

آموزش برنامه‌نویسی

SignalR

با استفاده از ASP.NET

مهندس سید منصور عمرانی

انتشارات پندار پارس

سرشناسه	: آگیلار، خوسه ام . .M Aguilar, José
عنوان و نام پدیدآور	: آموزش برنامه‌نویسی Signal R با استفاده از ASP.NET
مشخصات نشر	: تهران : پندار پارس ، ۱۳۹۴ .
مشخصات ظاهری	: ۴۰۶ص. : مصور، جدول.
شابک	: 978-600-6529-78-3 : ۳۳۰۰۰۰ ریال
وضعیت فهرست نویسی	: فیپای مختصر
یادداشت	: این مدرک در آدرس http://opac.nlai.ir قابل دسترسی است.
یادداشت	: عنوان اصلی: Microsoft ASP.NET Signal R Programming in
شناسه افزوده	: عمرانی، سیدمنصور، ۱۳۵۶ - مترجم
شماره کتابشناسی ملی	: ۳۸۰۱۱۳۴

انتشارات پندار پارس



دفتر فروش: انقلاب، ابتدای کارگر جنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ www.pendarepars.com
 تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸ همراه: ۰۹۱۲۲۴۵۲۳۴۸
info@pendarepars.com



نام کتاب	: آموزش برنامه‌نویسی SignalR با استفاده از ASP.NET
ناشر	: انتشارات پندار پارس
ترجمه و تألیف	: سید منصور عمرانی
چاپ نخست	: اردیبهشت ۹۴
شمارگان	: ۵۰۰ نسخه
چاپ، صحافی	: روز
قیمت	: ۳۳۰۰۰ تومان
شابک	: ۳-۷۸-۶۵۲۹-۶۰۰-۹۷۸

*هرگونه کپی برداری، تکثیر و چاپ کاغذی یا الکترونیکی از این کتاب بدون اجازه ناشر تخلف بوده و پیگرد قانونی دارد *

فهرست

فصل ۱. اینترنت، غیر همزمانی، چند کاربری.....	۱
برنامه‌های اینترنتی چند کاربردی غیر همزمان.....	۱
افزایش تقاضا و سطح انتظار برای سریع‌تر شدن وب.....	۲
جدیدترین تحول دنیای توسعه‌ی وب: ارتباط real-time.....	۳
تلاش‌های صورت گرفته.....	۳
چند نمونه در دنیای واقعی.....	۴
مایکروسافت و برنامه‌های real-time.....	۴
مروری بر محتوای کتاب.....	۵
فصل ۲. HTTP و مدل Push.....	۷
مدل pull.....	۷
محدودیت‌های پروتکل HTTP.....	۸
بهبود وب با Ajax.....	۸
محدودیت Ajax.....	۹
مدل push.....	۹
تکنیک سرکشی یا polling.....	۱۰
نقاط قوت و ضعف تکنیک polling.....	۱۰
بهبود تکنیک polling.....	۱۱
ترکیب تکنیک‌ها.....	۱۲
المان‌های گنجانده‌نی یا pluggable.....	۱۲
تکنیک‌های فراهم کردن real-time web.....	۱۳
روش‌های سنتی.....	۱۳
تکنیک سرکشی طولانی یا long polling.....	۱۳
نقاط قوت و ضعف تکنیک long polling.....	۱۴
تکنیک قاب ابدی یا forever frame.....	۱۵
نقاط قوت و ضعف تکنیک forever frame.....	۱۶
پیامد روش‌های سنتی.....	۱۷
روش‌های پیشرفته.....	۱۷
Server-Sent Events.....	۱۷
نحوه‌ی کار.....	۱۸
تحلیل و بررسی.....	۱۸
WebSockets.....	۱۹
نحوه‌ی استفاده.....	۲۱
مروری بر تکنیک‌های ارتباط real-time در وب.....	۲۲
دنیا علاوه بر هل دادن (push) به چیزهای دیگری هم نیاز دارد.....	۲۲
جمع‌بندی.....	۲۳
راه‌حل فراهم کردن مدل push در دنیای فعلی.....	۲۳
بالاخره چه باید کرد؟.....	۲۳
SignalR به یاری می‌شتابد.....	۲۴

۲۵	فصل ۳. معرفی SignalR
۲۵	مروری بر SignalR
۲۵	تاریخچه
۲۵	نسخه‌های مختلف SignalR
۲۶	از کجا می‌توانیم SignalR را به دست بیاوریم؟
۲۶	SignalR واقعا چیست؟
۲۷	نمونه‌ی عملی استفاده از SignalR
۲۷	آیا سرویس SignalR را تنها در وب می‌توان استفاده کرد؟
۲۷	آیا سرویس SignalR را تنها با ASP.NET می‌توان ایجاد کرد؟
۲۸	مدل برنامه‌نویسی و اجزای SignalR
۲۸	SignalR چه چیزی ارائه می‌کند؟
۲۹	استقلال از میزبان
۲۹	مدیریت اتصال
۲۹	حامل یا transport
۳۰	مهمترین ویژگی‌ها و مزایای SignalR
۳۰	چه موقع از SignalR استفاده نکنیم؟
۳۱	سکوهاى مورد پشتیبانی
۳۱	سکوهاى سرورى
۳۲	وب‌سرور مورد نیاز
۳۲	سکوهاى کلانیتی
۳۳	نیازمندی‌های حامل‌های مختلف SignalR
۳۳	لايه‌ها يا سطوح انتزاعی SignalR
۳۴	اتصال دائمی یا persistent connection
۳۴	هاب (hub)
۳۴	OWIN
۳۵	چرا OWIN ابداع شد؟
۳۵	OWIN چیست؟
۳۶	مزایای جداسازی برنامه‌ی وب و سرور وب از یکدیگر
۳۷	اجزای یک برنامه‌ی مبتنی بر OWIN
۳۹	OWIN چگونه کار می‌کند؟
۴۱	Katana
۴۲	کلاس راه‌نماز و پیکربندی برنامه
۴۲	یک برنامه‌ی ساده‌ی OWIN با استفاده از Katana
۴۴	مروری بر بسته‌های NuGet پروژه‌ی Katana
۴۵	نصب SignalR
۴۵	بسته‌های SignalR
۴۶	نصب SignalR
۴۷	یک برنامه‌ی ساده‌ی SignalR
۶۱	فصل ۴. Persistent Connection (قسمت اول)
۶۲	ویژگی‌های persistent connection
۶۳	مراحل بر پا شدن اتصال

