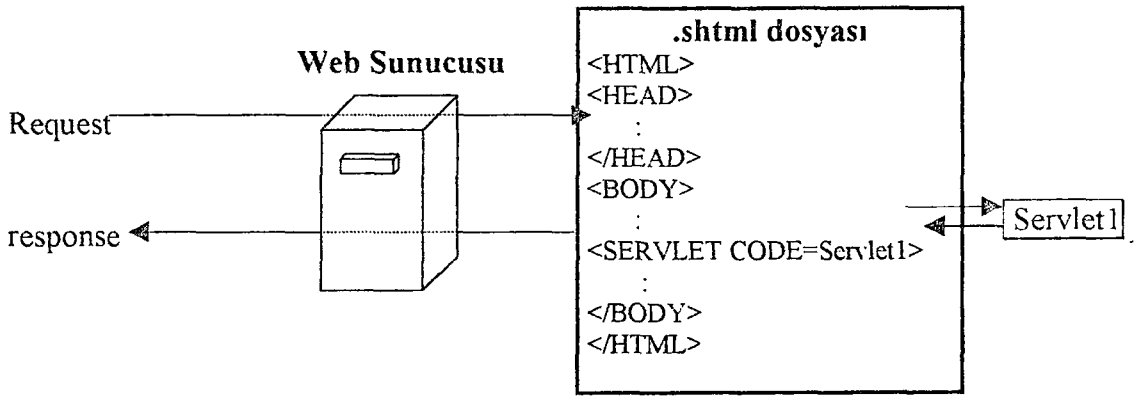
Bu servlet HttpServlet sınıfının bir uzantısıdır ve ondan kaynaklanan doGet metodunu tekrar yüklemektedir. Web sunucusu, bu servlet için her seferinde Get isteğini aldığında sunucu doGet metodunu kullanmakta ve onu HttpServletRequest nesnesi ve HttpServletResponse olarak geçirmektedir.

HttpServletRequest nesnesi, kullanıcı isteğini temsil etmektedir. Bu nesne, kullanıcı hakkındaki bilgiye servlet girişi istek parametreleri ve istekle birlikte Http başlıkları sağlamaktadır.

HttpServletResponse' u servlet' in cevabını temsil etmektedir. Servlet, bu nesneyi kullanıcıya veri göndermek amacıyla kullanmaktadır. Bu veri herhangi bir içerikte olabilmektedir. Ama çeşidi cevabın bir parçası olarak sunulmalıdır. Servlet bu nesneyi Http cevap başlıkları oluşturmak için kullanabilmektedir.

Örneği verilmiş olan servlet, önce text/html (HTML sayfaları için standart MIME içerik türü) cevabının içeriğini oluşturmak için cevap nesnesinin setContentType() metodunu kullanmaktadır. Daha sonra PrintWriter() nesnesinin elde edilmesi için ise getWriter() metodu kullanılmaktadır. PrintWriter(), Java Unicode karakterlerini yerel kodlamaya çevirmektedir. (Hunter, Crawford, 1998)

istenilen sayıda parametre servlete gönderilebilir. ServletRequest metodunun GetParameter metodu kullanılarak parametre değerleri servlete geçirilmektedir. (Hunter, Crawford, 1998)



Şekil 4-14: Sunucu tarafı içeriği işlem yapısı

ARAŞTIRMA BULGULARI

Bu tezde, Web tabanlı veri tabanlarına erişim amacıyla kullanılmakta olan iki farklı yöntem incelenmiştir. Uygulama geliştirme aracı olarak Visual Basic 5.0 ve Java programlama dilleri kullanılmıştır. Her iki programlama dili ile oluşturulan veri tabanı uygulamaları Internet ortamında denenmiş ve performansları açısından karşılaştırılmıştır.

CGI ve Servlet uygulamaları Java Web Server 1.1.6 üzerinde çalıştırılmıştır. Kullanılan sunucu bilgisayar, 33.600 bps dahili modem ve 64 Mb Ram' e sahip bir Pentium 166' dır. Performans sonuçlarının gözlemlendiği istemci bilgisayarların özellikleri ise:

1. İstemci: 56 kbps dahili modem ve 81 Mb Ram' e sahip bir Pentium MMX 220
2. İstemci: 33.600 kbps dahili modem ve 64 Mb Ram' e sahip bir Pentium III 500' dür.

Uygulamada, bir kütüphaneye ait kitapların bilgilerini tutacak, Ms Access ile oluşturulmuş olan ilişkisel bir veri tabanı kullanılmıştır. Bu veri tabanı, kitabın adı, türü, yazarı, ISBN numarası, sayfa sayısı, yayınevi ve kitaba ait açıklamayı içermektedir. Performans analizi, bu veri tabanına rasgele olarak sırasıyla, 500, 1.000, 2.500, 5.000 ve 10.000 kayıt girilmesi ve bu kayıtların her iki istemci bilgisayarda, CGI ve Servlet uygulamaları kullanılarak ayrı ayrı listelenmesi yoluyla yapılmıştır. İşlem süresi, veri tabanına erişimin başladığı ve sona erdiği anlardaki saniye cinsinden süre farkının alınmasıyla belirlenmiştir.

Çalışma sonunda elde edilen sonuçlar grafiksel olarak araştırma bulguları bölümünün sonunda görülebilir. Bununla birlikte, teorik ve deneysel çalışmaları iki ayrı başlık altında değerlendirmek de mümkündür:

A) Bir Web tabanlı veri tabanı uygulaması olarak CGI ve Servlet yöntemlerinin teknik açıdan karşılaştırılması:

1- CGI, derlenebilen ve yorumlanabilen birçok programlama dili ile yazılabilmektedir. Bunun yanında Servlet API yalnızca Java programlama dili için geliştirilmiştir.

2- Servlet kullanımını destekleyen Web sunucuların sayısı CGI kullanımını destekleyenlere göre daha azdır.

3- Bir Servlet' in çalıştırılmadan önce Java Web Sunucusu üzerinde konfigüre edilmesi gerekmektedir. CGI ise herhangi özel bir konfigürasyona ihtiyaç duymadan direkt olarak çalıştırılabilmektedir.

4- Bir Servlet, istemcinin Web tarayıcısından, GET ya da POST metotları ile gelen parametreleri getParameter() ifadesi yardımı ile ekstra bir işleme ihtiyaç duymadan alabilmektedir. CGI' da ise bu işlemi gerçekleştirmek için uzun kod satırlarının yazılmasına ihtiyaç duyulmaktadır. Bu da CGI programlarının Java Servlet' lerine göre daha fazla kod satırı yazmayı gerektirmektedir.

B) CGI ve Servlet yöntemlerinin performans açısından karşılaştırılması:

1- CGI ve Servlet uygulamalarının, veri tabanındaki kayıtları listelemesinde 500, 1.000 ve 2.500 kayıt için her iki istemcide de birbirine yakın sürelerde işlem yaptığı görülmüştür.

2- Kayıt sayısı 5000 ve sonrasında 10000 yapılarak gerçekleştirilen denemelerde, istemci 1 için CGI uygulamasının Servlet' e göre çok daha fazla süre harcayarak işlem yaptığı gözlemlenmiştir. Aynı miktarlarda kayıt sayısı için İstemci 2 ile

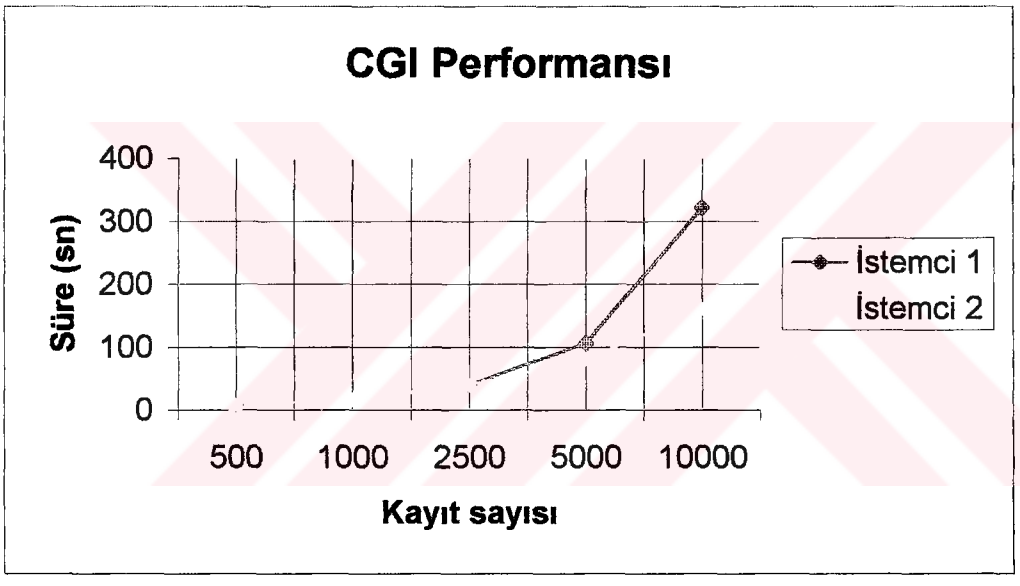
yapılan denemelerde ise CGI ve Servlet uygulamalarının listeleme işlemini birbirine çok yakın sürelerde gerçekleştirdiği tespit edilmiştir.

