# آموزش

# Microsoft Active Server Pages 3.0

مولف:

**امیر مرادآبادی**

**این مقاله در سایت www.b4c4.com ارائه شده و استفاده از آن با ذکر نام سایت مجاز می باشد.**

این پروژه قصد دارد یک راهنمای برنامه نویسی کامل، به خوبی یک کتاب مرجع برای برنامه نویسان جدی باشد.

فرض بر این است که شما هیچ دانشی از ASP ندارید و می خواهید از اول شروع کنید. فصول کتاب با توجه به پیچیدگی اسکریپت های ASP ای که می نویسید سازمان دهی شده اند. اسکریپت های ساده اول و تکنیک های لازم برای اسکریپت های پیچیده در آخر می آیند. این امکان مخصوصا برای مبتدیانی که معمولا با بحث های تکنیکی خسته کننده ی طولانی در اولین فصل کتاب های دیگر رو برو می شوند بسیار جالب است.

## مرجع VBScript

یک سند کامل همه ی توابع VBScript با اطلاعات لازم برای استفاده از آنها. مناسب برای مواقعی که شما می دانید می خواهید چه کاری انجام دهید ولی تابعی که آن کار را برای شما انجام می دهد را به یاد نمی آورید.

## نیاز به ASP

اصلا چرا خودتان را با ASP پریشان می کنید، در جایی که HTML نیاز های شما را برآورده می کند؟ اگر می خواهید اطلاعاتی را نمایش دهید، تمام کاری که لازم است انجام دهید این است که ویرایشگر متن مورد علاقه تان را باز کرده و تعدادی از تگهای HTML را در آن بنویسید و آن را با پسوند HTML ذخیره کنید.

اما صبر کنید. اگر بخواهید اطلاعاتی که تغییر می کنند را نشان دهید چه می کنید؟ اگر می خواهید صفحه ای درست کنید که به طور ثابت اطلاعات متغییری برای بازدیدکنندگان برای مثال گزارشات آب و هوا ، تغییرات سهام و ... فراهم کند، HTML دیگر شیوه ی مناسبی نیست. چیزی که نیاز دارید سیستمی است که بتواند اطلاعات پویا را نشان دهد. و ASP کاملا برای این کار مناسب است.

**ASP چیست؟**

در زبان مایکروسافت، صفحات فعال سرور ( ASP) یک محیط کاربردی باز و بدون نیاز به کامپایل به طوری که می توان در آن HTML، اسکریپت و کامپوننت های اکتیو ایکس قابل استفاده ی مجدد سرور را برای ایجاد مسائل تجاری پویا و قدرتمند تحت وب است می باشد.

ASP قابلیت Server-Side Scripting را برای IIS با پشتیبانی از VBScript و Jscript فراهم می سازد.

Active Server Pages صفحات وبی هستند که شامل Server-Side Scripts با ترکیبی از متن و تگهای HTML می باشد. Server-Side Scripts دستورات مخصوصی هستند که شما در صفحات وبتان قرار می دهید که قبل از اینکه صفحات از سرور به web-browser شخصی که در حال بازدید از سایتتان است مورد پردازش قرار می گیرد. وقتی URL ای را در i.e می نویسید یا بر روی لینکی در صفحه ای کلیک می کنید، شما از یک سرور وب در یک کامپیوتر در جایی درخواست می کنید که فایلی را به web-browser کامپیوترتان (که کلاینت نیز نامیده می شود) بفرستد. اگر آن فایل یک فایل نرمال HTML بود به همان صورتی که سرور فرستاده است دریافت می شود. بعد از دریافت فایل نشان دهنده ی صفحات وبتان محتوای آن را به صورت ترکیبی از متن، عکس و صدا نمایش می دهد.

درباره ی Active Server Pages فرایند به همین ترتیب است، فقط یک مرحله ی پردازش اضافی که درست قبل از اینکه سرور فایل را بفرستد اتفاق می افتد. قبل از اینکه سرور فایل ASP را به Browser بفرستد، تمام اسکریپت های Server-Side فایل را اجرا می کند. بعضی از این اسکریپت ها تاریخ و زمان جاری و اطلاعات دیگری را نشان می دهند. دیگری اطلاعات کاربر را که در فرم ها تایپ کرده است پردازش می کند. و شما می توانید کد خوتان را به صورت هر اطلاعات پویایی بنویسید. برای تشخیص فایل های HTML و ASP پسوند فایل های ASP را ".asp" انتخاب کرده اند.

**با ASP چه کارهایی می توانید انجام دهید؟**

- نمایش تاریخ، زمان و اطلاعات دیگر به روش های گوناگون.
- ساخت یک فرم و درخواست از کاربران برای پر کردن آن.
- فرستادن email، ذخیره ی اطلاعات در فایل و ... .
- می توانید بانک اطلاعاتی داشته باشید که کاربران از طریق وب به آن دسترسی داشته باشند. کاربران می توانند اطلاعات را نمایش، حذف و یا به روز رسانی کنند.
- می توانید قسمت های حمایت شده توسط رمز عبور داشته باشید و مطمئن باشید فقط کاربران مجاز می توانند به آنجا دسترسی داشته باشند.
- احتمالات دیگر واقعی به نظر می رسند. تمام چیز هایی که امروزه در صفحات اینترنت می بینید به راحتی با ASP قابل اجراست.

**اسکریپت های Server-Side شبیه چیست؟**

اسکریپت های Server-Side معمولا با %> شروع و با <% خاتمه می یابند. %> را تگ باز کننده و <% تگ پایان دهنده می نامند. اسکریپت های Server-Side را هر جایی از صفحه ی وبتان حتی در تگ های HTML می توانید بنویسید.

**برای اجرای ASP به چه چیز نیاز دارید؟**

از آنجایی که سرور باید پردازش های اضافی را بر روی فایل های ASP انجام دهد، باید قابلیت انجام چنین کاری را داشته باشد. سرور هایی که این امکان را به وجود می آورند Microsoft Internet Information Services و Microsoft Personal Web Server می باشند. اجازه دهید به توضیح هر کدام با ویژگی هایشان بپردازیم تا بتوانید آن یکی را که نیازهایتان را بر آورده می سازد انتخاب کنید.

**Internet Information services (IIS)**

وب سرور مایکروسافت طراحی شده برای پلت فورم ویندوز NT می باشد. IIS فقط می تواند در Windows NT 4.0 ، Windows 2000 Professional و Windows 2000 server Microsoft اجرا شود.