۶۴	نحوه‌ی ایجاد و استفاده از persistent connection
۶۴	نصب SignalR
۶۴	پیاده‌سازی بخش سمت سرور
۶۴	نحوه‌ی تعریف persistent connection
۶۴	رویدادهای persistent connection
۶۶	دریافت اطلاعات از کلاینت‌ها
۶۷	ارسال اطلاعات به کلاینت‌ها
۶۷	راه اول: متد Broadcast()
۶۸	راه دوم: متد توسعه‌ی Send()
۶۹	راه سوم: متد Send()
۷۰	دیگر متدهای کلاس PersistentConnection
۷۱	تعریف آدرس URL برای persistent connection
۷۳	پیاده‌سازی یک کلاینت وبی
۷۳	ضمیمه کردن کتابخانه‌های لازم به صفحه
۷۴	نحوه‌ی متصل شدن به یک persistent connection
۷۵	مشخص کردن تنظیمات اتصال
۷۶	پشتیبانی از مرورگرهای قدیمی
۷۶	ارسال اطلاعات
۷۹	دریافت اطلاعات
۸۰	ارسال اطلاعات اضافی به سرور
۸۱	دیگر رویدادهای قابل استفاده در سمت کلاینت
۸۳	مرور مثال MoveShape فصل گذشته
۸۵	فصل ۵. Persistent Connection (قسمت دوم)
۸۵	کلاس persistent connection و ملاحظات اجرای همروند
۸۶	اجرای غیر همزمان و جلوگیری از مصرف منابع سرور
۸۹	روند ارتباط کلاینت و سرور هنگام ایجاد persistent connection
۸۹	Negotiation
۹۱	Connect
۹۱	Start
۹۱	دیگر درخواست‌های کلاینت در ارتباط با سرور
۹۱	Ping
۹۲	Send
۹۲	Poll
۹۲	Abort
۹۳	مشاهده‌ی روند مذاکره
۹۴	ردگیری رویدادهای شی connection در کلاینت‌های جاوااسکریپتی
۹۵	محدودیت تعداد اتصال به سرور
۹۵	رویدادهای اتصال‌های persistent connection
۹۶	رویدادهای سمت سرور
۹۶	رویدادهای سمت کلاینت
۹۷	مرور خصوصیت‌های شی connection در کلاینت‌های جاوااسکریپتی
۹۸	گروه‌بندی کلاینت‌ها

۱۰۰	ایجاد گروه تک-کاربره
۱۰۱	بستن اتصال persistent connection
۱۰۲	آگاه شدن از علت قطع شدن کلاینت
۱۰۲	کلاس راه‌انداز OWIN
۱۰۴	تنظیمات HostContext
۱۰۵	اتصال به سرور SignalR از دیگر دامنه‌ها با استفاده از CORS و JSONP
۱۰۵	فعال‌سازی سمت سرور
۱۰۶	تنظیمات کلاینت
۱۰۷	درک بهتر رویدادهای چرخه‌ی حیات اتصال در SignalR
۱۰۷	واژگان پایه
۱۰۹	سناریوهای قطع شدن اتصال حامل
۱۱۱	نمودار تغییر وضعیت اتصال
۱۱۳	قطع شدن ارتباط کلاینت به سرور
۱۱۳	قطع شدن ارتباط سرور به کلاینت
۱۱۴	نحوه‌ی تغییر دادن تنظیمات timeout و keepalive
۱۱۷	فصل ۶. هاب (قسمت اول)
۱۱۷	هاب چیست؟
۱۱۹	هاب چگونه جادوی خود را انجام می‌دهد؟
۱۱۹	پیاده‌سازی سمت سرور
۱۱۹	ایجاد هاب
۱۱۹	تعریف متد برای هاب
۱۲۰	نمونه‌سازی از کلاس هاب
۱۲۰	پیکربندی برنامه برای استفاده از هاب
۱۲۱	پیاده‌سازی سمت کلاینت
۱۲۲	استفاده از پراکسی اختصاصی
۱۲۲	الف. استفاده از پراکسی‌های خودکار
۱۲۴	ب. تولید دستی کلاس‌های پراکسی
۱۲۸	به دست آوردن ارجاعی به شی پراکسی
۱۲۸	تعریف نام شخصی برای هاب و متدهای آن
۱۲۹	بر پا کردن اتصال
۱۳۰	مشخص کردن حامل اتصال
۱۳۰	نحوه‌ی فراخوانی متدهای هاب توسط کلاینت جاوااسکریپتی
۱۳۲	برگرداندن مقدار
۱۳۳	دریافت خطاهای به وقوع پیوسته در سمت سرور
۱۳۴	تعریف نسخه‌های سربارگذاری شده برای متدهای هاب
۱۳۵	استفاده از پارامترهای پیچیده در متدها
۱۳۶	متدهای زمان‌بر
۱۳۷	فراهم کردن قابلیت گزارش لحظه‌ای
۱۳۸	فراخوانی متدهای کلاینت توسط سرور
۱۳۸	الف. فراخوانی صریح
۱۴۰	ب. استفاده از Invoke()
۱۴۰	دیگر خصوصیت‌های شی Clients
۱۴۱	متدهای شی Clients