3- Kayıt sayısı 10.000 yapılarak İstemci 1 üzerinde gerçekleştirilen denemelerin bir çoğunda CGI' ın hafiza kullanımı konusunda problemler yaşadığı, cevap süresini yavaşlattığı ve hatta sistemi çökerttiği gözlemlenmiştir. Daha gelişmiş bir mikroişlemciye sahip olan İstemci 2 ile yapılan denemelerde ise bu tür problemlerle karşılaşılmamıştır.

Bu tez çalışmasında, 5000 ve bunun üzeri sayıda kullanıcıya sahip web siteleri söz konusu olduğunda Servlet' lerin daha güvenli bir çözüm olduğu görülmüştür. Bir CGI uygulamasının meydana getirilmesinin ve kullanımının Servlet' lere göre daha basit olduğu; bununla birlikte. ancak güçlü bir donanıma sahip web sunucuları üzerinde çalıştığı taktirde güvenli ve uygun bir çözüm olabileceği sonucuna varılmıştır.

Performans Analizinin Grafiksel Gösterimi 1: CGI Uygulaması

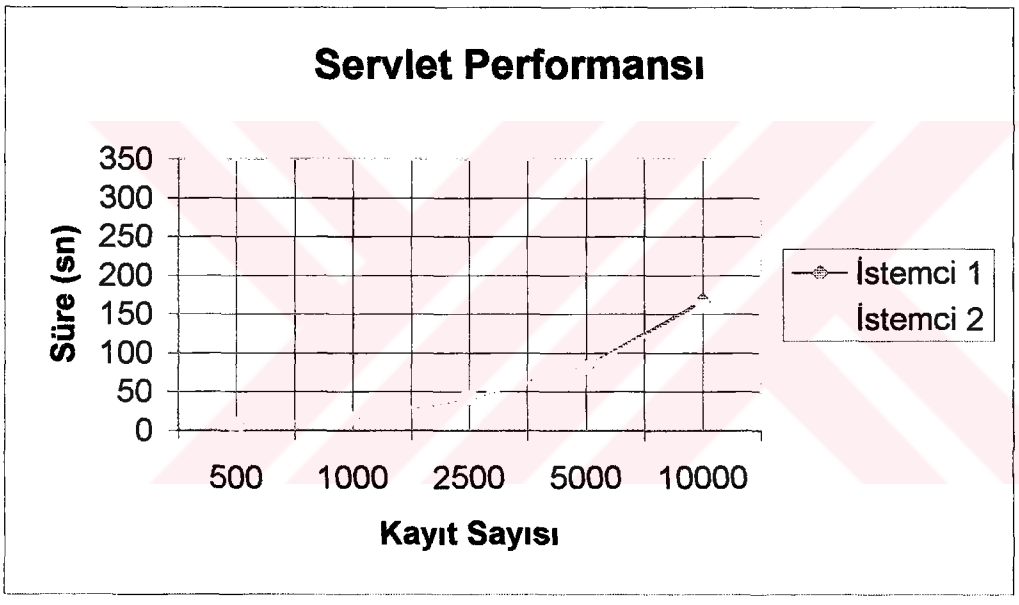
		Kayıt Sayısı				
		500	1000	2500	5000	10000
Süre (sn)	İstemci 1	7	16	40	106	321
	İstemci 2	7	16	40	80	159



İstemci 1: Pentium 220 MMX, 81 Mb Ram, 56 kpbs dahili modem
İstemci 2: Pentium III 500 MMX, 64 Mb Ram, 33600 kpbs dahili modem

Performans Analizinin Grafiksel Gösterimi 2:Servlet Uygulaması

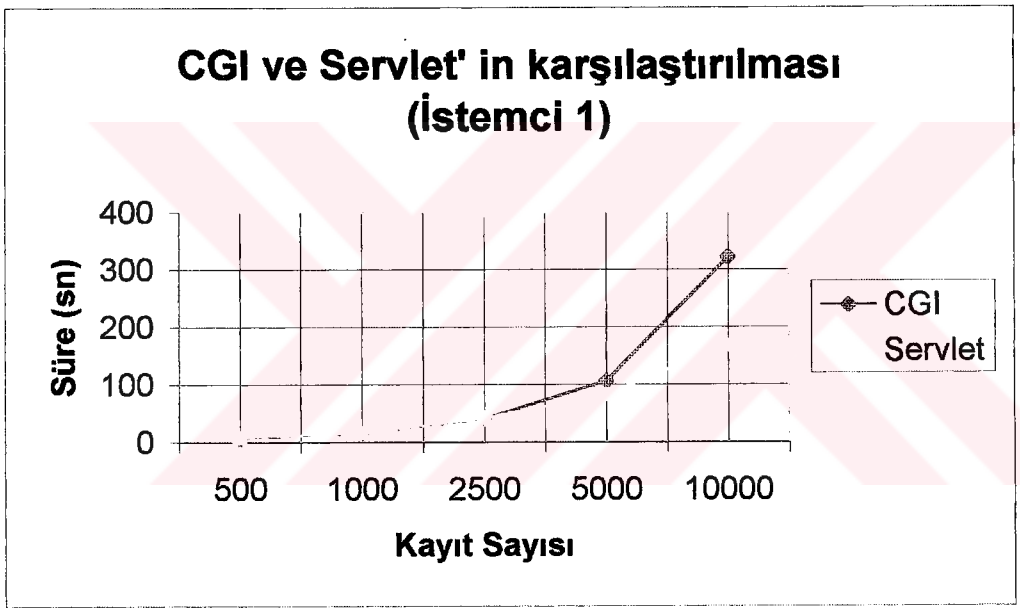
		Kayıt Sayısı				
		500	**1000**	**2500**	**5000**	**10000**
Süre (sn)	**İstemci 1**	8	16	41	82	167
	İstemci 2	8	16	42	81	160



İstemci 1: Pentium 220 MMX, 81 Mb Ram, 56 kpbs dahili modem
İstemci 2: Pentium III 500 MMX, 64 Mb Ram, 33600 kpbs dahili modem

Performans Analizinin Grafiksel Gösterimi 3: İstemci 1 İçin CGI ve Servlet Performanslarının Karşılaştırılması

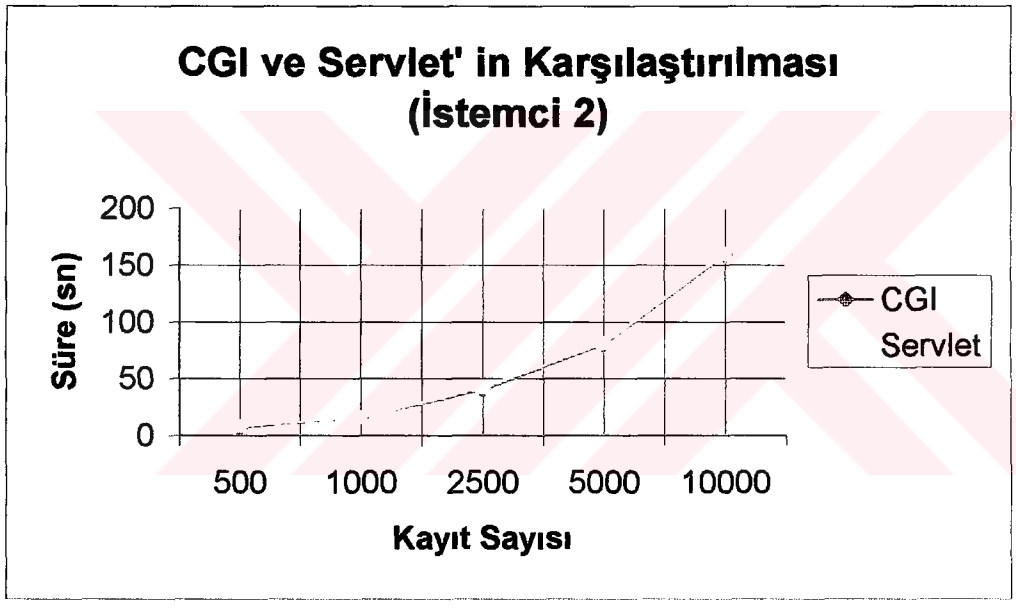
		Kayıt Sayısı				
		500	**1000**	**2500**	**5000**	**10000**
Süre (sn)	**CGI**	7	16	40	106	321
	Servlet	8	16	41	82	167



İstemci 1: Pentium 220 MMX, 81 Mb Ram, 56 kpbs dahili modem
İstemci 2: Pentium III 500 MMX, 64 Mb Ram, 33600 kpbs dahili modem