**Personal Web Server (PWS)**

نسخه ی جدیدی از IIS است که از بیشتر امکانات ASP پشتیبانی می کند. می تواند در تمام پلت فورم های ویندوز از جمله windows 95, 98, Me اجرا شود. معمولا برنامه نویسان ASP از PWS برای توسعه ی سایت هایشان در ماشین های خودشان استفاده می کنند و بعدا فایل هایشان را در سروری که IIS دارد بارگذاری ( upload) می کنند. اگر از ویندوز 9x یا Me استفاده می کنید تنها گزینه تان استفاده از PWS 4.0 می باشد.

**قبل از اینکه شروع کنید**

بر خلاف فایل های HTML نرمال نمی توانید بدون داشتن یک وب سرور فایل های ASP را اجرا کنید. برای تست فایل هایتان باید فایل ها را در پوشه ای که به صورت مجازی ترسیم شده است ذخیره کنید. سپس از web-browser تان برای نمایش آن استفاده کنید.

**مراحل نصب PWS**

از CD برنامه فایل SETUP.EXE را اجرا کنید. بعد از اینکه مراحل نصب به پایان رسید به آدرس زیر رفته

Start > All Programs > Microsoft PWS > Personal Web Manager.

و در برنامه زیر تب Publishing دکمه ی Start را کلیک کنید.

اکنون وب سرور تان در حال اجراست.

**مراحل نصب IIS**

از control panel گزینه ی Add or Remove Programs را اجرا کنید. گزینه ی Add/Remove Windows Components را انتخاب کنید. و در لیست باز شده گزینه ی Internet Information Services (IIS) را تیک بزنید. اکنون next را کلیک کنید. توجه داشته باشید که به CD ویندوزتان احتیاج دارید. بعد از نصب از Start menu گزینه ی Run را انتخاب کرده و inetmgr را در آن تایپ کنید. بعد از باز شدن IIS به آدرس زیر رفته

local computer/web sites/default web site

و دکمه ی play را از بالای صفحه انتخاب کنید.

اکنون وب سرور تان در حال اجراست.

**ساخت پوشه های مجازی در PWS**

بعد از اینکه سرور وبتان را نصب کردید به صورت زیر می توانید پوشه ی مجازی بسازید.

بر روی پوشه ای که می خواهید راست کلیک کنید و از منوی باز شونده Properties را انتخاب کنید. بر روی تب Web Sharing بر روی Share this folder و Add Alias کلیک کنید. اگر این گزینه ها برای شما خاموش می باشند سرور وبتان به درستی راه اندازی نشده است.

**دسترسی به صفحات**

اکنون سرورتان کاملا تنظیم و آماده برای استفاده است. اجازه دهید امتحان کنیم.

Web Browser تان را اجرا کرده و آدرس زیر را در آن تایپ کنید.

http://localhost/

اکنون باید صفحه ای را که توضیحاتی درباره ی IIS یا PWS داده است را مشاهده کنید.

**localhost چیست؟**

اجازه دهید اول ببینیم منظورمان از یک hostname چیست. وقتی از مکان دور به کامپیوتری با استفاده از URL ش دسترسی پیدا می کنید در واقع با استفاده از hostname آن کامپیوتر به آن وصل می شوید.

برای مثال:

http://www.google.com/

در واقع از شبکه می خواهید به کامپیوتری با نام www.google.com وصل شود. به این نام hostname آن کامپیوتر گفته می شود.

localhost یک hostname مخصوص است. آن همیشه به کامپیوتر خودتان اشاره می کند. پس کاری که تا به حال انجام داده اید این است که سعی کردید به صفحه ای که بر روی کامپیوترتان است دسترسی پیدا کنید. برای تست کردن صفحاتتان نیاز به localhost به عنوان یک hostname دارید. به هر حال IP مخصوصی نیز به localhost پیوند داده شده است. اگر از شبکه ی محلی در کامپیوترتان استفاده می کنید این IP همان IP کامپیوترتان در شبکه می باشد. ولی اگر شبکه ی محلی ندارید می توانید با این IP به همان صفحه ی localhost دسترسی داشته باشید.

http://127.0.0.1/

برای مثال برای دیدن فایل هایتان که در پوشه ی مجازی myscripts قرار دارند می توانید آدرس زیر را تایپ کنید.

http://localhost/myscripts/

اگر از IIS استفاده می کنید و می خواهید سایتتان را تست کنید و نمی خواهید پوشه ی مجازی تعریف کنید به راحتی می توانید در آدرس زیر پوشه ای ایجاد کرده و فایل هایتان را در آن قرار دهید. سپس دوباره با استفاده از روش بالا به سایتتان دسترسی پیدا کنید. اگرچه باز هم این پوشه به صورت مجازی تعریف می شود.

C:\Inetpub\wwwroot\

**مثال کلیشه ای Hello world!**

اجازه دهید ما هم مثل بقیه با این مثال ساده شروع کنیم.

```
<HTML>
    <HEAD>
        <TITLE>Hello, World! </TITLE>
    </HEAD>
    <BODY>
        <% Response.Write “Hello, World!” %>
    </BODY>
</HTML>
```

همان طور که می بینید یک خط VBScript را درون تگ های باز و بسته قرار دادیم.

```
Response.Write “Hello, World!”
```

این جمله رشته ی Hello, World! را در صفحه چاپ می کند.

**راهی دیگر**

اجازه دهید را کوتاه دیگری را پیشنهاد کنیم.

```
<HTML>
    <HEAD>
        <TITLE>Hello, World! </TITLE>
    </HEAD>
    <BODY>
        <%="Hello, World!" %>
    </BODY>
</HTML>
```

توجه کنید که علامت = درست بلافاصله بعد از %> نیز همان کار Response.write را برای ما انجام می دهد.

**نمایش تاریخ**

اجازه دهید یک قدم پیش رفته و صفحه ای بسازیم که تاریخ امروز را به ما نشان دهد.

```
<HTML>
    <HEAD>
        <TITLE>Hello, World !</TITLE>
    </HEAD>
    <BODY> <%= Date %> </BODY>
</HTML>
```

استفاده از تابع Date تاریخ جاری را بر می گرداند. همچنین تابع Time زمان فعلی را بر می گرداند. اگر می خواهید هم زمان هم تاریخ فعلی را در اختیار داشته باشید می توانید از تابع Now استفاده کنید. به اطلاعات دقیق تری نیاز دارید؟

همچنین می توانید عناصر سال، ماه، ساعت، دقیقه و ثانیه را با توابع زیر به دست آورید.

Year, month, hour, minute, second

کد زیر را نگاه کنید.

```
<HTML>
```

```
    <HEAD>
        <TITLE>Hello, World !</TITLE>
    </HEAD>
    <BODY>
    <%
    Response.Write “Year: ” & Year (Now)
    Response.Write “Month: ” & Month (Now)
    Response.Write “MonthName: ” & MonthName (Month(Now))
    Response.Write “Hour: ” & Hour (Now)
    Response.Write “Minute: ” & Minute (Now)
    Response.Write “Second: ” & Second (Now)
    %>
    </BODY>
</HTML>
```

به ترکیب متن با VBScript توجه کنید. با این مقدمه به بررسی متغییرها، ثابت ها و ساختارهای گوناگون می پردازیم.