۱۴۲	غیر همزمان بودن فراخوانی متدهای کلاینت در سمت سرور
۱۴۴	یادآوری دو نکته
۱۴۴	ارسال پیام به یک کاربر به جای یک کلاینت
۱۴۴	تغییر دادن سیستم واکنشی هویت کاربر
۱۴۶	پیاده‌سازی کلاینت بدون پراکسی اختصاصی
۱۴۶	الحاق کتابخانه‌های لازم به صفحه
۱۴۷	ایجاد اتصال به سرور
۱۴۷	مشخص کردن حامل اتصال
۱۴۷	به دست آوردن پراکسی عمومی
۱۴۸	فراخوانی متدهای سرور
۱۴۸	دریافت درخواست‌های سرور برای فراخوانی متدهای سمت کلاینت
۱۴۹	یک مثال کامل: تخته‌ی نقاشی اشتراکی
۱۴۹	ایجاد پروژه و تنظیم آن
۱۵۰	پیاده‌سازی سرور
۱۵۰	هاب (DrawingBoard.cs)
۱۵۰	کُد راه‌انداز (Startup.cs)
۱۵۱	پیاده‌سازی کلاینت
۱۵۱	کُد HTML صفحه‌ی تخته نقاشی (drawingboard.html)
۱۵۱	اسکرپت‌ها (Scripts/DrawingBoard.js)
۱۵۵	فصل ۷. هاب (قسمت دوم)
۱۵۵	تغییر دادن آدرس پیش فرض استفاده از هاب‌ها
۱۵۵	متصل شدن به هاب‌ها از دیگر دامنه‌ها
۱۵۶	فعال‌سازی در سمت سرور
۱۵۶	CORS
۱۵۷	JSONP
۱۵۷	تنظیمات کلاینت
۱۵۸	ارسال اطلاعات اضافی
۱۵۹	دسترسی به اطلاعات زمینه‌ی درخواست
۱۶۰	آگاه شدن از قطع و وصل شدن کلاینت‌ها
۱۶۲	رویدادهای اتصال هاب به سرور
۱۶۳	قطع کردن اتصال کلاینت
۱۶۳	راز فراخوانی متدهای جاوااسکریپتی توسط سرور
۱۶۷	ایجاد هاب با استفاده از <T>Hub
۱۶۸	نگاه دقیق‌تری به مکالمه‌ی سرور و کلاینت
۱۶۸	ساختار بسته‌های درخواست فراخوانی متد
۱۶۸	فراخوانی اشتباهی متدهای سمت سرور توسط کلاینت
۱۶۹	کلاینت جاوااسکریپتی و پراکسی اختصاصی خودکار
۱۷۰	کلاینت جاوااسکریپتی و پراکسی اختصاصی دستی
۱۷۰	کلاینت جاوااسکریپتی و پراکسی عمومی
۱۷۱	فراخوانی اشتباهی متدهای سمت کلاینت توسط سرور
۱۷۲	پراکسی چگونه کار می‌کند