Performans Analizinin Grafiksel Gösterimi 4: İstemci 2 İçin CGI ve Servlet Performanslarının Karşılaştırılması

		Kayıt Sayısı				
		500	**1000**	**2500**	**5000**	**10000**
Süre (sn)	**CGI**	7	16	40	80	159
	Servlet	8	16	42	81	160



İstemci 1: Pentium 220 MMX, 81 Mb Ram, 56 kpbs dahili modem
İstemci 2: Pentium III 500 MMX, 64 Mb Ram, 33600 kpbs dahili modem

KAYNAKLAR

Akın, C., 1996, "Her Yönüyle Internet", Alfa Y., 975-8052-30-6,729

Colburn, R., 1998, "Teach Yourself CGI Programming in a Week", Sams net, 0-57521-381-8, 387

Fournier, R., 1999, "A Methodology for Client/Server and Web Application Development", Yourdon Press Computing Series, 0-13-598426-2, 648

Grup Java, 1997, "Java", Beta Y., 975-486-581-7,749

Halvorson, M., 1999,"Microsoft Visual Basic 5", Microsoft Press, 957-509-161-0, 378

Heller, P., Seymour, P., vd., 1998, "Java 1.1 Uygulama Geliştirme Kılavuzu", Alfa Y., 975-316-046-1

Hoobs, A., 1997, "Teach Yourself Database Programming with JDBC", Sams net, 1-57521-123-8, 524

Hunter, J., Crawford, W., 1998, "Java Servlet Programming", O' Reilly, 1-56592-391-X, 510

Joch, A., 1998, JDBC's Growing Pains", http://www.byte.com/art/9805/sec20/art2.htm

Karagülle, İ., Pala, Z., 1997, "Microsoft Visual Basic Pro 5.0", Türkmen Kitapevi, 975-7337-65-X,883

LaOr, O., 1998, "CGI Programming With Visual Basic 5", McGrawHill, 0-07-913688-05, 578

Lubling, O., Malave, L., 1998, "Developing Scalable, Reliable, Business Applications with Servlets",
http://developer.java.sun.com/developer/technicalArticles/Servlets/Razor/index.html

MageLang Institute, Fundamentals of JavaTM Servlets
http://developer.java.sun.com/developer/onlineTraining/Servlets/Fundamentals/index.html

MageLang Institute, JDBC Short Course,
http://developer.java.sun.com/developer/onlineTraining/Database/JDBCShortCourse/index.html

Oktay, D., 1997, "Kim Korkar Bilgisayardan? ACCESS", Pusula, 975-7092-14-2,427

Özel, G., 1993, "Basic Uygulamaları", Beta Y., 975-486-268-0

Stanek, W., R., 1997, "HTML JAVA VRML SGML – UNLEASHED", Sistem Y., 975-322-011-1, 899

Swank, M., Kittel, D., 1996, "World Wide Web Database Developer' s Guide", Sams net, 1-57521-048-7, 782

Sun MS, 2000, "The Java Tutorial",
http://java.sun.com/docs/books/tutorial/getStarted/index.html

Uysal, M., 1998, "Visual Basic 5.0 ile İleri Uygulamalar", Beta Y., 975-486-696-8, 424

http://www.byte.com

http://java.sun.com

http://www.javaworld.com

TEŞEKKÜR

Öncelikle, bu çalışmanın başlangıcından bitiş aşamasına kadar bana bilgi ve tecrübeleriyle yol gösteren sayın hocam Yrd. Doç. Dr. Cavit TEZCAN' a;

Konu seçimindeki ve çalışmamın ilerleyen aşamalarındaki değerli ve samimi yardımlarından dolayı Rami ŞIK' a;

Yabancı kaynakların taranması konusundaki desteklerinden ötürü Erdem ÖNGÜN' e;

Tez çalışmamın her aşamasında ilgi ve desteklerini esirgemeyen ve bana moral kaynağı olan annem Songül ÇUHADAR, babam İsmet ÇUHADAR, kardeşim Serap ÇUHADAR ve nişanlım Selmin DEVREN' e;

Sonsuz teşekkürlerimi bir borç bilirim.

ÖZGEÇMİŞ

1974 yılında Üsküdar/İstanbul' da doğdum. İlk, orta ve lise eğitimimi daha sonra ailemin görevi sebebiyle yerleştiğimiz Çatalca' da tamamladım.

1991 yılında Çatalca Lisesi' nden mezun olduktan sonra, Fırat Üniversitesi, Teknik Eğitim Fakültesi, Elektronik-Bilgisayar Eğitimi Bölümü' nde lisans eğitimime başladım. Üniversite eğitimim sırasında "ESTA ile Bilgisayar Destekli Eğitim" (Prof. Dr. Asaf Varol, 1996) isimli kitabın hazırlık ve yazım aşamalarına katıldım. Fırat Üniversitesi' nin yayınlanmış olan ilk Web sayfalarını bitirme ödevi olarak hazırladım. 1997 yılında bu bölümden mezun oldum.

Yine 1997 yılı içerisinde Trakya Üniversitesi, Eğitim Fakültesi, Sınıf öğretmenliği Bölümü' nde Araştırma Görevlisi olarak göreve başladım. Aynı yıl Trakya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bölümü' nde Yüksek Lisans programına girmeye hak kazandım.

Halen Trakya Üniversitesi, Eğitim Fakültesi, Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Bölümün' nde Araştırma Görevlisi olarak çalışmaktayım.

EKLER

Ek A: Kitap bilgileri veri tabanına ait Login.java uygulaması

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Login extends HttpServlet
{

public void service(HttpServletRequest req, HttpServletResponse res)
                          throws ServletException, IOException
{

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

out.println("<HTML><HEAD>");
out.println("<META HTTP-EQUIV='Content-Type' CONTENT='text/html;
charset=iso-8859-9'>");
out.println("<META NAME='Author' CONTENT='Cem Cuhadar'>");
out.println("<META NAME='GENERATOR' CONTENT='Mozilla/4.04 [en]
(Win95; I) [Netscape]'>");
out.println("<TITLE>Kitap Bilgileri Veri
tabani</TITLE></HEAD><BODY>");
out.println("<CENTER><TABLE BORDER=3 COLS=1 WIDTH='56%'
BGCOLOR='#000066' ><TR><TD>");
out.println("<CENTER><B><FONT SIZE=+2><FONT
COLOR='#FF0000'> </FONT><FONT COLOR='#FFCC00'>KITAP");
out.println("BILGILERI VERI
TABANI</FONT></FONT></B></CENTER></TD></TR></TABLE></CENTER>");
out.println("<CENTER><br><br></CENTER>");
out.println("<CENTER><FONT FACE='Comic Sans MS'><FONT
COLOR='#000000'><FONT SIZE=+1>Lutfen");
out.println("asagidaki menuden yapmak istediginiz islemi
seciniz.</FONT></FONT></FONT></CENTER>");
out.println("<CENTER><br></CENTER><CENTER><HR NOSHADE
WIDTH='37%'></CENTER>");
out.println("<CENTER><form method='GET'
action='../servlet/Kitap'></CENTER>");
out.println("<CENTER><SELECT name='secenek'><OPTION>Kitap
Girisi<OPTION>Arama<OPTION>Silme<OPTION>Guncelleme<OPTION>Listeleme
</SELECT><INPUT type='submit' value='Tamam'></CENTER>");
out.println("<CENTER></form></CENTER><CENTER></CENTER><CENTER>");
out.println("<HR NOSHADE WIDTH='37%'></CENTER></BODY></HTML>");
}

else{
out.println("<html><head><title>Hatali Giris!!!</title></head>");
out.println("<body>");
```

```
out.println("<br><br><br><br><br><br>");
out.println("<center><h2>Kullanici adiniz ve/veya sifreniz
yanlis!</h2></center>");
out.println("<center>Lutfen yeniden denemek icin");
out.println("<a href ='../index.html'> <h3>-Giris-</h3> </a> sayfasina
donun.</center>");

out.println("</body>");
out.println("</html>");
}
}
}
```

Ek B: Kitap bilgileri veri tabanına ait Kitap.java uygulaması

```
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;


public class Kitap extends HttpServlet
{

public void doGet(HttpServletRequest req, HttpServletResponse res)
                          throws ServletException, IOException
{

res.setContentType("text/html");
ServletOutputStream out = res.getOutputStream();


xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

out.println("<HTML><HEAD>");
out.println("<META HTTP-EQUIV='Content-Type' CONTENT='text/html;
charset=iso-8859-9'>");
out.println("<META NAME='Author' CONTENT='Cem Cuhadar'>");
out.println("<META NAME='GENERATOR' CONTENT='Mozilla/4.04 [en] (Win95;
I) [Netscape]'>");
out.println("<SCRIPT LANGUAGE='JavaScript'>var zaman = new Date();var
saniye = zaman.getSeconds();");
out.println("var saniye = zaman.getSeconds(); var dakika =
zaman.getMinutes();var saat = zaman.getHours();");
out.println("document.write('BaŸlang□‡ s□resi:' + saat + ':' + dakika
+ ':' -saniye);</SCRIPT>");
out.println("<TITLE>Kitap Bilgileri Veri
Tabani</TITLE></HEAD><BODY>");


