Ajax : تحولی بزرگ در عرصه وب (بخش چهارم)

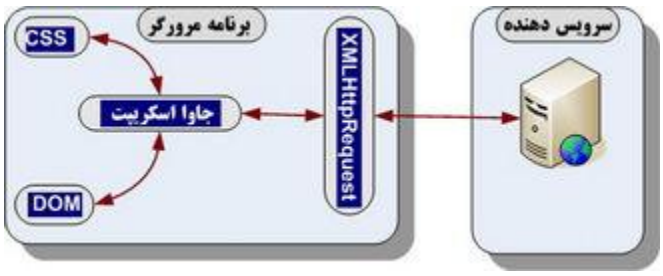
آنچه تاکنون گفته شده است :

بخش اول تاثیر متقابل وب و نرم افزار بر یکدیگر
بخش دوم Ajax و فناوری های مرتبط با آن
بخش سوم بررسی نمونه برنامه های مبتنی بر Ajax

در این بخش قرار بود که در رابطه با فریمورك های مختلف ارائه شده جهت بکارگیری فناوری Ajax آشنا شویم . ولی به دلیل درخواست تعداد زیادی از خوانندگان مبنی بر آشنائی بیشتر با معماری Ajax ، برنامه نویسی غیرهمزمان در برنامه های وب و شی XMLHttpRequest ، این بخش را به بررسی موارد فوق اختصاص دادیم تا علاقه مندان بتوانند قبل از پرداختن به اصل موضوع با برخی مفاهیم کلیدی و مهم بیشتر آشنا شوند .

مقدمه

Ajax یك رویکرد و یا الگوی جدید برای پیاده سازی برنامه های وب است که در آن از اسکریپت های سمت سرویس گیرنده برای مبادله داده با سرویس دهنده وب استفاده می گردد. رویکرد فوق باعث می شود که صفحات وب بدون نیاز به refresh کامل بتوانند بطور پویا بهنگام گردند (رویائی برای پیاده کنندگان برنامه های وب) . مهمترین دستاورد رویکرد فوق ، ارتباط بدون وقفه و پیوسته کاربران با برنامه های وب است .
برخی از کارشناسان بر این اعتقاد هستند که رویکرد فوق بیش از آن که یك الگو باشد یك فناوری است . در واقع ، Ajax ترکیبی از مجموعه فناوری های مرتبط به هم است که از آنها با یك نگرش جدید در جهت تولید نسل جدیدی از برنامه های وب استفاده می گردد .
نام بردن از فناورهائی که در Ajax از آنها استفاده می گردد کار مشکلی نیست ولی مهم این است که بدانیم این فناورها در کنار یکدیگر به چه صورت کار می کنند و هر یك از آنها در Ajax دارای چه مختصاتی است .شکل 1 ، نحوه تعامل و ارتباط این فناوری ها را از منظر مرورگر نشان می دهد .



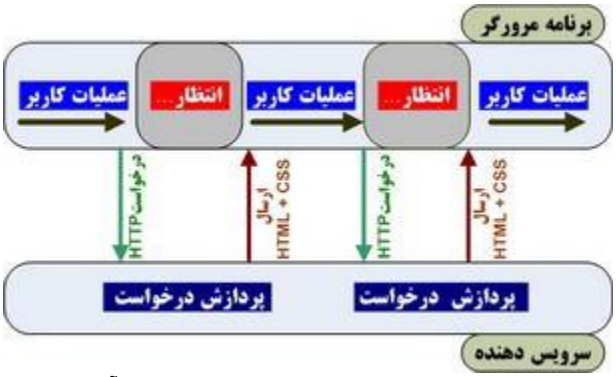
شکل 1 : عناصر Ajax

جاوا اسکریپت در Ajax دارای یك نقش محوری و تعیین کننده است و می توان آن را به منزله یك نیروی چسبنده در نظر گرفت که سایر فناوری ها را با هم مرتبط می نماید . زمانی که یك برنامه به داده نیاز داشته باشد ، از شی XMLHttpRequest به منظور ایجاد درخواست به سرویس دهنده استفاده می گردد . پس از برگرداندن داده توسط سرویس دهنده ، از فناورهای DOM (برگرفته شده از Document Object Model) و CSS (برگرفته شده از cascading style sheets) برای بهنگام سازی رابط کاربر مرورگر به صورت پویا استفاده می گردد .