۱۷۷.....	گروه‌بندی کلاینت‌ها.....
۱۷۸.....	به دست آوردن کلاینت‌ها بر حسب گروه.....
۱۸۰.....	حفظ وضعیت در سرور و امنیت نخي.....
۱۸۰.....	کلاس Interlocked.....
۱۸۱.....	دستور lock.....
۱۸۲.....	کلکسیون‌های همروند.....
۱۸۳.....	کلکسیون‌های همروند کجا thread-safety را فراهم می‌کنند؟.....
۱۸۴.....	مشکل فیلدهای استاتیک.....
۱۸۶.....	چالش SignalR در محیط‌های توزیع شده.....
۱۸۷.....	لاگ‌گیری.....
۱۸۸.....	مدیریت وضعیت.....
۱۹۱.....	سفارشی کردن خط لوله‌ی هاب با استفاده از HubPipelineMode.....
۱۹۳.....	فصل ۸. دسترسی به اتصالات‌های دائمی و هاب از نخ‌های دیگر.....
۱۹۳.....	دسترسی از نخ‌های دیگر.....
۱۹۴.....	دسترسی به هاب‌ها و اتصالات‌های دائمی از بیرون.....
۱۹۴.....	دسترسی به اتصالات‌های دائمی.....
۱۹۵.....	دسترسی به هاب.....
۱۹۶.....	به دست آوردن هاب نوع‌دار.....
۱۹۷.....	یک مثال کامل: نظارت بر متصل شدن کلاینت‌ها به سرور.....
۱۹۸.....	ایجاد پروژه و تنظیم آن.....
۱۹۸.....	پیاده‌سازی برنامه‌ی وب.....
۱۹۸.....	کُد HTML صفحه‌ی اصلی برنامه (default.aspx).....
۱۹۸.....	کُد C# صفحه‌ی اصلی برنامه (default.aspx.cs).....
۱۹۹.....	سیستم ردگیری درخواست‌ها (سمت سرور).....
۱۹۹.....	اتصال دائمی (ConnectionSpy.cs).....
۱۹۹.....	کُد راه‌انداز برنامه (Startup.cs).....
۱۹۹.....	کُد کلاس سراسری برنامه (global.asax.cs).....
۲۰۰.....	تنظیمات اختیاری (web.config).....
۲۰۱.....	سیستم ردگیری درخواست‌ها (سمت کلاینت).....
۲۰۱.....	صفحه‌ی ردگیری spy.html.....
۲۰۱.....	یک مثال کامل: میله‌ی پیشرفت عملیات یا Progress bar.....
۲۰۲.....	ایجاد و تنظیم پروژه.....
۲۰۲.....	پیاده‌سازی کلاینت.....
۲۰۲.....	فایل صفحه‌ی HTML (progress.html).....
۲۰۲.....	شیوه‌های CSS (Styles/ProgressBar.css).....
۲۰۳.....	کُد‌های جاوااسکریپت (Scripts/ProgressBar.js).....
۲۰۳.....	پیاده‌سازی سرور.....
۲۰۳.....	هاب.....
۲۰۴.....	صفحه‌ی اجرای عملیات سنگین (HarsProcess.aspx).....
۲۰۴.....	کُد راه‌انداز برنامه (startup.cs).....
۲۰۵.....	فصل ۹. برنامه‌های SinglaR در سکوها‌ی دیگر.....
۲۰۵.....	میزبانی سرویس‌های SignalR در برنامه‌های غیر وب.....

۲۰۶میزبانی در یک برنامه‌ی کنسول.....
۲۰۹میزبانی در یک سرویس ویندوز.....
۲۱۴میزبانی در سکوها‌ی غیر ویندوز.....
۲۱۶ استفاده از سرویس‌های SignalR در کلاینت‌های غیر وب
۲۱۷ استفاده از اتصال‌های دائمی.....
۲۱۷ ایجاد شی اتصال.....
۲۱۸ خصوصیت‌های شی اتصال.....
۲۱۹ متصل شدن به سرویس SignalR.....
۲۲۰ ارسال اطلاعات.....
۲۲۱ دریافت اطلاعات.....
۲۲۲ استفاده از هاب
۲۲۲ ایجاد شی هاب
۲۲۲ تنظیمات اتصال.....
۲۲۵ به دست آوردن پراکسی.....
۲۲۵ متصل شدن به سرور.....
۲۲۶ فراخوانی متدهای سمت سرور توسط کلاینت.....
۲۲۷ فراخوانی متدهای سمت کلاینت توسط سرور.....
۲۲۹ حذف متد سمت کلاینت
۲۲۹ حفظ وضعیت.....
۲۳۰ رویدادهای اتصال هاب به سرور.....
۲۳۰ لاگ‌گیری.....
۲۳۲ چند نکته در مورد نوشتن کلاینت‌های غیر وب
۲۳۲ دسترسی به واسط کاربر در کلاینت‌های رومیزی.....
۲۳۴ کلاینت Silverlight و Windows Store.....
۲۳۴ کلاینت Windows Phone 8.....
۲۳۶ چند مثال
۲۳۶ یک برنامه‌ی خط فرمان
۲۳۸ یک برنامه‌ی Windows 8/Windows 8.1 با استفاده از C#/XAML.....
۲۳۸ کد فایل اصلی برنامه (MainPage.xaml).....
۲۳۹ فایل MainPage.xaml.cs.....
۲۴۳ استفاده از سرویس‌های SignalR از دیگر سکوها
۲۴۳ برنامه‌های بومی C++.....
۲۴۵ فصل ۱۰. ایجاد برنامه‌های SignalR تحت لینوکس و Mac OS X با استفاده از Mono.....
۲۴۵ Mono چیست؟.....
۲۴۶ کامپایل سورس SignalR در لینوکس openSUSE 13.1
۲۴۶ ایجاد یک کلاینت SignalR.....
۲۴۸ نصب محیط توسعه‌ی MonoDevelop.....
۲۴۸ تنظیم Apache به عنوان میزبان.....
۲۴۹ ایجاد یک سرور SignalR.....
۲۵۲ اجرای برنامه‌ی SignalR در OS X با استفاده از Mono.....
۲۵۲ استفاده از Xamarin Add-in در Visual Studio برای ایجاد کلاینت‌های iOS و Android برای SignalR.....
۲۵۲ تنظیم Visual Studio در Xamarin Add-in.....
۲۵۳ ایجاد برنامه‌ی Android.....