**متغییر ها و ساختار ها**

در VBScript نیز برای تعریف متغییر همانند vb از کلمه ی Dim استفاده می شود.

```
<%
Dim  myVar
%>
```

برنامه نویسان vb توجه کنند که در اینجا به نوع متغییر هیچ اشاره ای نکردیم.مثلا نگفتیم

```
Dim st as string
```

در VBScript تمام متغییر ها از نوع Variant در نظر گرفته می شوند. نوع آنها به صورت اتوماتیک در زمان اجرا توسط مفسر تعیین می شود و برنامه نویس نیاز به تعیین نوع ندارد.

به صورت پیش فرض VBScript شما را مجبور به تعریف متغییر نمی کند. اما تمام برنامه نویسان حرفه ای می دانند که اگر از متغییری بدون اینکه آن را تعریف کرده باشند استفاده کنند ممکن است چه عواقبی برایشان داشته باشد. Error هایی که بسیار سخت آنها را پیدا می کنید. پس برای اینکه خودمان را مجبور به تعریف متغییر هایمان کنیم از دستور Option Explicit در ابتدای اسکریپتمان استفاده می کنیم.

```
<%
Option Explicit
Dim  myVar
%>
```

همیشه به یاد داشته باشید که Option Explicit لزوما باید اولین جمله ی صفحه تان باشد در غیر این صورت یک خطای سرور ایجاد می شود. برای اینکه منظورم را متوجه شوید مثال زیر را نگاه کنید.

```
<%
Pi = 3.141592654
Response.Write  Pi
%>
```

این دقیقا یک صفحه ی کامل است. همان طور که انتظار دارید مقدار Pi چاپ می شود. حال کد بالا را با استفاده از Option Explicit می نویسیم.

```
<%
Option Explicit
Pi = 3.141592654
Response.Write Pi
%>
```

حالا خطایی ایجاد می شود که به صورت زیر است.

```
Microsoft VBScript runtime (0x800A01F4)
Variable is undefined: 'Pi'
/asp/test.asp, line 3
```

دلیلش این است که چون از Option Explicit استفاده کرده ایم IIS تنها متغییر هایی را می شناسد که تعریف کرده باشیم. به هر حال در این مثال اسکریپت درست به صورت زیر می باشد.

```
<%
Option Explicit
Dim  Pi
Pi = 3.141592654
Response.Write  Pi
%>
```

ساده ترین ساختار که در تمام زبان ها یافت می شود ساختار تصمیم If-Then-Else است. با یک مثال شروع می کنیم.

```
<%    If OK = True Then
            Response.Write “OK”
      Else
            Response.Write “Error”
      End If   %>
```

نکات کلیدی:

- شرط بعد از If بر خلاف زبان های دیگر مثل c یا java حتما باید Then را به همراه داشته باشد.
- اگر تنها یک جمله باید در بلوک If قرار گیرد می توانید آن جمله را بلافاصله بعد از Then بنویسید. اما اگر بلوکی از دستورات را دارید باید جملات از خط بعد از Then شروع شوند.
- بلوک Else همانند دیگر زبان ها اختیاری است.
- بلوکی از دستورات باید با End if خاتمه یابد.

مثال های زیر استفاده های درست و نادرست از ساختار If را نشان می دهند.

```
<%
```

```
If OK = True Then Response.Write “OK” Else Response.Write “Error”
%>
```

این مثال درست است. از آنجایی که تنها یک جمله در هر بلوک Then و Else قرار دارد نیازی به End if نیست. به علاوه جمله ی Else نیز باید در همان خط باشد.

```
<%
If OK = True Then Response.Write “OK”
Else Response.Write “Error”
%>
```

این مثال نادرست است. چون یک دستور داریم و دستور بالایی در جلوی Then نوشته شده جمله ی پائین نیز باید در همان خط قرار گیرد.

یک قرارداد کلی برای خودتان استفاده از یک نوع If به صورت زیر است.

```
<%
If OK = True Then
        Response.Write “OK”
        ‘ ... Any more statements
Else
        Response.Write “Error”
        ‘ ... Any more statements
End If
%>
```

هر جمله ای که با ’ شروع می شود توضیح می باشد و کامپایل نمی شود. مثلا قسمتی از کد زیر اجرایی و قسمتی توضیح می باشد.

```
<%
OK = True  ‘Sets OK to True
%>
```

**حلقه های For-Next**

Syntax یا گرامر آن به صورت زیر می باشد

```
<% For I = 1 to 8
Response.Write “Number = ” & I & vbCrLf
Next %>
```

و خروجی

```
Number = 1
Number = 2
Number = 3
Number = 4
Number = 5
Number = 6
Number = 7
Number = 8
```

دستور vbcrlf که در جملات بالا استفاده شد ثابت از پیش تعریف شده ای می باشد که برابر ترکیبی از Carriage-Return و کاراکتر Line-Feed می باشد. استفاده از این ثابت باعث می شود که خروجی در خط بعدی ادامه یابد. (به جای این ثابت می توانستید از تگ <br> در HTML که معادل کلید Enter می باشد استفاده کنید)

```
<%
For I = 1 to 10
        Response.Write “Number = ” & I & "<br>"
Next
%>
```

بدون vbcrlf و <br> خروجی برنامه ی ما روی یک خط طولانی قرار می گیرد.

Number = 1Number = 2Number = 3Number = 4Number = 5Number =6Number = 7Number = 8Number = 9Number = 10

اجازه دهید مثالی برای حلقه های تو در تو بزنیم

```
<%
For I = 1 to 8
        For j =1 to 8
                Response.Write "X"
        Next
        Response.Write "<br>"
Next
%>
```

این کد یک الگوی صفحه ی شطرنج را روی صفحه درست می کند. البته که باید خروجی را در نمایش دهنده ی صفحات وبتان ببینید.

یک نکته ی مهم این است که کلمه ی Next که به بلاک حلقه ی For خاتمه می دهد هیچ آرگومانی را قبول نمی کند. مثلا شما نمی توانید بگوئید Next j

این اشتباه است. با هر جمله ی Next ای که روبرو می شود بلافاصله آن را برای انتهای بلاک دستور For قبلی خودش در کد در نظر می گیرد.