برنامه نویسی وب غیرهمزمان

حرف A موجود در Ajax از Asynchronous گرفته شده است که در زبان فارسی به غیرهمزمان و یا ناهمگام ترجمه می شود و بیانگر یکی از قابلیت های مهم و کلیدی الگوی برنامه نویسی Ajax است .
در برنامه های وب سنتی ، تعامل کاربر با برنامه بطور پیوسته نبوده و در مقاطع زمانی خاصی لازم است کاربر در انتظار اتمام یك عملیات باشد . زمانی که کاربر عملیات خاصی نظیر کلیك بر روی دکمه موجود بر روی یك فرم را انجام می دهد ، یك درخواست مبتنی بر پروتکل HTTP برای سرویس دهنده وب ارسال می گردد . در ادامه ، سرویس دهنده درخواست را پردازش (به عنوان نمونه ، انجام برخی محاسبات و یا عملیات مرتبط با بانك های اطلاعاتی) و نتایج تولید شده را در قالب یك صفحه وب با محتویات جدید برای سرویس گیرنده ارسال می نماید .
نحوه عملکرد صفحات وب متاثر از ماهیت stateless بودن پروتکل HTTP است . با توجه به این که تمامی منطق

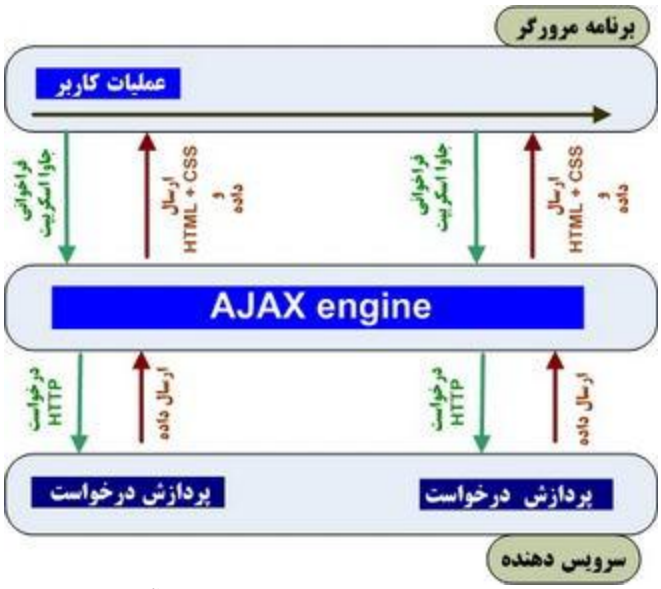
برنامه معمولا" بر روی سرویس دهنده قرار می گیرد ، نقش مرورگرها صرفا" نمایش بخش رابط کاربر و یا اصطلاحا" اینترفیس برنامه است . سرویس دهنده ، چرخه حیات یك صفحه وب را بطور کامل طی می نماید و برای مرورگر تگ های HTML ، کدهای CSS و سایر منابع مورد نیاز را جهت بازخوانی و نمایش مجدد صفحه ارسال می نماید .
ماهیت فرآیند فوق بگونه ای است که در دراز مدت نمی تواند رضایت خاطر کامل کاربران را حداقل در سطح بخش رابط کاربر برنامه تامین نماید . در این مدل کاربران از یك الگوی stop-start-stop تبعیت می نمایند . کاربران در برخی موارد و با توجه به شرایط حاکم بر برنامه بطور موقت و از روی ناچار ارتباط خود را با برنامه از دست داده و می بایست در انتظار بهنگام سازی صفحه وب درخواستی بمانند .
شکل 2 ، نحوه عملکرد برنامه های وب در یك فرآیند همزمان را نشان می دهد .



