NESNE YÖNELİMLİ PROGRAMLAMA 2(Object Oriented Programming 2/OOP)

Öğr. Gör. Celil ÖZTÜRK
Marmara Üniversitesi
Teknik Bilimler Meslek Yüksekokulu

İçerik

- Exceptions(İstisnalar)
- Lambda İfadeleri
- File Sınıfı
- JSP-Servlet

Exception Nedir?

- Exception, programın çalışma zamanı sırasında olağan dışı meydana gelen durumlardır . Programın çalışma zamanı sırasında olağan dışı bir durum gerçekleşirse compiler bir obje yaratır .
- Bu objeye exception object diyebiliriz.
- Exception object runtime sırasında oluşturulan hata hakkında bilgileri tutar.
- Exception (istisna) program çalışırken oluşan bir problemin gösterim şeklidir.
- Exception yönetimi programcıya oluşan hatalarda sonuçlar üretebilen program yazma olanağı sunar. Oluşan Exception ları yöneterek bu hatayı giderebiliriz ama bazen bu hata karşısında hiçbir şey yapamayabiliriz.

Exception (İstisna) nın Oluşabileceği Bazı Durumlar

- Dizi indisinin aşılması
- Aritmetik taşma (bir değişkenin sınırlarının aşılması)
- Sıfıra bölme
- Geçersiz metot parametreleri
- Thread in kesilmesi
- Başarısız bellek alanı ayırma işlemi
- Okuma yazma hataları

Exception’ın kullanımı

Bir metot herhangi bir problemle karşılaştığı zaman bir exception throw eder , yani oluşturur. Bu exception ı işleyecek kod olsa da olmasa da exception oluşturulur.

Exception’ın kullanımı

```
try{
        burada çeşitli kodlar yazılır. Bu kodlarda hata oluşma  olasılığı
yüksektir. Gereksiz kodları try bloğu içine almak gereksizdir ve programın
yavaşlamasını ve karmaşıklaşmasını sağlar.
}
catch(ExceptionTipi e){
}
catch(BaskabirExceptiontipi e){
}
...
finally{
        burada try bloğunda oluşan hata ne olursa olsun çalışması
        gereken kodlar yazılır.
}
```

Exception’ın kullanımı

```
try {
num1=Integer.parseInt(inputField1.getText());
num2=Integer.parseInt(inputField2.getText());
result=num1/num2;
outputField.setText(String.valueOf(result));
} catch (NumberFormatException ex) {
System.err.println(ex.getMessage());
} catch(ArithmeticException ex){
System.err.println(ex.getMessage());
}
}
}

public static void main(String[] args) {
DivideByZero app=new DivideByZero();
app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
/* (non-Javadoc)
 * @see java.awt.event.ActionListener#actionPerformed(java.awt.event.ActionEvent)
 */
public void actionPerformed(ActionEvent e) {
outputField.setText("");
try {
num1=Integer.parseInt(inputField1.getText());
num2=Integer.parseInt(inputField2.getText());
result=num1/num2;
outputField.setText(String.valueOf(result));
} catch (NumberFormatException ex) {
System.err.println(ex.getMessage());
} catch(ArithmeticException ex){
System.err.println(ex.getMessage());
}
}
}
```

Exceptions

Exceptionlar 3 tipe ayrılır .

- Error
- Checked Exception
- Runtime exception(unchecked exception)

ERROR

- Java.lang.Error sınıfı, uygulamaların yakalayıp düzeltmeye çalışmayacağı ciddi hataları ifade eder.
- Bir uygulamada hiçbir şekilde bu tipten nesnenin oluşumuna izin verilmemelidir.
- Bir metodun fırlattığı Error nesnelerini **throws** ifadesiyle belirtmesi gerekmediği gibi bu tipten nesnelerin **try-catch** ile yakalanması da gerekmez.

ERROR

Java.lang paketindeki bazı Error sınıfları şunlardır :

- NoClassDefFoundError : Yüklenecek sınıf bulunamadığında fırlatılır.
- VirtualMachineError : JVM'nin çalışması sırasında oluşan hatalı durumlar için kullanılır.

Checked Exception

- Java.lang.Exception sınıfı, yakalanması düşünülen her türlü sıra dışı durumu temsil eder.
- Uygulamada oluşan sıra dışı durumlar ya Exception sınıfının ya da alt sınıflarının nesnesi olarak fırlatılır.