/* ****************** KiTAP GIRI□I*************** */

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

out.println("<b><center><h2>" + secenek + "</h2></center></b><br>");
out.println("<br><center><b><h3>Lutfen asag□daki alanlara gerekli
bilgileri giriniz.</h3><pre>");
out.println("<form method='GET' action='../servlet/Kitap'><CENTER><HR
NOSHADE WIDTH='57%'></CENTER><input type='hidden' name='secenek'
value='kaydet'>");
out.println("      <b>Kitabin:</b>");
out.println("         Adi: <input type='text' name='ad'
maxlength='30'>");
```

```
out.println("        Turu: <input type='text' name='tur'
maxlength='10'>");
out.println("      Yazari: <input type='text' name='yazar'
maxlength='40'>");
out.println("        Isbn: <input type='text' name='isbn'
maxlength='5'>");
out.println("Sayfa sayisi: <input type='text' name='sayfa'
maxlength='5'>");
out.println("    Yayinevi: <input type='text' name='yayinevi'
maxlength='20'>");
out.println("  A‡iklamasi: <input type='text' name='aciklama'
maxlength='100'><br>");
out.println("           <input type='submit' value='Tamam'> <input
type='reset' value='Formu temizle'></pre>");
out.println("<br><CENTER><HR NOSHADE WIDTH='57%'></CENTER>");
}


else if(secenek.equals("kaydet")){

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

/*if (isbn!=null){*/
try
{

String driverName="sun.jdbc.odbc.JdbcOdbcDriver";
Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

String url="jdbc:odbc:Kitap.mdb";
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Statement stmt=con.createStatement();

String sql="INSERT INTO Tablo VALUES ('" + ad + "','" + tur + "','" +
yazar + "','" + isbn + "','" + sayfa + "','" + yayinevi + "','" +
aciklama + "')";
stmt.executeUpdate(sql);

out.println("<br><br><p><center><h2>Kayit kitap bilgileri veri
tabanina eklenmistir.</h2></center>");

stmt.close();
con.close();

}
catch(java.lang.Exception ex)
{
ex.printStackTrace();
```

```
}

/*
else{
out.println("<br><br><p><center><h2>Kayit islemi sirasinda kitaba ait
ad, tur, yazar, isbn ve yayinevi bilgilerini girmeniz gerekmektedir.
Lutfen bir onceki sayfaya donerek gerekli alanlari kontrol
ediniz.</h2></center>");
}
*/
}


/* ****************** ARAMA *************** */

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

out.println("<b><center><h2>" + secenek + "</h2></center></b><br>");
out.println("<br><br><CENTER><FONT FACE='Comic Sans MS'><FONT
COLOR='#000000'><FONT SIZE=+1>Lütfen");
out.println("aramak istediginiz kitabin <FONT
COLOR='#FF0000'>ISBN</FONT> numarasini asagidaki kutuya
giriniz.</FONT></FONT></FONT></CENTER>");
out.println("<CENTER><br></CENTER><CENTER><HR NOSHADE
WIDTH='37%'></CENTER>");
out.println("<CENTER><form method='GET' action='../servlet/Kitap'
></CENTER>");
out.println("<center><INPUT TYPE='HIDDEN' NAME='secenek'
VALUE='ara'><INPUT type='text' name='isbnara' size='20' maxlength='5'>
<INPUT type='submit' value='Tamam'></CENTER>");
out.println("<CENTER></form></CENTER><CENTER></CENTER><CENTER><HR
NOSHADE WIDTH='37%'></CENTER>");
}


xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
{
String driverName="sun.jdbc.odbc.JdbcOdbcDriver";
Driver driver=(Driver)Class.forName(driverName).newInstance();
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Connection con=DriverManager.getConnection(url,"cuhadar","cuhadar");
Statement stmt=con.createStatement();
String isbnara=req.getParameter("isbnara");
String sql="SELECT Ad,tur,yazar,isbn,sayfa,yayinevi,aciklama FROM
Tablo WHERE isbn='" + isbnara + "'";
ResultSet rs=stmt.executeQuery(sql);
out.println("<br><h2><center>-Arama Islemi Sonuclari-</center></h2>");