سرانجام VBScript همچنین اجازه ی استفاده از کلمه ی Step را برای تعیین گام حلقه می دهد.

```
<%
For I = 1 to 10 Step 2
        Response.Write “Number = ” & I & "<br>"
Next
%>
```

خروجی

```
Number = 1
Number = 3
Number = 5
```

Number = 7

Number = 9

در واقع شمارنده ی حلقه ی For اول برابر مقدار اولیه قرار می گیرد و دستورات داخل بلاک یک بار اجرا شده و با رسیدن به Next شمارنده به تعداد Step (اگر ننویسیم یک در نظر گرفته می شود) اضافه شده و اگر کمتر از مقدار پایانی حلقه بود دستورات داخل بلاک اجرا شده و ...

**ساختار For each** Object **In** Collection

ساختار For-each منحصر به VBScript و البته Visual Basic می باشد. که به شما اجازه می دهد که در میان اقلام یک مجموعه یک به یک حرکت کنید.

<% For Each **Member** in **Team**

Response.Write Member

Next %>

در اینجا Team مجموعه ای از اشیاء در نظر گرفته شده است. این دستور در متونی استفاده می شود که اندازه ی مجموعه معلوم نمی باشد. استفاده از این ساختار این اطمینان را می دهد که تمام اقلام داخل یک مجموعه پردازش می شود و هیچگاه با پیغام خطای زیر روبرو نمی شوید

"Array Index Out Of Bounds"

این خطا هنگامی ایجاد می شود که سعی کنید به اندیسی خارج از اندیس تعریف شده برای آرایه دسترسی پیدا کنید.

**ساختار While ... Wend**

یکی از مشهور ترین ساختار های حلقه در تمام زمان ها این حلقه می باشد.

<%

While Not RS.EOF

Response.Write RS.Fields ("Name")

```
        RS.MoveNext
Wend
%>
```

یا

```
<%
Do While Not RS.EOF
        Response.Write RS.Fields ("Name")
        RS.MoveNext
Loop
%>
```

جمله ی While جملات داخل بلاکش را تا زمانی که شرط برقرار است اجرا می کند و هنگامی که شرط نادرست شود اجرای حلقه به پایان می رسد.

به یاد داشته باشید که برای پایان بلاک جمله ی While از Wend استفاده کنید.

نوع دیگری از حلقه ی While که شرط حلقه را در پایان حلقه چک می کند حلقه Do-Loop می باشد.

```
<%
Do
        response.write x & "<br>"
        x = x + 1
loop Until x > 20
%>
```

اعداد 1 تا 20 را چاپ می کند.

### ساختار Select Case

برای انتخاب کردن از بین مجموعه ای از اقلامی که می توانند به یک متغییر اختصاص یابند از این ساختار استفاده کنید.

```
<%
```

```
Choice = 3
Select Case Choice
        Case 1
                Response.Write "You chose 1"
        Case 2
                Response.Write "You chose 2"
        Case 3
                Response.Write "You chose 3"
        Case 4
                Response.Write "You chose 4"
End Select %>
```

**روابط و حالات پیچیده**

ممکن است بیش از یک شرط را در ساختار های بالا استفاده کنید.

**And , Or , Not**

مثال

Condition1 **And** Condition2 **Or** Condition3

روابط شرطی کلمه های And , Or , Not می باشند. در اینجا دیگر از سمبل های زبان c مثل , && , ||

! استفاده نمی شود.

Not بالاترین تقدم و در ادامه And و در آخر Or می باشد. برای ایجاد تقدم های مورد نظرتان از پرانتز ها

استفاده کنید.

## ساختار FUNCTIONS , SUBROUTINES و INCLUDES

شبیه تمامی زبان های برنامه نویسی در VBScript هم می توانید تابع و زیر روال داشته باشید.

### زیر روال ها یا subroutines

زیر روال ها با کلمه ی sub تعریف می شوند.

```
<%
Sub SayHello()
        Response.Write “Hello !”
End Sub
%>
```

یک زیر روال ممکن است پارامترها را به عنوان ورودی دریافت کند.

```
<%    Sub SayHelloTo (Person)
            Response.Write “Hello, ” & Person & “!”
      End Sub  %>
```

پارامترها نوع های از پیش تعریف شده ای ندارند. طرز استفاده از آنها نوع آنها را مشخص می کند. نوع پیش فرض همه ی پارامتر ها به صورت پیش فرض variant می باشند.

زیر روال ها نمی توانند مقداری را بر گردانند. برای برگرداندن مقادیر از توابع استفاده می کنیم.

### توابع یا Functions

توابع نیز شبیه زیر روال ها تعریف می شوند.

```
<%
Function Add (A, B)
        Add = A + B
End Function
%>
```

همان طور که مشاهده می کنید تابع Add حاصل جمع دو مقدار A و B را در اسم خودش بر می گرداند.

برای فراخوانی این تابع به طریقه ی زیر عمل کنید.

```
<%
```

```
Response.Write Add (2, 3)
%>
```

مثال

```
<%
Function Calculate (A, B, Op)
        Select Case Op
        Case “+”
                Calculate = A + B
        Case “-”
                Calculate = A - B
        Case “*”
                Calculate = A * B
        Case “/”
                Calculate = A / B
        End Select
End Function
Response.Write Calculate(2, 3, “+”)
Response.Write Calculate(2, 3, “-”)
%>
```

**Includes**

Server Side Includes یا SSI یک زبان برنامه نویسی بسیار ساده می باشد که دارای تعداد دستورات محدودی می باشد. برای اضافه کردن فایلی به اسکریپتمان از روش زیر استفاده می کنیم.

```
<!-- #INCLUDE FILE=”filename.inc” -->
```

همانطوری که در C از #include <stdio.h> استفاده می کنیم در اینجا نیز این دستور را داریم.

هدف کاملا شبیه همان است. #INCLUDE فایل ذکر شده را به کد ما اضافه می کند به طوری که هر اسکریپتی که در آن فایل حاضر باشد اجرا می شود.

فایل های Include معمولا برای ذخیره ی توابعی که به صورت گسترده استفاده می شوند به کار می رود. مثل تابعی برای چک کردن درستی email ها. شما می توانید چنین توابعی را در یک فایل بنویسید و فقط آن یک فایل را به تمامی صفحاتی که می خواهید اضافه کنید.

همچنین می توانید از این توابع برای وارد کردن Header و Footer برای هر صفحه ای استفاده کنید. برای این کار می توانید تمامی کد ها را در یک فایل قرار دهید و فقط آن یک فایل را به هر صفحه ای که می خواهید اضافه کنید. بنابراین نیازی به Copy و Paste های زیاد در هر صفحه ای ندارید ، ویرایش آسان می شود ، بنابراین می توانید فقط یک فایل را تغییر دهید و نگران ویرایش تمام صفحات نباشید.