Checked Exception

Exception	Description
ClassNotFoundException	Sınıf bulunamadı.
CloneNotSupportedException	Nesne kopyalama, Kopyalanabilir arayüzde uygulanamadı.
IllegalAccessException	Sınıfa erişim engellendi.
InstantiationException	Abstract bir sınıf veya arayüzde nesne oluşturma.
InterruptedException	Bir thread, diğer bir thread tarafından kesildi.
NoSuchFieldException	İstenen dosya yok.
NoSuchMethodException	İstenen metot yok.

Runtime Exception

- Program çalışırken, erişilen bir değişkenin null olması, veya bir dizinin var olamayan bir indeks'teki elemanına erişmek gibi telafisi olmayan durumları ifade eder. Bunlar için çoğu zaman bir aksiyon almanız gerekmez. Programın böyle bir durumda işine devam etmemesi gerekebilir

Runtime Exception

- Aşağıdaki dizi, 2 elemanlı bir dizi olarak deklare edilmiştir. Ardından kod, 3. elemente erişmeyi deniyor. Burada bir istisna oluşmaktadır.

```
// File Name : ExcepTest.java
import java.io.*;
public class ExcepTest{

   public static void main(String args[]){
      try{
         int a[] = new int[2];
         System.out.println("Access element three :" + a[3]);
      }catch(ArrayIndexOutOfBoundsException e){
         System.out.println("Exception thrown  :" + e);
      }

   }
}
```

Bu aşağıdaki sonucu üretecektir:

```
Exception thrown  :java.lang.ArrayIndexOutOfBoundsException: 3
```

LAMBDA İFADELERİ (Lambda Expressions)

- Lambda ifadeleri de Java 8 ile birlikte gelen en büyük yenilik olarak görülmüştür. Ana amacı fonksiyonel programlamayı kolaylaştırarak kod geliştirmeyi veya yazmayı daha sade ve basit hale getirmeyi sağlamaktır. Bir lambda ifadesi anonim bir fonksiyonun kısa bir şekilde gösterimi de diyebiliriz. Bir isme sahip değildir ancak parametre listesine, bir gövdeye ve bir dönüş tipine, ayrıca fırlatılabilecek istisnaların (exceptions) bir listesine sahiptir.

LAMBDA İFADELERİ (Lambda Expressions)

- Anonim çünkü bir metodun normalde sahip olacağı gibi belirgin bir isme sahip değildir.
- Fonksiyon, çünkü bir lambda belirli bir sınıfla bir metod gibi ilişkili değildir. Ancak bir metod gibi parametre listesi, bir gövdesi, bir dönüş tipi ve muhtemel fırlatılabilecek istisna listesine sahiptir.
- Bir lambda ifadesi bir metotta argüman olarak geçirilebilir veya bir değişkende depolanabilir.
- Yazımı kısadır, anonim sınıflar gibi basmakalıp uzun bir yazımı yoktur

LAMBDA İFADELERİ (Lambda Expressions)

!! Bu özelliklerin kullanılabilmesi için makinede JRE (Java Runtime Environment) 1.7.x değil 1.8.x sürümü kurulu olmalıdır!!

Parametre listeleri : Buradaki iki "Elma"nın parametreleri, bir Comporator karşılaştırma metodu ile karşılaştırılır.

Lambda operatörü : Parametre listesini lambda'nın gövdesinden ayırır.

Lambda gövdesi : İki "Elma"nın ağırlıkları burada karşılaştırılır. Karşılaştırma sonucu da lambda'nın dönüş değeri olarak düşünülür.