while(rs.next()){

String ad=rs.getString("ad");
String tur=rs.getString("tur");
```

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

out.println("<br><br><h2>Bulunan kayda ait bilgiler
sunlardir:</h2><pre>");
out.println("<b>Ad:</b> " + ad);
out.println("<b>Tur:</b> " + tur);
out.println("<b>Yazar:</b> " + yazar);
out.println("<b>ISBN:</b> " + isbn);
out.println("<b>Sayfa sayisi:</b> " + sayfa);
out.println("<b>Yayinevi:</b> " + yayinevi);
out.println("<b>Aciklama:</b> " + aciklama);
out.println("</pre>");
}

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

}
catch(java.lang.Exception ex)
{
ex.printStackTrace();
}

}



/* ****************** SILME *************** */

else if(secenek.equals("Silme")){

out.println("<b><center><h2>" + secenek + "</h2></center></b><br>");
out.println("<br><br><CENTER><FONT FACE='Comic Sans MS'><FONT
COLOR='#000000'><FONT SIZE=+1>Lutfen");
out.println("silmek istedi$iniz kitabin <FONT
COLOR='#FF0000'>ISBN</FONT> numarasini asagidaki kutuya
giriniz.</FONT></FONT></FONT></CENTER>");
out.println("<CENTER><br></CENTER><CENTER><HR NOSHADE
WIDTH='37%'></CENTER>");
out.println("<CENTER><form method='GET'
action='../servlet/Kitap'></CENTER>");
out.println("<center><INPUT TYPE='HIDDEN' NAME='secenek'
VALUE='sil'><INPUT type='text' name='silisbn' size='20' maxlength='5'>
<INPUT type='submit' value='Tamam'></CENTER>");
out.println("<CENTER></form></CENTER><CENTER></CENTER><CENTER><HR
NOSHADE WIDTH='37%'></CENTER>");
}


xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
try
{
String driverName="sun.jdbc.odbc.JdbcOdbcDriver";
Driver driver=(Driver)Class.forName(driverName).newInstance();
String url="jdbc:odbc:Kitap.mdb";
Connection con=DriverManager.getConnection(url,"cuhadar","cuhadar");
Statement stmt=con.createStatement();
String silisbn=req.getParameter("silisbn");
String sql="SELECT Ad,tur,yazar,isbn,sayfa,yayinevi,aciklama FROM
Tablo WHERE isbn='" + silisbn + "'";
ResultSet rs=stmt.executeQuery(sql);

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

String ad=rs.getString("ad");
String tur=rs.getString("tur");
String yazar=rs.getString("yazar");
String isbn=rs.getString("isbn");
String sayfa=rs.getString("sayfa");
String yayinevi=rs.getString("yayinevi");
String aciklama=rs.getString("aciklama");

out.println("<br><br><h2>Silmek istediginiz kayda ait bilgiler
sunlardir:</h2><pre>");
out.println("<b>Ad:</b> " + ad);
out.println("<b>Tur:</b> " + tur);
out.println("<b>Yazar:</b> " + yazar);
out.println("<b>ISBN:</b> " + isbn);
out.println("<b>Sayfa sayisi:</b> " + sayfa);
out.println("<b>Yayinevi:</b> " + yayinevi);
out.println("<b>Aciklama:</b> " + aciklama);
out.println("</pre>");

out.println("<br><center><h2>Bu kayd veri tabanindan silmek
istediginize emin misiniz?");
out.println("<form method='GET' action='../servlet/Kitap'>");
out.println("<input type='hidden' name='secenek' value='evet'>");
out.println("<input type='hidden' name='no' value='" + isbn + "'>");
out.println("<center><input type='submit' value='  Evet
'></center></form>");
out.println("<form method='GET' action='../servlet/Kitap'>");
out.println("<input type='hidden' name='secenek' value='hayir'>");
out.println("<input type='hidden' name='no' value='" + isbn + "'>");
out.println("<center><input type='submit' value=' Hayir
'></center></form>");


}

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

}
```

```
catch(java.lang.Exception ex)
{
ex.printStackTrace();
}
}


else if(secenek.equals("evet")){

try
{

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Connection con=DriverManager.getConnection(url,"cuhadar","cuhadar");
Statement stmt=con.createStatement();

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
out.println("<br><br><p><center><h2>Istenen kayit veri tabanindan
silinmistir!</h2></center>");

stmt.close();
con.close();

}
catch(java.lang.Exception ex)
{
ex.printStackTrace();
}
}



/* ****************** GUNCELLEME *************** */

else if(secenek.equals("Guncelleme")){

out.println("<b><center><h2>" + secenek + "</h2></center></b><br>");
out.println("<br><br><CENTER><FONT FACE='Comic Sans MS'><FONT
COLOR='#000000'><FONT SIZE=+1>Lutfen");
out.println("bilgilerinde guncelleme yapmak istediginiz kitabin <FONT
COLOR='#FF0000'>ISBN</FONT> numarasini asagidaki kutuya
giriniz.</FONT></FONT></FONT></CENTER>");
out.println("<CENTER><br></CENTER><CENTER><HR NOSHADE
WIDTH='37%'></CENTER>");
out.println("<CENTER><form method='GET' action='../servlet/Kitap'
></CENTER>");
out.println("<center><INPUT TYPE='HIDDEN' NAME='secenek'
VALUE='guncelleisbn'><INPUT type='text' name='guncelisbn' size='20'
maxlength='5'> <INPUT type='submit' value='Tamam'></CENTER>");
```

```
out.println("<CENTER></form></CENTER><CENTER></CENTER><CENTER><HR
NOSHADE WIDTH='37%'></CENTER>");
}

else if(secenek.equals("guncelleisbn")){

try
{
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
String sql="SELECT Ad,tur,yazar,isbn,sayfa,yayinevi,aciklama FROM
Tablo WHERE isbn='" + guncelisbn + "'";
ResultSet rs=stmt.executeQuery(sql);

while(rs.next()){

String ad=rs.getString("ad");
String tur=rs.getString("tur");
String yazar=rs.getString("yazar");
String isbn=rs.getString("isbn");
String sayfa=rs.getString("sayfa");
String yayinevi=rs.getString("yayinevi");
String aciklama=rs.getString("aciklama");

out.println("<br><center><b><h3>Lutfen guncelleme islemi icin
asagidaki alanlar uzerinde gerekli degisiklikleri
yapiniz.</h3></center><pre>");
out.println("<form method='GET' action='../servlet/Kitap' ><CENTER><HR
NOSHADE WIDTH='57%'></CENTER><input type='hidden' name='secenek'
value='guncelleson'><INPUT TYPE='HIDDEN' NAME='eskiisbn' VALUE='" +
isbn + "'>");
out.println("          Ad: <input type='text' name='ad' maxlength='30'
value='" + ad + "'>");
out.println("         Tur: <input type='text' name='tur'
maxlength='10' value='" + tur + "'>");
out.println("       Yazar: <input type='text' name='yazar'
maxlength='40' value='" + yazar + "'>");
out.println("        Isbn: <input type='text' name='isbn'
maxlength='5' value='" + isbn + "'>");
out.println("Sayfa sayisi: <input type='text' name='sayfa'
maxlength='5'  value='" + sayfa + "'>");
out.println("    Yayinevi: <input type='text' name='yayinevi'
maxlength='5' value='" + yayinevi + "'>");
out.println("    Aciklama: <input type='text' name='aciklama'
maxlength='20' value='" + aciklama + "'><br>");
out.println("           <input type='submit' value='Kaydi bu sekilde
guncelle'></pre>");
out.println("<br><CENTER><HR NOSHADE WIDTH='57%'></CENTER>");
}
```

```
stmt.close();
con.close();

}
catch(java.lang.Exception ex)
{
ex.printStackTrace();
}
}


xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

try
{

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

String url="jdbc:odbc:Kitap.mdb";
Connection con=DriverManager.getConnection(url,"cuhadar","cuhadar");
Statement stmt=con.createStatement();

String eskiisbn=req.getParameter("eskiisbn");
String sql1="DELETE FROM Tablo WHERE isbn='"+ eskiisbn +"'";
stmt.executeUpdate(sql1);


String ad=req.getParameter("ad");
String tur=req.getParameter("tur");
String yazar=req.getParameter("yazar");
String isbn=req.getParameter("isbn");
String sayfa=req.getParameter("sayfa");
String yayinevi=req.getParameter("yayinevi");
String aciklama=req.getParameter("aciklama");


xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
yazar + "','" + isbn + "','" + sayfa + "','" + yayinevi + "','" +
aciklama + "')";
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

out.println("<br><br><center><h2>istediginiz kayda ait guncelleme
islemi yapilmistir.</h2></center>");

stmt.close();
con.close();

}
catch(java.lang.Exception ex)
{
ex.printStackTrace();
}
}
```

```
/* ****************** LISTELEME *************** */

else if(secenek.equals("Listeleme")){

out.println("<b><center><h2>" + secenek + "</h2></center></b><br>");
out.println("</B><br></CENTER><TABLE BORDER COLS=7><TR><TD
nowrap><B><U>Kitabin");
out.println("Adi</U></B></TD><TD nowrap><B><U>Turu</U></B></TD><TD
nowrap>");
out.println("<B><U>Yazari</U></B></TD><TD
nowrap><B><U>ISBN</U></B></TD><TD nowrap>");
out.println("<B><U>Sayfa Sayisi</U></B></TD><TD
nowrap><B><U>Yayinevi</U></B>");
out.println("</TD><TD nowrap><B><U>Aciklama</U></B></TD></TR>");

try
{
String driverName="sun.jdbc.odbc.JdbcOdbcDriver";
Driver driver=(Driver)Class.forName(driverName).newInstance();
String url="jdbc:odbc:Kitap.mdb";
Connection con=DriverManager.getConnection(url,"cuhadar","cuhadar");
Statement stmt=con.createStatement();
String sql="SELECT Ad,tur,yazar,isbn,sayfa,yayinevi,aciklama FROM
Tablo";
ResultSet rs=stmt.executeQuery(sql);

while(rs.next()){

String ad=rs.getString("ad");
String tur=rs.getString("tur");
String yazar=rs.getString("yazar");
String isbn=rs.getString("isbn");
String sayfa=rs.getString("sayfa");
String yayinevi=rs.getString("yayinevi");
String aciklama=rs.getString("aciklama");

out.println("<TR><TD nowrap> " + ad + "</TD><TD nowrap>" + tur +
"</TD><TD nowrap>" + yazar + "</TD><TD nowrap>" + isbn + "</TD><TD
nowrap>" + sayfa + "</TD><TD nowrap>" + yayinevi + "</TD><TD nowrap>"
+ aciklama + "</TD></TR>");
}

stmt.close();
con.close();

}
catch(java.lang.Exception ex)
{
ex.printStackTrace();
}

out.println("</TABLE>");
```

```
}


else{
out.println("<CENTER><TABLE BORDER=3 COLS=1 WIDTH='56%'
BGCOLOR='#000066' ><TR><TD>");
out.println("<CENTER><B><FONT SIZE=+2><FONT
COLOR='#FF0000'> </FONT><FONT COLOR='#FFCC00'>KİTAP");
out.println("BILGILERI VERI
TABANI</FONT></FONT></B></CENTER></TD></TR></TABLE></CENTER>");
out.println("<CENTER><br><br></CENTER>");
out.println("<CENTER><FONT FACE='Comic Sans MS'><FONT
COLOR='#000000'><FONT SIZE=+1>Lutfen");
out.println("asagidaki menuden yapmak istedi§iniz islemi
seciniz.</FONT></FONT></FONT></CENTER>");
out.println("<CENTER><br></CENTER><CENTER><HR NOSHADE
WIDTH='37%'></CENTER>");
out.println("<CENTER><form method='GET'
action='../servlet/Kitap'></CENTER>");
out.println("<CENTER><SELECT name='secenek'><OPTION>Kitap
Girisi<OPTION>Arama<OPTION>Silme<OPTION>Guncelleme<OPTION>Listeleme
</SELECT><INPUT type='submit' value='Tamam'></CENTER>");
out.println("<CENTER></form></CENTER><CENTER></CENTER><CENTER>");
out.println("<HR NOSHADE WIDTH='37%'></CENTER>");
}

out.println("<CENTER><br><br><B><FONT SIZE=+1><A
HREF='../kitap.htm'>Ana Sayfa</A></FONT></B><br><br>");
out.println("<SCRIPT LANGUAGE='JavaScript'>var zaman = new Date();var
saniye = zaman.getSeconds();");
out.println("var saniye = zaman.getSeconds(); var dakika =
zaman.getMinutes();var saat = zaman.getHours();");
out.println("document.write('BitiŸ s□resi' + saat + ':' + dakika + ':'
+saniye);</SCRIPT>");
out.println("</BODY></HTML>");

}
}
```