شکل 2 : نحوه عملکرد برنامه های وب در یك فرآیند همزمان
(عدم تعامل کاربر با برنامه در زمان درخواست های HTTP)

در ASP.NET زمانی که یك صفحه داده را برای خود و یا حتی صفحه ای دیگر ارسال می نماید ، یك postback اتفاق می افتد . در حین این فرآیند ، وضعیت جاری صفحه به همراه کنترل های موجود بر روی آن جهت پردازش برای سرویس دهنده ارسال می گردند . مکانیزم postback با هدف تامین خواسته هائی نظیر نگهداشت وضعیت صفحه و کنترل های سرویس دهنده موجود بر روی آن دنبال می شود . فرآیند فوق گرچه در نهایت می تواند منجر به refresh صفحه وب و نمایش محتویات جدید برای کاربر گردد ولی هزینه انجام آن زیاد خواهد بود چراکه اولا" یك حجم داده می بایست برای سرویس دهنده ارسال گردد و ثانیا" ارتباط منطقی کاربر با برنامه از بین خواهد رفت .

یك برنامه وب مبتنی بر Ajax با مدل و یا رویکردی متفاوت نسبت به آنچه اشاره گردید ، کار می کند . در این مدل ، تعامل مستمر کاربر با برنامه از طریق معرفی یك نماینده که بین سرویس گیرنده و سرویس دهنده قرار می گیرد ، تامین می گردد . این نماینده و یا agent ، با سرویس دهنده بطور غیرهمزمان ارتباط برقرار می نماید (از طرف سرویس گیرنده) تا درخواست HTTP را ایجاد و آن را برای سرویس دهنده ارسال نماید . وظایف نماینده فوق به این نقطه ختم نمی گردد و مسئولیت بهنگام سازی صفحه پس از دریافت داده از سرویس دهنده نیز بر عهده وی می باشد .
در مدل غیر ه مزمان ، Ajax engine توسط جاوا اسکریپت فراخوانده می شود تا داده مورد نظر را درخواست نماید . پس ایجاد درخواست توسط Ajax engine و ارسال آن برای سرویس دهنده و انجام پردازش های ضروری در سمت سرویس دهنده ، نتایج توسط Ajax engine دریافت و بخش رابط کاربر برنامه متناسب با آن بهنگام می گردد .
شکل 3 ، نحوه عملکرد برنامه های وب در یك فرآیند غیرهمزمان را نشان می دهد .



شکل 3 : نحوه عملکرد برنامه های وب در یك فرآیند غیرهمزمان
(ارسال درخواست های HTTP از طریق Ajax engine برای سرویس دهنده)

در هسته Ajax engine ، شی مهم و کلیدی XMLHttpRequest قرار دارد که در ادامه با آن بیشتر آشنا می شویم .

شی XMLHttpRequest

شی XMLHttpRequest به منزله قلب برنامه نویسی Ajax مطرح می گردد چراکه شی فوق باعث می شود جاوا اسکریپت بتواند درخواست هائی را ایجاد تا برای سرویس دهنده ارسال و نتایج ارسالی از سرویس دهنده را نیز پردازش نماید .
شی فوق اولین مرتبه و به صورت یك شی اکتیوایکس در Internet Explorer 5 عرضه گردید و هم اینك از آن در اکثر مرورگرها حمایت می گردد . سایر مرورگرها نظیر Safari ، Opera ، Mozilla و فایرفاکس پتانسیل های XMLHttpRequest را به صورت یك شی ذاتی جاوا اسکریپت ارائه کرده اند (در IE 7.0 شی فوق بطور ذاتی در جاوا اسکریپت تعبیه شده است) .
با توجه به این که تاکنون نسخه های مختلفی از شی فوق در مرورگرها پیاده سازی شده است ، پیاده کنندگان می بایست کد لازم به منظور تشخیص نوع شی فوق را در زمان ایجاد یك نمونه از آن را در برنامه خود پیش بینی نمایند . برای تعیین نسخه در دسترس شی XMLHttpRequest می توان از روشی موسوم به " تشخیص شی " استفاده کرد .