Lambda İfadesi

Bu şekildeki lambda syntax'ı Java tasarımcıları tarafından, C# ve Scala gibi benzer özellikleri taşıyan dillerde başarılı olduğu için seçilmiştir. Yukarıda da anlatıldığı üzere temel syntax;

(Parametreler) -> ifade

veya

(Parametreler) -> { ifadeler; }

Lambda Örnekleri

Kullanım Senaryosu	Lambda Örneği
Boolean ifadesi	(List liste) -> liste.isEmpty()
Nesnelerin oluşturulması	() -> new Elma(35)

Nesnenin tüketilmesi	(Elma e) -> {
	System.out.println(e.getAgirlik());
	}

Bir nesnede seçim/ayıklama	(String s) -> s.length()
İki değerli işlem	(int a, int b) -> a * b
İki nesneyi karşılaştırma	(Elma e1, Elma e2) -> e1.getAgirlik().compareTo(e2.getAgirlik())

- String listesindeki verileri yazdırma

```
String[] strArray = {"Bruce Wayne", "Jason Todd", "Dick Grayson", "Tim
Drake", "Barbara Gordon"};

List<String> karakterler = Arrays.asList(strArray);

for (String karakter : karakterler){

System.out.println(karakter);

}
```

Lambda'lar ile for döngüsü

```
karakterler.forEach((karakter)->System.out.println(karakter));
```

FILE SINIFI

Bu dosya diskteki bir dosya yada klasör hakkında bilgi almamızı sağlar. File sınıfı bir dosyayı açmaz yada dosya işlemi yapmaz sadece o dosya yada klasör hakkında ayrıntılı bilgi verir. File sınıfının bazı metotları :

Boolean canRead(): dosya okunabilirse true
Boolean canWrite():dosya yazılabilir mi ?
Boolean exists() :var mı ? yokmu?
Boolean isFile() : dosya mı ?
Boolean isDirectory() : klasör mü ?
String getPath() : dosya yada klasörün yolunu gösterir.
String getParent() : bir üst klasöre gider ve onun tam yolunu verir.
Long length(): dosyanın byte uzunluğunu verir.
String[] list() : Eğer klasörse içerisindeki tüm içeriği geriye döndürür.

JSP-Servlet

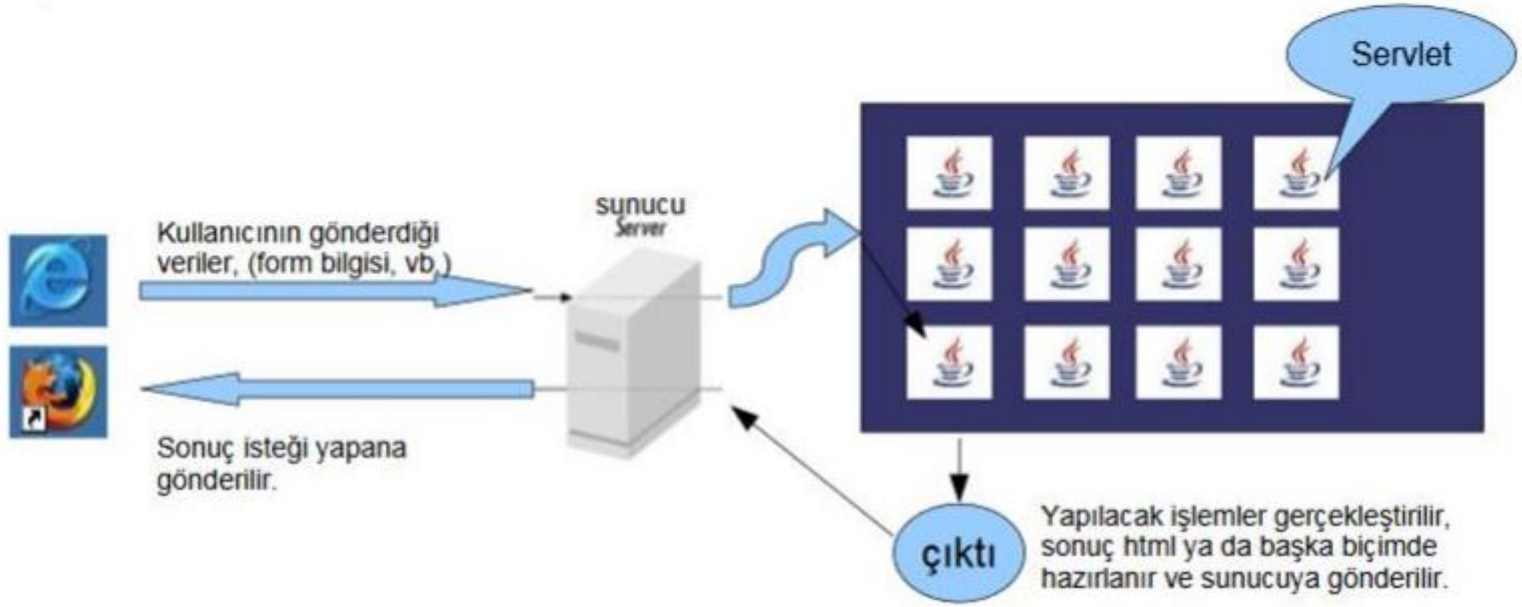
- Web sayfaları ilk başlarda durağan bir yapıya sahipti ve kullanıcıdan bilgi alarak işlemler yapmıyordu. Zamanın geçmesiyle kullanıcıya göre değişen ya da işlemler yapabilen web sayfalarının geliştirilmesi gerekli olmuştur. Web sayfalarında kullanıcılardan veri alarak bu verilerin işlenmesi ve kullanıcıya işlemlerin sonuçlarının gösterilmesi önemli olmuştur. Java ile dinamik web uygulamaları geliştirmek için farklı teknolojiler bulunmaktadır. Servlet bu teknolojilerden biridir.