Ek C: Kitap bilgileri veri tabanına ait Login.exe uygulaması

```
Option Explicit
Option Base 1
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  lpBuffer As Any, _
  ByVal nNumberOfBytesToRead As Long, _
  lpNumberOfBytesRead As Long, _
  lpOverlapped As Any) As Long
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  (ByVal hFile As Long, _
  ByVal lpBuffer As String, _
  ByVal nNumberOfBytesToWrite As Long, _
  lpNumberOfBytesWritten As Long, _
  lpOverlapped As Any) As Long

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Public CGI_Accept          As String
Public CGI_AuthType        As String
Public CGI_ContentLength   As String
Public CGI_ContentType     As String
Public CGI_GatewayInterface As String
Public CGI_PathInfo        As String
Public CGI_PathTranslated  As String
Public CGI_QueryString     As String
Public CGI_Referer         As String
Public CGI_RemoteAddr      As String
Public CGI_RemoteHost      As String
```

```
Public CGI_RemoteIdent      As String
Public CGI_RemoteUser       As String
Public CGI_RequestMethod    As String
Public CGI_ScriptName       As String
Public CGI_ServerSoftware   As String
Public CGI_ServerName       As String
Public CGI_ServerPort       As String
Public CGI_ServerProtocol   As String
Public CGI_UserAgent        As String

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Public esittirs As Long
Public uzunluk As Long
Public alan As String
Public say As String


Public Type dizi
   degerd As String
End Type

Public Type alan
   degera As String
End Type

Public Type son
   ads As String
   degers As String
```

```
End Type

Dim dizit() As dizi
Dim alant() As alan
Dim sont() As son
"Dim login As String
"Dim password As String

Const login = "cuhadar"
Const password = "1974"


Sub Main()
If Trim$(Command$) = "" Then
Inter_Main
Exit Sub
End If

InitCgi
CGI_Main

End Sub

Sub Inter_Main()
MsgBox "Bu bir CGI programıdır."
End Sub

Sub InitCgi()
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
Dim x, y, p, k As Long
Dim katar As String

CGI_Accept = Environ("HTTP_ACCEPT")
CGI_AuthType = Environ("AUTH_TYPE")
CGI_ContentLength = Environ("CONTENT_LENGTH")
CGI_ContentType = Environ("CONTENT_TYPE")
CGI_GatewayInterface = Environ("GATEWAY_INTERFACE")
CGI_PathInfo = Environ("PATH_INFO")
CGI_PathTranslated = Environ("PATH_TRANSLATED")
CGI_QueryString = Environ("QUERY_STRING")
CGI_Referer = Environ("HTTP_REFERER")
CGI_RemoteAddr = Environ("REMOTE_ADDR")
CGI_RemoteHost = Environ("REMOTE_HOST")
CGI_RemoteIdent = Environ("REMOTE_IDENT")
CGI_RemoteUser = Environ("REMOTE_USER")
CGI_RequestMethod = Environ("REQUEST_METHOD")
CGI_ScriptName = Environ("SCRIPT_NAME")
CGI_ServerSoftware = Environ("SERVER_SOFTWARE")
CGI_ServerName = Environ("SERVER_NAME")
CGI_ServerPort = Environ("SERVER_PORT")
CGI_ServerProtocol = Environ("SERVER_PROTOCOL")
CGI_UserAgent = Environ("HTTP_USER_AGENT")

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
For x = 1 To uzunluk
   say = Mid(katar, x, 1)
   If say = "=" Then esittirs = esittirs + 1
```

```
Next x

ReDim alant(esittirs)
ReDim sont(esittirs)
ReDim dizit(uzunluk)

For x = 1 To uzunluk
   dizit(x).degerd = Mid(katar, x, 1)
Next x

p = 0
k = 0

For y = 1 To esittirs
   For x = (1 + p + k) To uzunluk
      If dizit(x).degerd = "&" Or dizit(x).degerd = "" Then Exit For
      alant(y).degera = alant(y).degera + dizit(x).degerd
   Next x

   p = p + 1
   k = k + Len(alant(y).degera)

Next y

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

End Sub
```

```
Sub SendHeader(baslik As String)
Send "Status: 200 OK"
Send "Content-type: text/html" & vbCrLf
Send "<HTML><HEAD><TITLE>" & baslik & "</TITLE></HEAD>"
End Sub

Sub Send(s As String)
Dim lBytesWritten As Long

s = s & vbCrLf
WriteFile hStdOut, s, Len(s), lBytesWritten, ByVal 0&
End Sub



xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Dim x, y As Long

   Send "Status: 200 OK"
   Send "Content-type: text/html" & vbCrLf
   Send "<HTML><HEAD><TITLE>Kitap Bilgileri Veritabanı</TITLE></HEAD>"
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
      anasayfa
      GoTo son
   End If

   Send "<p>Giriş başarısız!!!"

son:
   Send "</BODY></HTML>"
```

```
End Sub


Sub anasayfa()
Send " <CENTER><TABLE BORDER=3 COLS=1 WIDTH='56%'
BGCOLOR='#000066' ><TR><TD>"
Send "<CENTER><B><FONT SIZE=+2><FONT
COLOR='#FF0000'> </FONT><FONT COLOR='#FFCC00'>KİTAP"
Send "BİLGİLERİ VERİTABANI</FONT></FONT></B></CENTER>"
Send "</TD></TR></TABLE></CENTER><CENTER><br><br></CENTER>"
Send "<CENTER><FONT FACE='Comic Sans MS'><FONT
COLOR='#000000'><FONT SIZE=+1>Lütfen"
Send "aşağıdaki menüden yapmak istediğiniz işlemi
seçiniz.</FONT></FONT></FONT></CENTER>"
Send "<CENTER><br></CENTER><CENTER><HR NOSHADE
WIDTH='37%'></CENTER>"
Send "<CENTER><form method='GET' action='../cgi-bin/hello.exe' ></CENTER>"
Send "<CENTER><SELECT name='secenek'><OPTION>Kitap
Girisi<OPTION>Arama"
Send "<OPTION>Silme<OPTION>Guncelleme<OPTION>Listeleme</SELECT>"
Send "<INPUT type='submit' value='Tamam'></CENTER>"
Send "<CENTER></form></CENTER><CENTER></CENTER><CENTER><HR
NOSHADE WIDTH='37%'></CENTER>"

End Sub
```