راه دیگر استفاده از INCLUDE استفاده از کلمه ی virtual می باشد.

<!-- #INCLUDE VIRTUAL="/directory/file.inc" -->

که فایل را با توجه به آدرس مجازیش اضافه می کند. با توجه به خط بالا انتظار می رود فایل در آدرس زیر باشد

www.domain.com/directory/file.inc

**مدل شئ (THE OBJECT MODEL)**

ASP محیط کد نویسی بر پایه ی مدل های اشیاء می باشد. یک مدل شیء به سادگی یک سلسله مراتبی از اشیاء می باشد که ممکن است شما از آنها سرویس دریافت کنید.

**Scripting Context**

Request
Response
Server
Session
Application

**Request**: برای گرفتن اطلاعات از کاربر

**Response**: برای فرستادن اطلاعات به کاربر

**Server**: برای کنترل IIS

**Session**: برای ذخیره ی اطلاعات و تعویض تنظیمات برای دوره ی وب سرور کاربر فعلی

**Application**: برای به اشتراک گذاشتن اطلاعات و تنظیمات برنامه ها برای طول زمان زندگی یک برنامه

اشیاء Request و Response شامل مجموعه می باشند (بیت های اطلاعاتی که با همان روش پردازش می شوند). اشیاء از method برای انجام روال های گوناگون (اگر یکی از زبان های شیءگرا را بشناسید می دانید یک متد چیست) و خواص برای نگه داری خواص اشیاء مثل رنگ ، اندازه و ... استفاده می کنند.

**شیء Request**:

این شیء مقادیری که browser کلاینت به سرور در طول یک درخواست HTTP پاس شده است را نگه می دارد.

Request[.collection|property|method](variable)

**مجموعه ها (Collections)**:

ClientCertificate

برای گرفتن فیلد های گواهینامه ( Certification) از درخواست صادر شده به وسیله ی نمایش دهنده ی وبتان. فیلد هایی را می توانید درخواست کنید که از استاندارد X.509 استفاده می کنند.

Cookies

مقادیر cookie های فرستاده شده در درخواست HTTP

Form

مقادیر عناصر فرم در بدنه ی درخواست HTTP

QueryString

مقادیر متغییر ها در رشته ی Query ی HTTP

ServerVariables

مقادیر متغییر های محیطی از پیش تعیین شده.

**خواص (Properties):**

TotalBytes

این خاصیت به صورت Read-only می باشد. تعداد تمام بایت هایی را که کلاینت در بدنه ی درخواست در حال فرستادن می باشد را مشخص می کند.

**متد ها (Methods):**

BinaryRead

اطلاعات فرستاده شده به سرور از طرف کلاینت به صورت POST را دریافت می کند. پارامتر های متغییر رشته هایی هستند که عنصری را که باید از مجموعه دریافت شود یا باید مثل یک ورودی برای یک متد یا خاصیت مورد استفاده قرار گیرد را مشخص می کنند.

**نکته:**

تمام متغییر ها می توانند بدون نام مجموعه با فراخوانی Request(variable) مستقیما مورد دسترسی قرار گیرند. در این حالت سرور وب مجموعه ها را به ترتیب زیر جستجو می کند:

- QueryString
- Form
- Cookies
- ClientCertificate
- ServerVariables

اگر متغییر هایی هم نام در بیش از یک مجموعه وجود داشته باشد ، شیء Request اولین نمونه ای را که با آن مواجهه می شود بر می گرداند.

به شدت پیشنهاد می شود وقتی می خواهید به اعضای مجموعه ی ServerVariables دسترسی پیدا کنید از نام کامل آن استفاده کنید. برای مثال به جای Request.(AUTH_USER) از Request.ServerVariables(AUTH_USER) استفاده کنید.

**Request.ServerVariables**

لیست کاملی از تمام متغییر های سرور در زیر آورده شده است:

| متغییر | مفهوم |
|---|---|
| ALL_HTTP | هدر های HTTP از کلاینت |
| ALL_RAW | هدر های HTTP ی نارس (RAW) از کلاینت |
| APPL_MD_PATH | مسیر متابیس فایل ISAPI DLL |
| APPL_PHYSICAL_PATH | مسیر فیزیکی به متابیس |
| AUTH_PASSWORD | کاربر چه چیزی را در پنجره ی تصدیق کلاینت وارد کرده |

| | |
|---|---|
| | است |
| AUTH_TYPE | تصدیق استفاده شده ی سرور |
| AUTH_USER | شناسه ی کاربر تصدیق شده |
| CERT_COOKIE | شناسه ی واحد گواهینامه ی کلاینت |
| CERT_FLAGS | آیا گواهی کلاینت معتبر می باشد؟ |
| CERT_ISSUER | توزیع کننده ی گواهی نامه ی کاربر |
| CERT_KEYSIZE | تعداد بیت ها در کلید SSL |
| CERT_SECRETKEYSIZE | تعداد بیت ها در کلید محرمانه |
| CERT_SERIALNUMBER | شماره ی سریال گواهینامه ی کلاینت |
| CERT_SERVER_ISSUER | توزیع کننده ی گواهی نامه ی سرور |
| CERT_SERVER_SUBJECT | موضوع گواهینامه ی سرور |
| CERT_SUBJECT | موضوع گواهینامه ی کلاینت |
| CONTENT_LENGTH | طول محتوا |
| CONTENT_TYPE | نوع MIME صفحه ی جاری |
| GATEWAY_INTERFACE | نسخه ی CGI (Common Gateway Interface) از سرور |
| HTTPS | آیا تمام SSL (Secure Socket Layer) ایمن می باشد |
| HTTPS_KEYSIZE | تعداد بیت ها در کلید SSL |
| HTTPS_SECRETKEYSIZE | تعداد بیت ها در کلید های مخفی |
| HTTPS_SERVER_ISSUER | توزیع کننده ی گواهینامه ی سرور |
| HTTPS_SERVER_SUBJECT | موضوع گواهینامه ی سرور |

| | |
|---|---|
| INSTANCE_ID | ID برای این نمونه در IIS |
| INSTANCE_META_PATH | مسیر متابیس برای این نمونه |
| LOCAL_ADDR | IP سرور |
| LOGON_USER | نام login ویندوز NT برای کاربر جاری |
| PATH_INFO | مسیر مجازی سرور |
| PATH_TRANSLATED | مسیر کامل سرور |
| QUERY_STRING | جفت اسم و مقدار متغییر های رشته ی URL |
| REMOTE_ADDR | آدرس IP کلاینت برای ماشین درخواست کننده |
| REMOTE_HOST | آدرس IP کلاینت برای ماشین میزبان |
| REMOTE_USER | کاربر از راه دور |
| REQUEST_METHOD | شیوه ی درخواست |
| SCRIPT_NAME | مسیر مجازی و نام فایل برای اسکریپت جاری |
| SERVER_NAME | نام سرور |
| SERVER_PORT | پورت در حال بازرسی |
| SERVER_PORT_SECURE | 0: نا امن 1: امن |
| SERVER_PROTOCOL | نام و نسخه ی پروتکل مورد استفاده |
| SERVER_SOFTWARE | برنامه ی HTTP مورد استفاده ی سرور |
| URL | URL بدون نام Domain |
| HTTP_ACCEPT | انواع MIME های شناخته شده برای Browser تان |
| HTTP_ACCEPT_LANGUAGE | تنظیمات زبان Browser |
| HTTP_CONNECTION | ارتباط HTTP |