Dinamik Sayfa

1. Web sayfası kullanıcının gönderdiği veriler temelinde sonuç üretir. (Örn: Google gibi arama sayfaları)
2. Web sayfasında görüntülenen veriler sürekli güncellenmektedir. (Örn: haber siteleri, hava durumu gibi)
3. Web sayfası veri tabanı bilgisi gösteriyorsa, veri tabanı değiştiğinde web sayfası da güncellenmelidir.

Servlet Nasıl Çalışır?

Servlet web sunucuları üzerinde çalışan, kullanıcıdan ya da farklı yerlerden aldığı verilere göre sonuç üreten Java sınıflarıdır. Oluşan sonuç HTML ya da başka biçimli olabilir.



Servlet Nasıl Çalışır?

1. Kullanıcı web tarayıcısından ulaşmak istediği sayfayı belirtir ve bu bilgiyi sunucuya gönderir ya da bir sayfada bulunan web form bilgilerini doldurur ve gönderme tuşuna tıklar.
2. Web sunucusu kullanıcı tarafından gelen bu http ya da başka biçimdeki isteği (request) alır. Bu isteğe uygun olan servlet'i belirler. Bu servlet'e ait bir nesne bellekte var mı diye kontrol eder. Bu kontrolde var ise istek be tüm bilgi bu servlet'e gönderilir. Eğer istenilen servlet nesnesi bellekte yok ise yeni bir tane oluşturulur, istek ve bilgi daha sonra bu servlet'e gönderilir.