Ek D: Kitap bilgileri veri tabanına ait Kitap.exe uygulaması

```
Option Explicit

Option Base 1

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  (ByVal nStdHandle As Long) As Long
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  (ByVal hFile As Long, _
  lpBuffer As Any, _
  ByVal nNumberOfBytesToRead As Long, _
  lpNumberOfBytesRead As Long, _
  lpOverlapped As Any) As Long
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  (ByVal hFile As Long, _
  ByVal lpBuffer As String, _
  ByVal nNumberOfBytesToWrite As Long, _
  lpNumberOfBytesWritten As Long, _
  lpOverlapped As Any) As Long

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Public CGI_Accept          As String
Public CGI_AuthType        As String
Public CGI_ContentLength   As String
Public CGI_ContentType     As String
Public CGI_GatewayInterface  As String
Public CGI_PathInfo        As String
Public CGI_PathTranslated  As String
Public CGI_QueryString     As String
```

```
Public CGI_Referer          As String
Public CGI_RemoteAddr       As String
Public CGI_RemoteHost       As String
Public CGI_RemoteIdent      As String
Public CGI_RemoteUser       As String
Public CGI_RequestMethod    As String
Public CGI_ScriptName       As String
Public CGI_ServerSoftware   As String
Public CGI_ServerName       As String
Public CGI_ServerPort       As String
Public CGI_ServerProtocol   As String
Public CGI_UserAgent        As String

Public hStdIn        As Long
Public hStdOut       As Long

Public esittirs As Long
Public uzunluk As Long
Public alan As String
Public say As String


Public Type dizi
   degerd As String
End Type

Public Type alan
   degera As String
End Type
```

```
Public Type son
   ads As String
   degers As String
End Type

Dim dizit() As dizi
Dim alant() As alan
Dim sont() As son

Dim veritabani As Database
Dim kayit As Recordset

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

If Trim$(Command$) = "" Then
Inter_Main
Exit Sub
End If

InitCgi
CGI_Main

End Sub

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
MsgBox "Bu bir CGI programıdır."
End Sub

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Dim x, y, p, k As Long
Dim katar As String


CGI_Accept = Environ("HTTP_ACCEPT")
CGI_AuthType = Environ("AUTH_TYPE")
CGI_ContentLength = Environ("CONTENT_LENGTH")
CGI_ContentType = Environ("CONTENT_TYPE")
CGI_GatewayInterface = Environ("GATEWAY_INTERFACE")
CGI_PathInfo = Environ("PATH_INFO")
CGI_PathTranslated = Environ("PATH_TRANSLATED")
CGI_QueryString = Environ("QUERY_STRING")
CGI_Referer = Environ("HTTP_REFERER")
CGI_RemoteAddr = Environ("REMOTE_ADDR")
CGI_RemoteHost = Environ("REMOTE_HOST")
CGI_RemoteIdent = Environ("REMOTE_IDENT")
CGI_RemoteUser = Environ("REMOTE_USER")
CGI_RequestMethod = Environ("REQUEST_METHOD")
CGI_ScriptName = Environ("SCRIPT_NAME")
CGI_ServerSoftware = Environ("SERVER_SOFTWARE")
CGI_ServerName = Environ("SERVER_NAME")
CGI_ServerPort = Environ("SERVER_PORT")
CGI_ServerProtocol = Environ("SERVER_PROTOCOL")
CGI_UserAgent = Environ("HTTP_USER_AGENT")

uzunluk = Len(CGI_QueryString)
```

```
katar = CGI_QueryString


For x = 1 To uzunluk
   say = Mid(katar, x, 1)
   If say = "=" Then esittirs = esittirs + 1
Next x


xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

For x = 1 To uzunluk
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Next x


p = 0
k = 0


For y = 1 To esittirs
   For x = (1 + p + k) To uzunluk
      If dizit(x).degerd = "&" Or dizit(x).degerd = "" Then Exit For
      alant(y).degera = alant(y).degera + dizit(x).degerd
   Next x


   p = p + 1
   k = k + Len(alant(y).degera)

Next y


For y = 1 To esittirs
```

```
    x = InStr(1, alant(y).degera, "=", 1)
    sont(y).ads = Left(alant(y).degera, x - 1)
    sont(y).degers = Right(alant(y).degera, Len(alant(y).degera) - x)
Next y

End Sub

Sub Send(s As String)
Dim lBytesWritten As Long

s = s & vbCrLf
WriteFile hStdOut, s, Len(s), lBytesWritten, ByVal 0&
End Sub


xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Dim x As Long
Dim a, b As String
Dim saniye


xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

    Send "Status: 200 OK"
    Send "Content-type: text/html" & vbCrLf
    Send "<HTML><HEAD base='http://localhost:8080'><META HTTP-
    EQUIV='Content-Type' CONTENT='text/html; charset=iso-8859-9'>"
    Send "<META NAME='Author' CONTENT='Cem Çuhadar'><META NAME =
'GENERATOR'"
```

```
   Send "CONTENT='Mozilla/4.04 [en] (Win95; I) [Netscape]'><TITLE>Kitap Sipariş
Formu</TITLE>"
   Send "</HEAD><BODY> <CENTER><TABLE BORDER=3 COLS=1
WIDTH='56%' BGCOLOR='#000066' >"
   Send "<TR><TD><CENTER><B><FONT SIZE=+2><FONT
COLOR='#FF0000'> </FONT><FONT"
   Send "COLOR='#FFCC00'>KİTAP BİLGİLERİ
VERİTABANI</FONT></FONT></B></CENTER>"
   Send "</TD></TR></TABLE></center>"

   b = sont(1).degers

   Select Case b

      Case "Kitap+Girisi"
         kgirisi
         GoTo son
      Case "Listeleme"
         listeleme
         GoTo son
      Case "Silme"
         silme
         GoTo son
      Case "Arama"
         arama
         GoTo son
      Case "Guncelleme"
         guncellemenogir
         GoTo son
      Case "isbn"
```

```
            aramagoster
            GoTo son
        Case "silisbn"
            silisbn
            GoTo son
        Case "guncelleisbn"
            guncelleisbn
            GoTo son
        Case "kgirisi"
            kayitekle
            GoTo son
        Case "guncelleson"
            guncelleson
            GoTo son
        Case "evet"
            kayitsil
            GoTo son
        Case "hayir"
            Send "<center><h2>Silme işlemi iptal edilmiştir!</h2></center>"
            GoTo son
        Case Else
            Send "Hata!!!!"
    End Select

    kayit.Close
    veritabani.Close

son:
    Send "<CENTER><br><br><B><FONT SIZE=+1><A HREF='../kitap.htm'>Ana
Sayfa</A></FONT></B><br><br>"
```

```
    Send "</BODY></HTML>"
End Sub

Sub listeleme()

Send "</CENTER><CENTER><br><B><FONT SIZE=+1>'KİTAP
LİSTELEME'</FONT>"
Send "</B><br></CENTER><TABLE BORDER COLS=7><TR><TD
nowrap><B><U>Kitabın"
Send "adı</U></B></TD><TD nowrap><B><U>Türü</U></B></TD><TD nowrap>"
Send "<B><U>Yazarı</U></B></TD><TD
nowrap><B><U>ISBN</U></B></TD><TD nowrap>"
Send "<B><U>Sayfa Sayısı</U></B></TD><TD
nowrap><B><U>Yayınevi</U></B>"
Send "</TD><TD nowrap><B><U>Açıklama</U></B></TD></TR>"

 If kayit.EOF And kayit.BOF Then
      Send "<br><br><p><center><h2>Veritabanı içerisinde hiçbir kayıt
bulunmamaktadır...</h2></center>"
      GoTo sonliste
   Else
      Do Until kayit.EOF
         Send " <TR><TD nowrap> " & kayit!ad & "</TD><TD nowrap>" & kayit!tur
& "</TD><TD nowrap>" & kayit!yazar & "</TD><TD nowrap>" & kayit!isbn &
"</TD><TD nowrap>" & kayit!sayfa & "</TD><TD nowrap>" & kayit!yayinevi &
"</TD><TD nowrap>" & kayit!aciklama & "</TD></TR>"
         kayit.MoveNext
      Loop
   End If
```

```
sonliste:

Send "</TABLE>"
End Sub


Sub arama()
Send "<br><br><CENTER><FONT FACE='Comic Sans MS'><FONT
COLOR='#000000'><FONT SIZE=+1>Lütfen"
Send "aramak istediğiniz kitabın <FONT COLOR='#FF0000'>ISBN</FONT>
numarasını aşağıdaki kutuya giriniz.</FONT></FONT></FONT></CENTER>"
Send "<CENTER><br></CENTER><CENTER><HR NOSHADE
WIDTH='37%'></CENTER>"
Send "<CENTER><form method='GET' action='../cgi-bin/hello.exe' ></CENTER>"
Send "<center><INPUT TYPE='HIDDEN' NAME='secenek' VALUE='isbn'><INPUT
type='text' name='isbn' size='20' maxlength='5'> <INPUT type='submit'
value='Tamam'></CENTER>"
Send "<CENTER></form></CENTER><CENTER></CENTER><CENTER><HR
NOSHADE WIDTH='37%'></CENTER>"

End Sub

Sub aramagoster()
Dim SQL As String

On Error GoTo hataarama
    Do Until kayit.EOF
      If sont(2).degers = kayit!isbn Then Exit Do
      kayit.MoveNext
    Loop
```

```
If sont(2).degers <> kayit!isbn Then GoTo hataarama

If Len(sont(2).degers) < 5 Then
   Send "<br><br><h3><center>Girdiğiniz ISBN numarası
geçersizdir!</center></h3>"
   GoTo sonarama
End If

Send "<br><br><h2>Bulunan kayda ait bilgiler şunlardır:</h2><pre>"
Send "<b>Ad:</b> " & kayit!ad
Send "<b>Tür:</b> " & kayit!tur
Send "<b>Yazar:</b> " & kayit!yazar