۲۵۸ نصب Xamarin Studio روی سیستم عامل اپل
۲۵۸ عضویت در Apple Developer Program
۲۵۹ تنظیم کردن رسانه‌ی اپل
۲۵۹ تنظیم و راه‌اندازی Xamarin.iOS Build Host
۲۶۲ ایجاد یک کلاینت SignalR برای iOS
۲۶۹ فصل ۱۱. اشکال‌زدایی برنامه‌های SignalR
۲۶۹ ردگیری اجرای برنامه در سمت سرور
۲۷۰ ردگیری اجرای برنامه در سمت کلاینت
۲۷۲ استفاده از Fiddler
۲۷۳ اداره کردن خطاها در هاب‌ها
۲۷۷ لاگ‌گیری
۲۷۷ ردگیری
۲۸۳ فصل ۱۲. انتشار و مقیاس پذیر کردن سرویس‌های SignalR
۲۸۴ سردرد بی‌انتهای
۲۸۷ مقیاس کردن برنامه‌های SignalR
۲۸۸ backplane
۲۹۰ محدودیت backplane
۲۹۱ backplane کجا مفید است؟
۲۹۱ راه‌حل نهایی چیست؟
۲۹۱ پارتیشن‌بندی کاربران به طور استاتیک
۲۹۲ توازن بار هوشمند و پارتیشن‌بندی داینامیک
۲۹۳ نحوه‌ی استفاده از backplane
۲۹۳ Windows Azure Service Bus
۲۹۴ پیکرندگی سرویس
۲۹۶ فعال کردن backplane
۲۹۸ SQL Server
۲۹۸ پیکرندگی دیتابیس
۲۹۹ فعال کردن backplane
۳۰۰ Redis
۳۰۱ نصب Redis
۳۰۲ فعال کردن backplane
۳۰۳ Backplane های شخصی
۳۰۷ فصل ۱۳. بهبود دادن کارایی سرویس‌های SignalR
۳۰۹ پیکرندگی سرور
۳۰۹ حداکثر تعداد درخواست همزمان به ازای هر برنامه (Maximum Concurrent Request Per Application)
۳۰۹ حداکثر تعداد درخواست همزمان به ازای هر CPU (Maximum Concurrent Request Per CPU)
۳۱۰ سرحد صف درخواست‌ها (Request Queue Limit)
۳۱۰ اندازه‌ی پیش فرض بافر پیام‌ها (Default Message Buffer Size)
۳۱۱ ردگیری کارایی
۳۱۳ Connection metrics
۳۱۳ Message metrics

۳۱۴ Errors metrics
۳۱۴ Message bus metrics
۳۱۵ Scale-out metrics
۳۱۶ تنظیم تعداد جدول (جریان) و طول صف برای SQL Server
۳۱۶ تنظیم تعداد topic (جریان) و طول صف برای Service Bus
۳۱۶ دیگر شمارنده‌های کارایی مفید
۳۱۹ فصل ۱۴. مباحث پیشرفته
۳۱۹ امنیت برنامه‌های وب
۳۱۹ گام‌های لازم برای نوشتن یک برنامه‌ی ایمن
۳۲۰ امنیت در برنامه‌های SignalR
۳۲۱ مدل امنیتی ASP.NET در گذشته
۳۲۲ تصدیق هویت فرم و ویندوز
۳۲۳ مدل امنیت در OWIN
۳۲۴ نحوه‌ی فعال کردن تصدیق هویت مبتنی بر کوکی
۳۲۵ نحوه‌ی تولید کوکی تصدیق هویت
۳۲۶ نحوه‌ی فعال کردن تصدیق هویت ویندوز
۳۲۶ امنیت در SignalR
۳۲۶ امنیت شناسه‌ی اتصال
۳۲۷ امنیت عضویت گروه‌ها
۳۲۸ حمله‌ی CSRF
۳۲۹ ایمن‌سازی هاب‌ها و اتصال‌های دائمی
۳۲۹ ایمن‌سازی اتصال‌های دائمی
۳۳۰ ایمن‌سازی هاب‌ها
۳۳۲ سفارشی کردن سیستم تصدیق هویت
۳۳۳ مجتمع کردن برنامه با IIS Integrated Pipeline
۳۳۳ استفاده از شیوه‌های مختلف تصدیق هویت در کلاینت‌های غیر وب
۳۳۳ Forms Authentication
۳۳۵ Windows Authentication
۳۳۶ Cookie Authentication
۳۳۶ نوشتن یک مدل تصدیق هویت شخصی
۳۳۶ تنظیم کردن برنامه
۳۳۷ نوشتن میان‌افزار تصدیق هویت
۳۴۰ تصدیق هویت بر اساس header
۳۴۱ دیگر توصیه‌های امنیتی
۳۴۲ SignalR، فریم‌ورکی توسعه‌پذیر
۳۴۲ SignalR چطور قابل توسعه است؟
۳۴۴ رافع وابستگی یا Dependency Resolver
۳۴۵ تغییر دادن اجزای SignalR با استفاده از رافع وابستگی
۳۴۷ تغییر دادن شی رافع وابستگی پیش فرض SignalR
۳۵۰ رافع وابستگی چگونه کار می‌کند؟
۳۵۱ ایجاد شی تک نمونه یا Singleton برای پاسخ‌دادن یک نوع
۳۵۲ مثال
۳۵۲ استفاده از تکنیک تزریق وابستگی در SignalR

۳۵۵	تزریق به طور دستی.....
۳۵۷	آزاد کردن اشیا وابستگی
۳۵۷	ظرف‌های IoC.....
۳۵۹	نحوه‌ی استفاده از ظرف IoC در SignalR.....
۳۶۰	استفاده از Unity در SignalR.....
۳۶۲	استفاده از Ninject در SignalR.....
۳۶۳	آزمون واحد در SignalR.....
۳۶۳	یک مثال عملی.....
۳۶۶	آزمایش اجزای دارای وابستگی.....
۳۶۸	استفاده از اشیا ساختگی برای وابستگی‌ها.....
۳۷۰	اجرای آزمون واحد برای هاب‌ها.....
۳۷۵	اجرای آزمون واحد برای اتصال‌های دائمی.....
۳۷۸	شنود مبادله‌ی پیام در هاب‌ها.....
۳۸۲	استفاده از SignalR در فریم‌ورک‌های دیگر.....
۳۸۲Web API
۳۸۵ASP.NET MVC
۳۸۶Knockout
۳۸۹ AngularJS
۳۹۲	به روز شدن دیر هنگام واسط کاربر.....
۳۹۲	سیستم تزریق وابستگی خودکار.....
۳۹۴	مطالب تکمیلی.....
۳۹۴	شی پیکربندی ConnectionConfiguration و HubConfiguration.....
۳۹۴	شی ConnectionConfiguration.....
۳۹۴	شی HubConfiguration.....
۳۹۴	مقایسه هاب و persistent connection.....