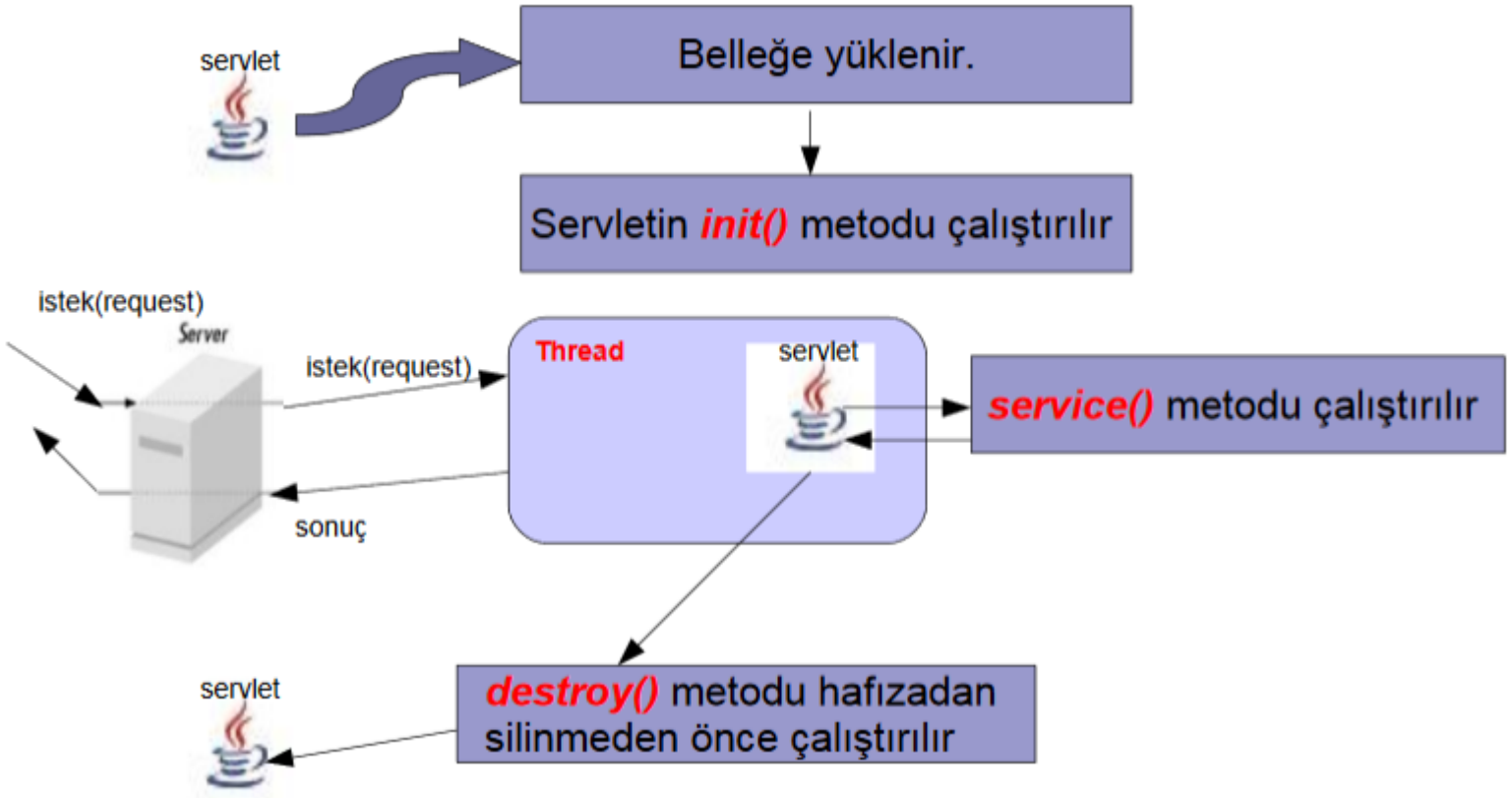
Servlet Nasıl Çalışır?

3. Servlet kendisine gelen istek ve verileri alır ve bunları kullanarak oluşturması gerekli olan sonucu oluşturur. Servlet'in oluşturacağı sonuç bilgisi farklı biçimlerde olabilir. Genel olarak oluşturulan sonuç HTML sayfası şeklindedir.
4. Servlet oluşturduğu sonucu web sunucusuna gönderir.
5. Web sunucusu servlet'ten gelen sonucu isteği yapan kullanıcıya gönderir.

Servlet Nasıl Çalışır?

Web sunucusuna belirli bir servlet için istek geldiğinde web sunucusu bu servletin bellekte olup olmamasını kontrol eder. Eğer yok ise oluşturur. Aynı servlet'e birden fazla istek aynı anda gelebilir. Sunucu bu servlet sınıfından tekbir nesne oluşturur. Bu nesneyi çok iş parçacıklı (multi threaded) şekilde tüm istekler için ortak kullanır. Yazılan servlet kodu çoklu işparçacığı (multi threaded) yapısına uygun olarak güvenli bir şekilde yazılmalıdır.

Servlet Nasıl Çalışır?



İki Servlet Geliştirilmesi

GetServlet : GET İsteklerine cevap versin

PostServlet: POST isteklerine cevap versin.

İki Servlet Geliştirilmesi

GetServlet.java

```
package ce.ustduzey.ders01;

import ...

public class GetServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {...}

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
            response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        out.println("<html>");
        out.println("<head>");
        out.println("<title>GetServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Get Servlet Örneği</h1>");
        out.println("</body>");
        out.println("</html>");

        out.close();
    }
```

İki Servlet Geliştirilmesi

PostServlet.java

```
package ce.ustduzey.ders01;

import ...

public class PostServlet extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {...}

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {...}

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
            response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        out.println("<html>");
        out.println("<head>");
        out.println("<title>PostServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Post Servlet Örneği</h1>");
        out.println("</body>");
        out.println("</html>");

        out.close();
    }
```