| | |
|---|---|
| HTTP_HOST | Domain میزبان این درخواست |
| HTTP_USER_AGENT | Browser مورد استفاده |
| HTTP_PRAGMA | صفحه Cache شده یا نه؟ |
| HTTP_COOKIE | Cookie مربوط به این صفحه |
| HTTP_ACCEPT_CHARSET | مجموعه کاراکتر های ISO ی مورد قبول |

**شئ Response**:

از شئ Response برای فرستادن اطلاعات به کاربر استفاده می شود. شئ Response فقط کوکی ها را به صورت یک مجموعه پشتیبانی می کند. همچنین تعدادی از خواص ( properties) و متد ( Methods) را پشتیبانی می کند.

**فرم کلی**

Response.collection|property|method

**مجموعه ها (Collections)**:

Cookies

برای مقدار دهی به کوکی ها استفاده می شود.

**خواص (Properties)**:

Buffer

نشان می دهد که آیا خروجی صفحه بافر شده است یا نه؟!

CacheControl

نشان می دهد که آیا سرور های پروکسی ( proxy) توانایی کش کردن ( cache) خروجی ساخته شده با ASP را دارند یا نه؟!

Charset

نام مجموعه کاراکتر ها (character set) را به هدر Content-Type اضافه می کند.

ContentType

Content-Type را برای response مشخص می کند.

Expires

مدت زمان را قبل از این که یک صفحه ی کش شده در Browser از بین برود (Expire شود) را مشخص می کند.

ExpiresAbsolute

زمان و تاریخی را که یک صفحه ی کش شده در Browser از بین برود ( Expire شود) را مشخص می کند.

IsClientConnected

نشان می دهد که آیا کاربر (کلاینت) از سرور قطع (Disconnect) شده است یا خیر؟!

Pics

مقادیر برچسب های PICS را در هدر Response به فیلد pics-label اضافه می کند.

Status

مقدار status line بر گشت داده شده توسط سرور.

**متد ها (Methods):**

AddHeader

نام هدر HTML را به مقدار آن نسبت می دهد.

AppendToLog

رشته ای را به پایان فایل log سرور وب  برای این درخواست اضافه می کند.

BinaryWrite

اطلاعات داده شده را در خروجی HTTP جاری بدون هیچ تبدیل Character-set ای می نویسد.

Clear

هر خروجی HTML بافر شده ای را پاک می کند.

END

اجرای فایل asp. را متوقف ساخته و نتیجه ی کار را بر می گرداند.

Flush

خروجی بافر شده را سریعا می فرستد.

Redirect

یک پیغام redirect را به Browser می فرستد ، تا برای وصل شدن به URL ای دیگر تلاش کند.

Write

یک متغییر را در خروجی HTTP جاری به صورت رشته می نویسد. برای این کار می توان هم از ساختار زیر استفاده کرد:

```
Response.Write("Hello")
```

و یا از روش کوتاه تر زیر:

```
<%= "Hello" %>
```

**شئ Server**:

شئ سرور دستیابی به متدها و خواص را در سرور فراهم می آورد. بیشتر این متدها و خواص به صورت توابع utility به کار می روند.

فرم کلی

```
Server.property|method
```

**خواص (Properties)**:

ScriptTimeout

مقدار زمانی که یک اسکریپت قبل از اینکه زمانش به اتمام برسد می تواند اجرا شود.

**متد ها (Methods):**

CreateObject

یک نمونه از یک کامپوننت سرور ایجاد می کند. این کامپوننت می تواند هر کامپوننتی که روی سرور نصب کرده اید باشد. (مثل یک ActiveX)

HTMLEncode

رمز گذاری HTML را روی رشته ی مشخص شده انجام می دهد.

MapPath

مسیر مجازی مشخص شده را به مسیر فیزیکی روی سرور تبدیل می کند.

URLEncode

قوانین رمز گذاری URL را روی رشته ی مشخص شده انجام می دهد.

**شئ Session:**

می توانید از Session برای ذخیره ی اطلاعات لازم برای یک کاربر خاص استفاده کنید. متغییر های ذخیره شده در این شئ هنگامی که از صفحه ای به صفحه ی دیگر می روید از بین نمی روند. در عوض این متغییر ها برای تمام user-session باقی می ماند.

سرور وب به صورت اتوماتیک ، هنگامی که یک صفحه به وسیله ی کاربری که قبلا Session ی نداشته ، یک Session ایجاد می کند. سرور شئ session را هنگامی که Expire یا Abandon می شود ویران می کند.

یک استفاده ی مشترک برای شئ Session برای ذخیره ی تنظیمات کاربر می باشد. برای مثال اگر کاربری تنظیم کرده که گرافیکی را نبیند می توانید این تنظیم را در یک Session ذخیره کنید.

Session تنها در BROWSER هایی که کوکی ها را پشتیبانی می کنند استفاده می شود.

فرم کلی:

Session.collection|property|method

**مجموعه ها (Collections):**

Contents:

شامل اقلامی می باشد که با دستورات اسکریپت به شئ session اضافه کرده اید.

StaticObjects:

شامل اشیائی می باشد که با تگ <OBJECT> ساخته و به آنها میدان قلمرو داده شده است.

**خواص (Properties):**

CodePage:

Codepage ای که برای نگاشت نماد ها به کار می رود.

:LCID

مشخصه ی منطقه را مشخص می کند.

:SessionID

شناسه ی session را برای کاربر فعلی بر می گرداند.

:Timeout

زمان Timeout برای session این برنامه بر حسب دقیقه.

**متد ها (Methods):**

:Abandon

این متد یک session را از بین برده و منابع آن را آزاد می کند.