ایجاد یك نمونه از شی XMLHttpRequest با توجه به نوع مرورگر

```
var xmlHttp = null;
 if (window.XMLHttpRequest)     {     //IE7 , Mozilla  ,...
          xmlHttp = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
      try{
         xmlHttp = new ActiveXObject("Microsoft.XMLHTTP"); //IE 5.x, 6
       }
 catch(e) {}
}
```

مثال

برای آشنائی با نحوه عملکرد شی فوق و برنامه نویسی وب غیرهمزمان ، در ادامه به بررسی یك نمونه مثال ساده خواهیم پرداخت . فرض کنید قصد داریم یك درخواست غیرهمزمان به یك منبع موجود بر روی سرویس دهنده (در این

مثال خاص يك صفحه html كه حاوی يك متن ساده است ، صفحه ArticleSummery.htm) را ايجاد نمائيم .
صفحه فوق يك صفحه وب با حداقل اطلاعات و شامل يك متن ايستا است .

صفحه ArticleSummery.htm

```
<html xmlns="http://www.w3.org/1999/xhtml" dir="rtl">
<head>
  <meta content="text/html; charset=utf-8" http-equiv="content-type" />
  <title>برنامه نويسی وب غيرهمزمان تست </title>
</head>
<body style="font-family: Tahoma;">
 <div>اين مقاله با برنامه نويسی وب ه مزمان و غيرهمزمان آشنا شديم در </div>
</body>
</html>
```

مسئوليت ارسال يك درخواست غيرهمزمان به تابع SendRequest سپرده شده است .

ارسال يك درخواست غيرهمزمان

مرحله	کد

1
ارتباط فعال کردن غيرهمزمان
	function **sendRequest**(url) {
	if (xmlHttp) {
	xmlHttp.open("GET", url, true); // true = async

2 نسبت دهی تابع callback	xmlHttp.onreadystatechange = onCallback;
	xmlHttp.setRequestHeader('Content-type','application/x-www-form-urlencoded');

3
درخواست ارسال غيرهمزمان
	xmlHttp.send(null);
	}
	}

توضيحات

- متد sendRequest ، يك پارامتر كه در واقع URL مربوطه به درخواست HTTP است را دريافت می نمايد .
- مرحله اول : يك ارتباط غيرهمزمان ايجاد می گردد (در نظر گرفتن مقدار true به عنوان سومين پارامتر در زمان فعال كردن ارتباط نشان دهنده يك ارتباط غيرهمزمان است).
- مرحله دوم : پس از مقداردهی اوليه ارتباط مورد نظر ، به خصلت onreadystatechange شی XMLHttpRequest يك تابع محلی با نام onCallback نسبت داده می شود . توجه داشته باشيد كه فراخوانی تابع فوق به صورت غيرهمزمان است . تابع Callback مشخص می نمايد كه چه زمانی درخواست تكميل و يا بهنگام شده است .
- مرحله سوم : پس از مشخص كردن نوع محتوا در هدر درخواست ، با استفاده از متد Send شی XMLHttpRequest ، درخواست HTTP برای سرويس دهنده ارسال می گردد .

فراخوانی تابع onCallback

هر مرتبه ای كه وضعيت ready تغيير می يابد ، از تابع callback جهت ايجاد يك درخواست غيرهمزمان استفاده می گردد . در مرحله نهائی ، وضعيت بررسی و بخش رابط كاربر به همراه محتويات برگردانده شده از صفحه ArticleSummery.htm ، بهنگام می گردد.