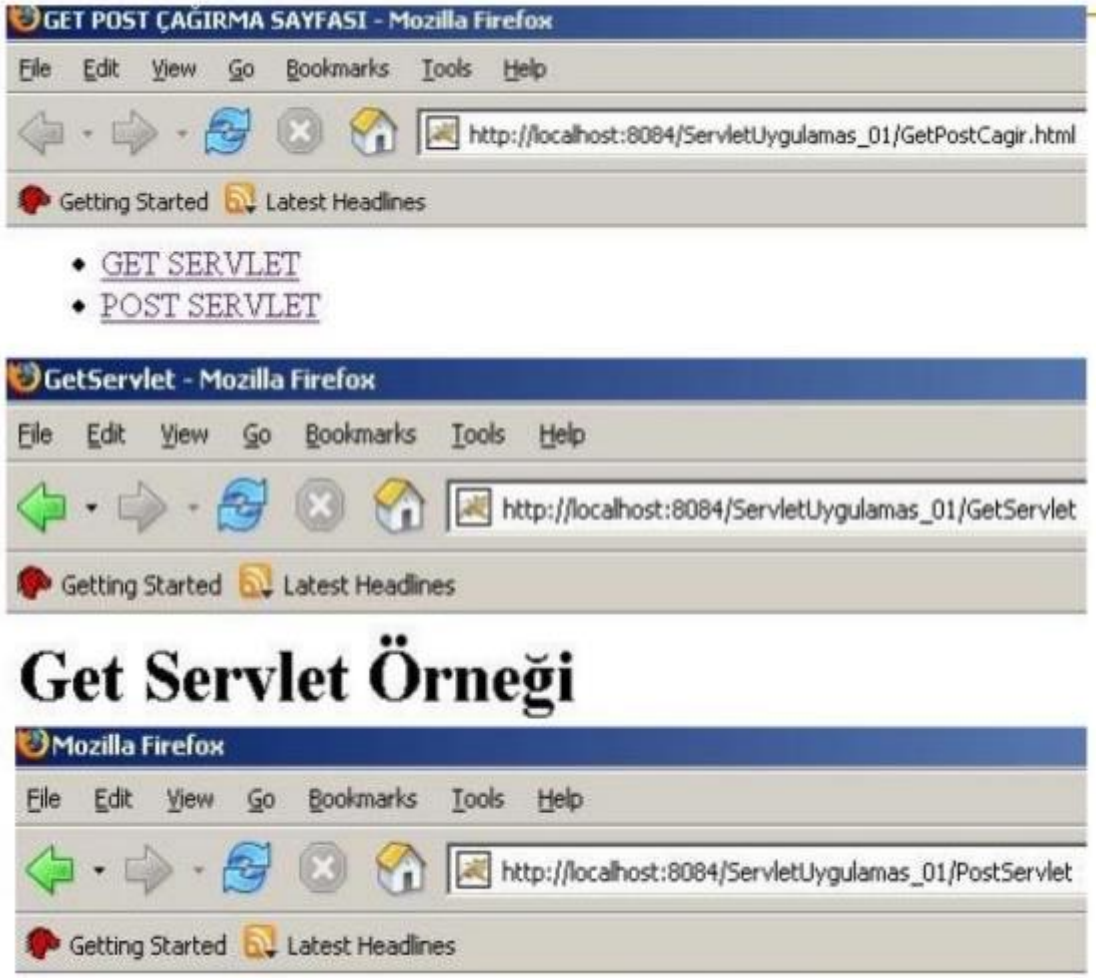
İki Servlet Geliştirilmesi

GetPostCagir.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
  <head>
      <meta http-equiv="Content-Type" content="text/html;charset=iso-8859-9">
    <title>GET POST ÇAĞIRMA SAYFASI</title>
  </head>
  <body>
      <ul>
          <li><a href="GetServlet">GET SERVLET</a></li>
          <li><a href="PostServlet">POST SERVLET</a></li>
      </ul>
  </body>
</html>
```

İki Servlet Geliştirilmesi



Kaynaklar

- Dr. Öğr. Üyesi Aysun ALTIKARDEŞ Nesne Yönelimli Programlama 2 Ders Notları
- Öğr. Gör. Mustafa ŞAHİN ders notları