مقدمه مترجم

دنیای وب در سال‌های اخیر به دلیل رشد شبکه‌های اجتماعی، موبایل‌های هوشمند، تبلت‌ها و تعامل بیشتر کاربران با یکدیگر شتاب زیادی گرفته است. در این میان نیاز به سیستم‌هایی که به طور آنی یا **real-time** بتوانند با کاربران در ارتباط باشند بیشتر احساس شده است. اما سبک سنتی وب و پروتوکل **HTTP** تنها ارتباط یک طرفه‌ی مرورگر به سرور را فراهم کرده و تا این اواخر راهی وجود نداشت که سرور بتواند با مرورگر ارتباط برقرار کرده و چیزی برایش بفرستد. از این رو برای نوشتن برنامه‌های **real-time** از تکنیک‌های ابتکاری مانند سرکشی دائمی یا **polling** استفاده می‌شد.

این روش‌ها نیز مشکلات خودش را داشت و برخی سناریوها مانند بازی‌های آن‌لاین در آنها قابل اجرا نبود. دنیای وب به چیز جدیدی نیاز داشت. این نیاز در **HTML5** با ابداع پروتوکل‌های **WebSockets** و **Server-Sent Events** پوشش داده شد. مشکل این بود که دنیا هنوز آماده‌ی پذیرش نبود و همه‌ی مرورگرها از این پروتوکل‌ها به یک شکل پشتیبانی نمی‌کردند. لذا برنامه‌نویسان برای مهیا کردن بهترین مکانیزم ارتباط **real-time** با کدهایی درگیر می‌شدند که برنامه‌های آنها را کثیف و شلوغ می‌کرد. خوشبختانه میکروسافت با ارائه‌ی فریم‌ورک **SignalR** بهترین ابزار مورد نیاز یک برنامه‌نویس را برای نوشتن برنامه‌های **real-time web** فراهم کرد. **SignalR** فریم‌ورکی است که بدون درگیری با جزئیات سطح پایین پروتوکل و تکنیک‌های فراهم کردن ارتباط آنی کلاینت و سرور مدل انتزاعی سطح بالایی فراهم می‌کند که با آن می‌توانید به ساده‌ترین شکل ممکن برنامه‌های وبی **real-time** بنویسید.

کتابی که در دست دارید این فریم‌ورک را به طور کامل توضیح داده و نحوه‌ی نوشتن برنامه‌های **real-time** در وب را به شما آموزش می‌دهد. منبع اصلی ترجمه، کتاب **SignalR Programming in Microsoft ASP.NET** از انتشارات میکروسافت در سال ۲۰۱۴ بوده است. اما به دلیل تغییرات سریع سکوی **SignalR** که سال‌های اولیه‌ی انتشار هر فریم‌ورک نوپایی را در بر می‌گیرد، مترجم ترجیح داد محتوای برخی فصل‌ها را تغییر بدهد. این تغییرات به طور عمده به فصل‌های ۴ تا ۷ در خصوص اتصال دائمی و هاب (دو لایه‌ی انتزاعی اصلی **SignalR**) اعمال شد. برای این کار نیز مترجم به طور گسترده از مستندات **SignalR** در سایت www.asp.net، مقالات مختلف و همچنین بررسی سورس **SignalR** و **Katana** در **github** استفاده نموده است. همچنین مترجم برای نوشتن این کتاب از کتاب **Pro SignalR 2.1** از انتشارات **Apress** در سال ۲۰۱۴ نیز بهره گرفته که در این زمینه به طور ویژه می‌توان به فصل ۱۰ (ایجاد برنامه‌های **SignalR** تحت لینوکس و **Mac OS X** با استفاده از **Mono**) اشاره کرد. در پایان، مترجم امیدوار است خواننده از مطالعه‌ی کتاب بهره ببرد. در صورت داشتن هرگونه نظر، پیشنهاد، انتقاد یا پرسشی می‌توانید از طریق آدرس mansoor.omrani@gmail.com با مترجم در ارتباط باشید.

با آرزوی موفقیت

سید منصور عمرانی

بهار ۱۳۹۴

فصل ۱. اینترنت، غیر همزمانی^۱، چند کاربری^۲

برنامه‌های اینترنتی چند کاربری غیر همزمان

امروز دنیا شاهد وبی است که افراد هرچه بیشتر در آن مشارکت دارند. دیگر زمان سایت‌هایی سپری شده که با وجود انبوهی از گرافیک و افکت‌های چشم‌نواز به دلیل تک‌کاربری خشک و بی‌روح بودند. به جای آنها سایت‌هایی رواج پیدا کرده که در آنها بازدیدکنندگان به طور زنده از تعامل یکدیگر آگاه بوده و با هم مرتبط هستند. از نظر فنی برنامه‌ای که هم تحت وب باشد، هم غیر همزمان کار کند و هم چند کاربره بوده و به طور همزمان چندین کاربر بتوانند با آن کار کنند همیشه تحسین‌برانگیز است. بدون تردید بسیاری از ما با دیدن سایت‌هایی مانند فیسبوک، تویتر، جیمیل، اسناد گوگل، برنامه‌های وب آفیس یا بسیاری از دیگر سایت‌ها که به روزرسانی صفحه در آنها به طور آنی یا real-time انجام می‌شود بدون آن که نیازی به بازتازه شدن صفحه باشد شگفت‌زده شده‌ایم.