Send "<b>ISBN:</b> " & kayit!isbn
Send "<b>Sayfa sayısı:</b> " & kayit!sayfa
Send "<b>Yayınevi:</b> " & kayit!yayinevi
Send "<b>Açıklama:</b> " & kayit!aciklama
Send "</pre>"

hataarama:
   Send "<br><br><h3><center>Veritabanında böyle bir kayda rastlanmamıştır!"
   Send "<p>Lütfen tarayıcınızın back (geri) butonunu kulanarak bir önceki sayfaya
dönünüp gerekli düzeltmeleri yapınız.</center></h3>"
   GoTo sonarama
   Resume Next
sonarama:
End Sub
```

```
Sub kgirisi()

Send "<br><center><b><h3>Lütfen aşağıdaki alanlara gerekli bilgileri
giriniz.</h3><pre>"
Send "<form method='GET' action='../cgi-bin/hello.exe' ><CENTER><HR NOSHADE
WIDTH='57%'></CENTER><input type='hidden' name='kgir' value='kgirisi'>"
Send "        Adı: <input type='text' name='ad' maxlength='30'>"
Send "       Türü: <input type='text' name='tur' maxlength='10'>"
Send "     Yazarı: <input type='text' name='yazar' maxlength='40'>"
Send "       Isbn: <input type='text' name='isbn' maxlength='5'>"
Send "Sayfa sayısı: <input type='text' name='sayfa' maxlength='5'>"
Send "   Yayınevi: <input type='text' name='yayinevi' maxlength='20'>"
Send " Açıklaması: <input type='text' name='aciklama' maxlength='100'><br>"
Send "          <input type='submit' value='Tamam'> <input type='reset' value='Formu
temizle'></pre>"
Send "<br><CENTER><HR NOSHADE WIDTH='57%'></CENTER>"

End Sub

Sub kayitekle()

Dim hata As Boolean
hata = False

On Error GoTo hataekleme

Set kayit = veritabani.OpenRecordset("SELECT * FROM tablo")
   kayit.FindFirst "isbn='" & sont(5).degers & "'"
If kayit.NoMatch = False Then
```

```
   Send "<br><br><center><h3>Veritabanı içerisinde <FONT COLOR='#FF0000'>" &
sont(5).degers & " ISBN </font>numarasına sahip başka bir kayda rastlanmıştır!"
   Send "<p>Lütfen tarayıcınızın back (geri) butonunu kulanarak bir önceki sayfaya
dönüp gerekli düzeltmeleri yapınız.</center></h3>"
   GoTo sonekleme:
End If

kayit.AddNew
kayit!ad = sont(2).degers
kayit!tur = sont(3).degers
kayit!yazar = sont(4).degers
kayit!isbn = sont(5).degers
kayit!sayfa = sont(6).degers
kayit!yayinevi = sont(7).degers
kayit!aciklama = sont(8).degers
kayit.Update
Send "<br><center><h2>Kayıt Kitap Bilgileri Veritabanına
eklenmiştir.</h2></center>"
If hata = False Then GoTo sonekleme


hataekleme:
   Send "<br><br><h3><center>Kayıt sırasında tüm alanlara giriş yapmanız
gerekmektedir."
   Send "<p>Lütfen tarayıcınızın back (geri) butonunu kulanarak bir önceki sayfaya
dönünüp gerekli düzeltmeleri yapınız.</center></h3>"
   hata = True
   If hata = True Then GoTo sonekleme
   Resume Next
sonekleme:
```

```
End Sub


Sub silme()

Send "<br><br><CENTER><FONT FACE='Comic Sans MS'><FONT
COLOR='#000000'><FONT SIZE=+1>Lütfen"
Send "silmek istediğiniz kitabın <FONT COLOR='#FF0000'>ISBN</FONT>
numarasını aşağıdaki kutuya giriniz.</FONT></FONT></FONT></CENTER>"
Send "<CENTER><br></CENTER><CENTER><HR NOSHADE
WIDTH='37%'></CENTER>"
Send "<CENTER><form method='GET' action='../cgi-bin/hello.exe' ></CENTER>"
Send "<center><INPUT TYPE='HIDDEN' NAME='secenek'
VALUE='silisbn'><INPUT type='text' name='isbn' size='20' maxlength='5'>  <INPUT
type='submit' value='Tamam'></CENTER>"
Send "<CENTER></form></CENTER><CENTER></CENTER><CENTER><HR
NOSHADE WIDTH='37%'></CENTER>"
End Sub

Sub silisbn()

    Do Until kayit.EOF
      If sont(2).degers = kayit!isbn Then Exit Do
      kayit.MoveNext
    Loop

If sont(2).degers = "" Then GoTo hatasilme

Send "<br><center><h2>Silmek istediğiniz kayıt aşağıda
gösterilmektedir.</h2></center>"
```

```
Send "</CENTER><CENTER><br><B><FONT SIZE=+1>'KİTAP
LİSTELEME'</FONT>"
Send "</B><br></CENTER><TABLE BORDER COLS=7><TR><TD
nowrap><B><U>Kitabın"
Send "adı</U></B></TD><TD nowrap><B><U>Türü</U></B></TD><TD nowrap>"
Send "<B><U>Yazarı</U></B></TD><TD
nowrap><B><U>ISBN</U></B></TD><TD nowrap>"
Send "<B><U>Sayfa Sayısı</U></B></TD><TD
nowrap><B><U>Yayınevi</U></B>"
Send "</TD><TD nowrap><B><U>Açıklama</U></B></TD></TR>"
Send " <TR><TD nowrap> " & kayit!ad & "</TD><TD nowrap>" & kayit!tur &
"</TD><TD nowrap>" & kayit!yazar & "</TD><TD nowrap>" & kayit!isbn &
"</TD><TD nowrap>" & kayit!sayfa & "</TD><TD nowrap>" & kayit!yayinevi &
"</TD><TD nowrap>" & kayit!aciklama & "</TD></TR></table>"
Send "<br><center><h2>Bu kaydı veritabanından silmek istediğinize emin misiniz?"
Send "<form method='GET' action='../cgi-bin/hello.exe' >"
Send "<input type='hidden' name='silmeonay' value='evet'>"
Send "<input type='hidden' name='kayit' value='" & sont(2).degers & "'>"
Send "<center><input type='submit' value=' Evet '></center></form>"
Send "<form method='GET' action='../cgi-bin/hello.exe' >"
Send "<input type='hidden' name='silmeonay' value='hayir'>"
Send "<input type='hidden' name='kayit' value='" & sont(2).degers & "'>"
Send "<center><input type='submit' value=' Hayır '></center></form>"

hatasilme:
   Send "<br><br><h3><center>Veritabanında böyle bir kayıt bulunamamıştır!"
   Send "<p>Lütfen tarayıcınızın back (geri) butonunu kulanarak bir önceki sayfaya
dönünüp gerekli düzeltmeleri yapınız.</center></h3>"

End Sub
```

```
Sub kayitsil()

   Set kayit = veritabani.OpenRecordset("SELECT * FROM tablo")
   kayit.FindFirst "isbn='" & sont(2).degers & "'"
   kayit.Delete
   Send "<br><br><p><center><h2>İstenen kayıt silinmiştir!</h2></center>"

End Sub



Sub guncellemenogir()

Send "<br><br><CENTER><FONT FACE='Comic Sans MS'><FONT
COLOR='#000000'><FONT SIZE=+1>Lütfen"
Send "bilgilerinde güncelleme yapmak istediğiniz kitabın <FONT
COLOR='#FF0000'>ISBN</FONT> numarasını aşağıdaki kutuya
giriniz.</FONT></FONT></FONT></CENTER>"
Send "<CENTER><br></CENTER><CENTER><HR NOSHADE
WIDTH='37%'></CENTER>"
Send "<CENTER><form method='GET' action='../cgi-bin/hello.exe' ></CENTER>"
Send "<center><INPUT TYPE='HIDDEN' NAME='secenek'
VALUE='guncelleisbn'><INPUT type='text' name='isbn' size='20' maxlength='5'>
<INPUT type='submit' value='Tamam'></CENTER>"
Send "<CENTER></form></CENTER><CENTER></CENTER><CENTER><HR
NOSHADE WIDTH='37%'></CENTER>"
```

```
End Sub


Sub guncelleisbn()

    Do Until kayit.EOF
       If sont(2).degers = kayit!isbn Then Exit Do
       kayit.MoveNext
    Loop


Send "<br><center><b><h3>Lütfen güncelleme işlemi için aşağıdaki alanlar üzerinde
gerekli değişiklikleri yapınız.</h3></center><pre>"
Send "<form method='GET' action='../cgi-bin/hello.exe' ><CENTER><HR NOSHADE
WIDTH='57%'></CENTER><input type='hidden' name='islem' value='guncelleson'>"
Send "Adı: <input type='text' name='ad' maxlength='30' value='" & kayit!ad & "'>"
Send "Türü: <input type='text' name='tur' maxlength='10' value='" & kayit!tur & "'>"
Send "Yazarı: <input type='text' name='yazar' maxlength='40' value='" & kayit!yazar &
"'>"
Send " Isbn: <input type='text' name='isbn' maxlength='5' value='" & kayit!isbn & "'>"
Send "Sayfa sayısı: <input type='text' name='sayfa' maxlength='5'  value='" &
kayit!sayfa & "'>"
Send "Yayınevi: <input type='text' name='yayinevi' maxlength='5' value='" &
kayit!yayinevi & "'>"
Send "Açıklaması: <input type='text' name='aciklama' maxlength='20' value='" &
kayit!aciklama & "'><br><input type='hidden' name='eskiisbn' value='" &
sont(2).degers & "'>"
Send "<input type='submit' value='Kaydı bu şekilde güncelle'> <input type='reset'
value='Formu temizle'></pre>"
```

```
Send "<br><CENTER><HR NOSHADE WIDTH='57%'></CENTER>"
End Sub


Sub guncelleson()
Dim onay As Integer
onay = MsgBox("Kayıdın bu hali ile güncellenmesini onaylıyor musunuz?", 4 + 32 +
256, "Güncelleme uyarısı")
If onay = 7 Then
   Send "<br><br><h2><center>Güncelleme işlemi iptal edilmiştir!</center></h2>"
   Exit Sub
End If

   Set kayit = veritabani.OpenRecordset("SELECT * FROM tablo")
   kayit.FindFirst "isbn='" & sont(9).degers & "'"
   kayit.Delete

kayit.AddNew
kayit!ad = sont(2).degers
kayit!tur = sont(3).degers
kayit!yazar = sont(4).degers
kayit!isbn = sont(5).degers
kayit!sayfa = sont(6).degers
kayit!yayinevi = sont(7).degers
kayit!aciklama = sont(8).degers
kayit.Update

Send "<br><br><center><h2>İstediğiniz kayda ait güncelleme işlemi
yapılmıştır.</h2></center>"
End Sub
```