رویداد ها (Events):

اسکریپت های زیر در فایل Global.asa تعریف شده اند:

Session_OnEnd
Session_OnStart

**شئ Application:**

این شئ می تواند اطلاعاتی را که در کل زمان یک برنامه ایستا می باشند را در خود ذخیره کنند. در واقع در یک مجموعه ای از صفحات با یک ریشه ی مشترک. معمولا این کل زمانی می باشد که سرور IIS در حال

اجراست. این شئ جایی مناسب برای ذخیره ی اطلاعاتی می باشد که باید برای بیشتر از یک کاربر وجود داشته باشد. برای مثال یک Page counter. اشکال این شئ این است که این شئ برای هر یک کاربر جدید ساخته نمی شود. همچنین به دلیل این که این شئ به وسیله ی کل کاربران به اشتراک گذاشته می شود اجرای بند کشی (threading) می تواند یک کابوس تلقی شود.

شما می توانید از این شئ برای به اشتراک گذاشتن اطلاعات میان تمام کاربران یک برنامه ی داده شده استفاده کنید. یک برنامه ی بر مبنای ASP به صورت تمام فایل های **asp**. در یک پوشه ی مجازی و تمام زیر مجموعه های آن تعریف می شود. به دلیل این که این شئ می تواند بین بیشتر از یک کاربر به اشتراک گذاشته شود متد های Lock و Unlock برای اطمینان از این که چندین کاربر سعی نمی کنند که به طور همزمان یک خاصیت را تغییر دهند وجود دارند.

فرم کلی:

Application.method

**مجموعه ها (Collections):**

Contents:

شامل اقلامی می باشد که با دستورات اسکریپت به شئ Application اضافه کرده اید.

StaticObjects:

شامل اشیائی می باشد که با تگ <OBJECT> ساخته و به آنها میدان قلمرو داده شده است.

Lock:

این متد از ایجاد تغییر کلاینت ها بر روی خواص شئ Application جلوگیری می کند.

:Unlock

این متد اجازه می دهد دیگر کلاینت ها خواص شئ Application را تغییر دهند.

رویداد ها (Events):

اسکریپت های زیر در فایل Global.asa تعریف شده اند:

Application_OnEnd

Application_OnStart

**بررسی اطلاعات ورودی کاربر (FORM ها و QUERYSTRING ها)**

یک زبانی که به شما اجازه نمی دهد اطلاعات ورودی کاربر را به طور موثر بخوانید چقدر خوب است!

HTML ، زبان خوب قدیمی برای کاربر فرم هایی را برای ورود اطلاعاتش می سازد و شما به عنوان یک برنامه نویس ASP می توانید اسکریپت هایی را برای پردازش آن اطلاعات بنویسید.

**مجموعه ی Request.Form:**

وقتی فرم html ای به صورت زیر دارید:

```
<FORM METHOD=”post” ACTION=”process.asp”>
<INPUT TYPE=”text” NAME=”FirstName”>
<INPUT TYPE=”text” NAME=”LastName”>
<INPUT TYPE=”radio” NAME=”Sex” VALUE=”M”>
<INPUT TYPE=”radio” NAME=”Sex” VALUE=”F”>
<TEXTAREA NAME=”Address”>
</TEXTAREA>
<INPUT TYPE=”submit” VALUE=”Send”>
```

</FORM>

تعدادی از اشیاء را با یک نام واحد برای هر کدام دارید. فیلد ها در فرم بالا

FirstName(Text), LastName(Text), Sex(Option: M or F), Address(Multiline Text)

می باشند. آخرین نوع تگ Input دستور Submit می باشد که دکمه ای برای عمل submit فرم ها می باشد. با کلیک بر روی دکمه ی submit محتویات هر کدام از این فیلد ها به اسکریپتی که در خاصیت فرم مشخص کرده اید پست می شوند. در مثال بالا فایل "process.asp" می باشد.

اسکریپت پردازش فرم می تواند شبیه زیر باشد:

Request.Form (“FirstName”)

Request.Form (“LastName”)

یکبار که این مقدار را می گیرید همان طور که می خواهید می توانید آن را پردازش کنید – به بانک اطلاعاتیتان اضافه کنید ، به خودتان mail کنید و یا هر کاری که دوست دارید با آن انجام دهید.

توجه داشته باشید که خاصیت METHOD در تگ فرمتان حتما باید برابر POST باشد تا بتوانید از Request.Form استفاده کنید.

**مجموعه ی Request.QueryString:**

احتمالا به کرات صفحاتی را با URL هایی شبیه خط زیر دیده اید:

http://www.greetings.com/show.asp?CardID=128762173676

این می تواند آدرس یک کارت تبریک باشد که دوستتان برای شما فرستاده. فقط لازم است که بر روی این لینک کلیک کرده تا کارتتان را مشاهده کنید. لازم نیست که هیچ اطلاعاتی را در جایی از سایت وارد کنید بلکه تمام آن چیز هایی که لازم است در رشته زیر کد گذاری شده است.

CardID=128762173676

این رشته به نام یک Query String شناخته می شود. همچنین می توانید چندین مقدار را با چیزی شبیه زیر پاس کنید.

Page.asp?FirstName=Manas&LastName=Tungare&Sex=M

مجموعه ی Request.QuerySting به شما کمک می کند که مقادیری را که لازم دارید از این رشته استخراج کنید- در واقع مقادیر متغییرهای آن ها را.

بنابراین برای دسترسی به محتویات فیلد FirstName در مثال بالا می توانید مثل زیر عمل کنید. البته به شرطی که خاصیت METHOD را در تگ فرم به صورت GET تغییر دهید.

Request.QueryString (“FirstName”)

می توانید این خط را به متغییری انتساب داده و یا در محاسبات از آن استفاده کنید.

مجموعه ی Request.QueryString به شما اجازه می دهد که به متغییر ها (فیلد ها) دسترسی پیدا کنید – البته آنهایی که با متد get در تگ فرم پست شده اند. اگرچه محدودیتی در میزان اطلاعاتی که می توانند با این روش پست شوند وجود دارد ولی برای اکثر مواقع روش خوبی است. اگر لازم بود اطلاعات زیادی را پست کنید از متد POST در تگ فرم استفاده کنید.

**در چه مواقعی از GET استفاده کنیم؟**

هر داده ای را که با این متد پست می کنید با Request.QueryString در اسکریپتتان قابل دسترسی است. GET اصولا برای مقادیر کوچک به کار می رود – علت این است که اقلام داده به URL ای که Browser تان نشان می دهد اضافه می شود. شما نمی توانید URL ای با طول نامحدود با QueryString داشته باشید.

**در چه مواقعی از POST استفاده کنیم؟**

تقریبا همیشه. متد POST را برای فرم هایتان مشخص کنید و اطمینان پیدا کنید که برای دسترسی به آنها از Request.Form استفاده کنید.

## دستکاری داده ها با ASP

در این قسمت نگاهی به یک مثال کاربردی با ASP و ADO برای ساختن سایتی که با بانک اطلاعاتی کار می کند داریم. ابتدا با Query های ساده شروع و به مثال های پیچیده تر هم می رسیم.