برای نمونه هنگامی که در برنامه‌ی تحت وب آفیس در حال ویرایش سندی هستیم و فردی از جای دیگری می‌خواهد با آن کار کند همان لحظه متوجه حضورش می‌شویم و می‌توانیم تغییراتی را که او اعمال کرده مشاهده کنیم. حتی در سناریوی عمومی‌تری مانند یک برنامه‌ی چت، پیامی که مخاطب ارسال می‌کند بی‌درنگ در سمت ما ظاهر می‌شود. چنین سیستم‌هایی همگی از یک روش استفاده می‌کنند: انتقال داده به صورت غیر همزمان و real-time^۳.

■ **توضیح اصطلاح پردازش آنی یا بی‌درنگ (real time processing یا real time computing) از سال‌ها پیش هم وجود داشته است اما real time web تقریباً اصطلاح جدیدی است و چند سالی است که مطرح شده است. real-time web به معنی وبی است که در آن ارتباط سرور و کلاینت به شکل آنی انجام شود، گویی هر دو به طور دائم به یکدیگر وصل هستند.**

به عنوان برنامه‌نویس وب ما از پیش با سبک سنتی برنامه‌نویسی وب آشنا هستیم. کلاینت درخواستی برای سرور فرستاده و اطلاعاتی را درخواست می‌کند. سرور هم به درخواست‌های کلاینت‌ها پاسخ داده و چیزی بر می‌گرداند. اما تاکنون وارون این ارتباط را مشاهده نکرده‌ایم. یعنی ندیده‌ایم سرور بدون درخواست کلاینت چیزی برای او بفرستد. شاید به همین دلیل است که از دیدن برنامه‌هایی که وضعیت کلاینت در آنها بدون درخواست کلاینت به روز می‌شود به هیجان آمده‌ایم.

^۱ asynchrony

^۲ multiuser

^۳ Asynchronous data transfer between server and client in real time

افزایش تقاضا و سطح انتظار برای سریع تر شدن وب

مساله این است که امروزه دنیا نیز متقاضی چنین فوریت و سرعتی است. کاربران دوست دارند هر لحظه از اسنادی که مشغول کار روی آنها هستند، شبکه‌های اجتماعی‌شان، بازی‌های آن‌لاین‌شان و بسیاری چیزهای دیگر با خبر باشند. چیزهایی که تعدادشان هر روز هم در حال افزایش است. در واقع به جای این که کاربران خودشان بخواهند به سبک چند سال پیش در پی اطلاعات باشند و اطلاعات مورد نیازشان را به دست بیاورند، دوست دارند اطلاعات خودش به دست آنها برسد.

عوامل دخیل در بالا رفتن سطح انتظار و تقاضای کاربران را می‌توان چنین دانست:

- افزایش پهنای باند
هر سال پهنای باند اینترنت بهبود پیدا کرده و تعداد کاربران بیشتر می‌شود.
- دیجیتالی‌تر شدن دنیا
دنیا هر روز بیش از پیش در حال دیجیتالی شدن است. روزی نیست که سرویس اینترنتی دیگری ارائه شود که توسط آن بتوان کارها را بدون رفت و آمد یا مراجعه‌ی فیزیکی انجام داد.
- گسترش موبایل‌های هوشمند و تبلت‌ها
هر دهه از تاریخ ادوات دیجیتال را می‌توان عصر یک وسیله قلمداد کرد. هر وسیله نیز جایگزین وسیله‌ی دهه‌ی پیش شده است: زمانی ترمینال‌ها (دهه‌ی ۸۰)، سپس کامپیوترهای شخصی (دهه‌ی ۹۰)، بعد لپ‌تاپ‌ها (دهه‌ی ۲۰۰۰) و امروز موبایل‌های هوشمند و تبلت‌ها (دهه‌ی ۲۰۱۰).
- اگر زمانی از تلفن‌های همراه تنها برای تماس استفاده می‌شد، امروز بیشتر کارها را می‌توان با تلفن همراه انجام داد. در واقع استفاده از موبایل برای کارهای غیر تلفنی بیشتر از تماسی است که تلفن همراه برای آن طراحی شده و این دستگاه به نام آن نام‌گذاری شده است. و برنامه‌های موبایلی بار سرورهای اینترنت را به طور چشمگیری افزایش داده است.
- گسترش سرویس‌های آن‌لاین
بورس سهام و ارز، اخبار، شبکه‌های اجتماعی، برنامه‌های چت، بازی‌های آن‌لاین، فروشگاه‌های اینترنتی، ایمیل، کنفرانس‌های ویدئویی، فعالیت‌های بیزینسی و تجارت آن‌لاین، برنامه‌های گروهی آن‌لاین، وبینارها همگی نمونه‌هایی از سرویس‌هایی هستند که به سوی وارونه شدن ارتباط سنتی کلاینت به سرور پیش رفته‌اند.
- تقاضا از جانب بیزینس
شرکت‌های تجاری بیش از پیش مایلند با مشتریان خود ارتباط داشته باشند و مشتریان‌شان را افزایش بدهند. اگرچه گسترش موبایل‌های هوشمند و تبلت، دنیای بیزینس را به تهیه‌ی نسخه‌ی موبایلی سایت‌ها و سرویس‌هایشان سوق داده، اما بیزینس نیز از یک سو آتش بیار معرکه است و با فراهم کردن سرویس‌های جدید، بیشتر به کاربران خوراک می‌دهد و تقاضا و انتظار را بالا می‌برد.