فراخوانی تابع oncallback

مرحله	کد
1 بررسی تکمیل عملیات	function onCallback() { if (xmlHttp.readyState == 4) {
2 مقدار 200 نشان دهنده انجام موفقیت آمیز عملیات است	if (xmlHttp.status == 200){

3
نتایج نمایش
	var r = document.getElementById('results');
	r.innerHTML = xmlHttp.responseText;
	}

	else {
	alert('Error: ' + xmlHttp.status);
	}

توضیحات

- وضعیت درخواست از طریق خصلت readyState برگردانده می شود .
- مرحله اول : در صورتی که مقدار خصلت readyState شی XMLHttpRequest برابر با مقدار 4 باشد ، درخواست به اتمام رسیده است .
- مرحله دوم : در ادامه ، پاسخ برگردانده شده از سرویس دهنده بررسی می شود تا این اطمینان حاصل گردد که همه چیز با موفقیت انجام شده است .مقدار کد وضعیت 200 مربوط به پروتکل HTTP ، نشان دهنده این موضوع است که درخواست با موفقیت انجام شده است .
- مرحله سوم : در نهایت ، خصلت innerHTML مربوط به عنصر span متاثر از محتویات برگردانده شده ، بهنگام می گردد .

کد زیر ، محتویات صفحه Ajax1.aspx را بطور کامل نشان می دهد .

صفحه Ajax1.aspx

```
<%@ Page Language="VB" Culture="fa-IR" %>
<script runat="server">
</script>
<html xmlns="http://www.w3.org/1999/xhtml" dir="rtl" >
    <head id="Head1" runat="server">
        <title>استفاده از شی نحوه XMLHttpRequest</title>
    </head>
 <body style="font-family: Tahoma">
  <form id="form1" runat="server">
    <div>
      <span id="results">صفحه بارگذاری ...</span>
    </div>
  </form>
<script type="text/javascript">
   var xmlHttp = null;
   window.onload = function() {
   loadXmlHttp();
   sendRequest("ArticleSummery.htm");
 }
```

```
 function loadXmlHttp() {
   if (window.XMLHttpRequest) { // IE7, Mozilla, Safari, Opera, etc.
   xmlHttp = new XMLHttpRequest();
   } else if (window.ActiveXObject) {
   try{
    xmlHttp = new ActiveXObject("Microsoft.XMLHTTP"); // IE 5.x and 6
  }
catch (e){}
  }
 }
 function sendRequest(url) {
 if (xmlHttp) {
    xmlHttp.open("GET", url, true); // true = async
    xmlHttp.onreadystatechange = onCallback;
    xmlHttp.setRequestHeader('Content-type', 'application/x-www-form-
urlencoded');
  // Send request without any additional parameters
  xmlHttp.send(null);
  }
}
function onCallback() {
if (xmlHttp.readyState == 4) {
if (xmlHttp.status == 200){
var r = document.getElementById('results');
r.innerHTML = xmlHttp.responseText;
 }
else {   // HTTP error
  alert('Error: ' + xmlHttp.status);
   }
 }
}
 </script>
 </body>
</html>
```

شکل 4 خروجی مثال فوق را نشان می دهد .

شکل 4 : ایجاد يك درخواست Http غیرهمزمان توسط شی XMLHttpRequest

در این مثال با نحوه ایجاد يك درخواست HTTP غیرهمزمان توسط شی XMLHttpRequest به صفحه دیگر موجود بر روی سرویس دهنده آشنا شدیم . پس از اتمام درخواست ، کاربران صفحه نهائی را که محتویات عناصر رابط کاربر موجود در آن (يك span) به صورت پویا بهنگام شده اند ، مشاهده خواهند کرد .

خلاصه

در این مقاله با برنامه نویسی وب همزمان و غیرهمزمان و نحوه عملکرد شی XMLHttpRequest آشنا شدیم . هدف از بیان موارد فوق ، صرفا" آشنائی با الگوی برنامه نویسی وب مبتنی بر Ajax بود . تمامی داستان به این نقطه ختم نمی شود و در مقالات آتی به سایر پتانسیل های Ajax به منظور پیاده سازی برنامه های وب اشاره خواهیم کرد .