### نمایش اطلاعات از یک جدول:

اجازه دهید مثالی از یک بانک اطلاعاتی دانشجویان یک کلاس داشته باشیم.

شکل بانک اطلاعاتی به صورت زیر می باشد:

| Table: **Student** | |
|---|---|
| ID | شماره ی شناسایی دانشجو (کلید اصلی) |
| FirstName | نام |
| LastName | نام خانوادگی |
| DateofBirth | تاریخ تولد |
| Email | پست الکترونیکی |

### بازیابی داده ها:

برای خواندن داده ها از یک جدول لازم است که از جمله ی SELECT زبان SQL استفاده کنید.

برای نمونه می خواهیم لیست کاملی از تمام دانشجویان در کلاس را نمایش دهیم. در زیر صفحه ی کاملی که تمام رکورد های دانشجویان کلاس را نشان می دهد آمده است:

```
<HTML>
    <HEAD>
        <TITLE>Student Records</TITLE>
    </HEAD>
    <BODY>
        <%
        Dim DB
        Set DB = Server.CreateObject (“ADODB.Connection”)
        DB.Open          ("PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA
        SOURCE=" + “C:\Databases\Students.mdb”)
        Dim RS
        Set RS = Server.CreateObject (“ADODB.Recordset”)
        RS.Open “SELECT * FROM Students”, DB
        If RS.EOF And RS.BOF Then
            Response.Write “There are 0 records.”
        Else
            RS.MoveFirst
            While Not RS.EOF
            Response.Write RS.Fields (“FirstName”) & " "
            Response.Write RS.Fields (“LastName”)
            Response.Write “<HR>”
            RS.MoveNext
            Wend
        End If
        RS.Close
```

```
            set RS=nothing
            DB.Close
            set DB=nothing
            %>
        </BODY>
</HTML>
```

اجازه دهید خط به خط به مثال نگاه کنیم. جند خط اولتگ های باز کننده ی HTML می باشندکه باید برای تمام صفحات به کار روند. تا اینجا هیچ اسکریپت ASP ای وجود ندارد.بلاک ASP با خط زیر شروع می شود.

Dim DB

که تعریف متغییری می باشد که می خواهیم بعدا با آن کار کنیم. خط زیر

Set DB = Server.CreateObject (“ADODB.Connection”)

دو کار زیر را انجام می دهد:

جمله ی Server.CreateObject() برای ایجاد یک نمونه از یک شئ COM که ADODB.Connection می باشد استفاده می شود. سپس کلمه ی Set این منبع را بهمتغییر ما اختصاص می دهد. در واقع برای اینکه بتوانیم از یک شئ جدید استفاده کنیم باید برای آن از سیستم حافظه بگیریم که این کار را با Set انجام می دهیم. اکنون از این شئ برای ایجاد ارتباط با یک بانک اطلاعاتی با استفاده از یک رشته ی اتصال استفاده می کنیم.

"PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA SOURCE=" +
“C:\Databases\Students.mdb”

رشته ی بالا عبارتی می باشد که جای بانک اطلاعاتی را مشخص کرده و مهمتر این که نوع آن را (در واقع درایور مورد نیاز برای اتصال به آن) مشخص می کند. که می تواند بانک Access ، Sybase ، Oracle و

یا ... باشد. توجه داشته باشید که این خط مشخص می کند که می خواهیم به یک بانک Access 2000 متصل شویم. این مثال از یک ODBC استفاده نمی کند.

اگر جمله ی DB.open بدون هیچ خطایی به انجام برسد یک ارتباط درست با بانک اطلاعاتی داریم. تنها زمانی می توانیم با بانک اطلاعاتی کار کنیم که چنین ارتباطی بر قرار باشد.

```
Dim RS
Set RS = Server.CreateObject ("ADODB.Recordset")
```

اکنون نیز مثل بالا شیئی برای مجموعه رکورد هایمان ایجاد می کنیم. شما هر پرس و جویی را که از بانک اطلاعاتی می گیرید رکورد ها تحت یک مجموعه مثل جدول برگشت داده می شوند که به آن Recordset گفته می شود.

```
RS.Open "SELECT * FROM Students", DB
```

خط بالا تقریبا مهمترین قسمت مثال می باشد. با جمله SQL ای که مشخص می کنید بانک اطلاعاتی مورد پرس و جو قرار می گیرد و سپس رکورد های برگشتی را به شئ Recordset ما که در این مثال RS می باشد اختصاص می دهد. همان طور که می بینید RS.open آرگومان های دیگری نیز دارد که در واقع ارتباط فعالی را که با بانک داریم مشخص می کند. البته که آرگومان های دیگری نیز وجود دارد که آنها اختیاری می باشند و در اینجا لازم نیست که آنها را مقدار دهیم.

حال با فرض این که تمامی رکورد های مورد نظرمان در شئ Recordset مان قرار دارند به نمایش آنها می پردازیم.

```
If RS.EOF And RS.BOF Then
Response.Write "There are 0 records."
```

تقریبا در تمامی پرس و جو ها ممکن است هیچ اطلاعاتی در رکوردست وجود نداشته باشد. بنابراین خط بالا کنترل مهمی روی ایجاد خطای وجود نداشتن رکورد در رکوردست انجام می دهد. هنگامی که هیچ رکوردی برگشت داده نشود هر دوی Recordset.BOF و Recordset.EOF برابر True می شوند. بنابراین می توانید با نوشتن جمله ی شرطی If این خطا را کنترل کنید. اگر این خطا کنترل نشود ممکن است در ادامه ی اسکریپت با خطا ایجاد شوید.

```
Else
RS.MoveFirst
While Not RS.EOF
Response.Write RS.Fields (“FirstName”)
Response.Write RS.Fields (“LastName”)
Response.Write “<HR>”
RS.MoveNext
Wend
End If
```

RS.MoveFirst متدی می باشد که مکان نمای رکورد را به اولین رکورد مجموعه انتقال می دهد. به صورت پیش فرض ، ممکن است مکان نما به جای درستی نرود بنابراین لازم است قبل از هر عملی مکان نمای رکورد را به جای درستی ببرید. سپس یک حلقه ی While داریم که با آن به تمام رکورد های مجموعه رکوردهایمان دسترسی پیدا می کنیم. شرطی را که در حلقه چک می کنیم Not RS.EOF می باشد. یعنی تا وقتی به پایان مجموعه رکورد نرسیده ایم حلقه ادامه داشته باشد.

RS.Fields("FirstName") مقدار فیلد FirstName رکورد جاری (جایی که مکان نمای مجموعه رکورد ها آنجاست) می باشد.