جدیدترین تحول دنیای توسعه‌ی وب: ارتباط real-time

محور همه‌ی بحث‌های وبی همیشه «کاربر» است. به هر حال منبع درآمد و سود، کاربران هستند. از این رو باید به هر نحو ممکن بیشتر رضایت آنها را فراهم کرد. یعنی محصول باید کاربرپسندتر باشد، سریع‌تر باشد، انعطاف‌پذیرتر باشد، کار با آن ساده‌تر باشد و پاسخ‌پذیرتر باشد تا مشتریان بیشتری فراهم کرده و سود بیشتری عاید بیزینس کند. از این رو دامنه‌ی استفاده از وب سرویس‌ها، سرویس‌های REST، Ajax و برنامه‌های موبایل به شدت افزایش پیدا کرده و این حوزه دستخوش تحولات زیادی بوده است.

تا چند سال پیش Ajax تکنیک خیلی خوبی بود اما دیگر به نیاز امروز پاسخ نمی‌دهد. چیز دیگری لازم است و آن چیز، متصل بودن دائمی کلاینت به سرور است. یکی از خوش‌یمن‌ترین تحولاتی که دنیای وب در سال‌های اخیر شاهد آن بود، شکل‌گیری فریم‌ورک‌های MVC جاوااسکریپتی بود که نوشتن برنامه‌های وب را به طور چشمگیری متحول کرده و بار سرورها را کاهش داده است.

تحولات جهان و بیزینس باعث شده دنیای IT (یعنی سازمان‌های استانداردسازی همانند کنسرسیوم جهانی وب یا W3C^۴، شرکت‌های تولیدکننده‌ی مرورگر، شرکت‌های تولیدکننده‌ی سرور و ادوات شبکه‌ای و شرکت‌های سازنده‌ی تکنولوژی‌های نرم‌افزاری) بسترهای بهتری برای برنامه‌های تحت وب پیدا کنند تا با آنها بتوان برنامه‌های وب را سریع‌تر کرد، کاربر را معطل نکرد (یا حداقل کمتر معطل کرد) و از دیدگاه سرعت و پاسخ‌پذیری، تجربه‌ی بهتری برای کاربر فراهم کرد.

این نیازمندی از پیش هم در پروتوکل HTTP وجود داشت. اما پروتوکل HTTP به گونه‌ای تعریف شده بود که توانایی رفع چنین نیازی را به شکل مناسبی نداشت. دنیای IT از این مساله آگاه بود. از این رو اکنون چندین سال است روی روش‌های جدیدی برای وارون کردن ارتباط کلاینت به سرور مشغول کار هستند تا بتوانند ارتباطی فراهم کنند تا سرور بتواند بدون آن که کلاینت درخواست کرده باشد، چیزی برایش بفرستد.

تلاش‌های صورت گرفته

تلاش سازمان‌ها و شرکت‌های IT در سال‌های اخیر به شکل‌گیری پروتوکل‌های جدیدی مانند WebSockets و Server Sent Events منجر شد که تا حدی قابل اعتماد و قابل استفاده هستند. اما هنوز فاصله‌ی زیادی دارند تا به عنوان راه حلی نهایی و جهانی مطرح شوند. زیرا از یک سو کلاینت‌ها و سرورهای متنوعی در اینترنت وجود دارد. از سوی دیگر زیرساخت شبکه چندان قابل دستکاری نیست. از این رو اتخاذ راه‌حل‌های جدید به کندی پیش می‌رود.

اما اینها تنها مشکلاتی هم نیست که هنگام نوشتن برنامه‌های آبی چند کاربره باید رفع شود. ماهیت ارتباط شبکه‌ای به گونه‌ای است که با عوامل غیر قابل پیش‌بینی و ناپایداری همراه است که مدیریت و توزیع پیام بین کاربران را مشکل می‌کند. برای نمونه در یک برنامه‌ی اتاق گفتگو، هر کاربر می‌تواند پهنای باند متفاوتی داشته باشد. حتی این پهنای باندها در حین گفتگو نیز می‌تواند نوسان پیدا کند.

^۴ World Wide Web Consortium: <http://www.w3.org>

در چنین سناریویی سرور باید برای جلوگیری از گم شدن پیام‌ها، آنها را به طور موقت ذخیره کرده و بعد برای گیرندگان ارسال کند. سپس بررسی کند کدام کاربرها واقعا پیام‌های ارسال شده را دریافت کردند یا نکردند. همچنین سرور همیشه باید وضعیت اتصال هر کاربر و مشکلات بالقوه‌ای که می‌تواند حین ارسال پیش بیاید را در نظر بگیرد و اگر ارسال پیامی شکست بخورد دوباره آن را برای کاربر ارسال کند. در واقع نوشتن چنین برنامه‌هایی مانند نوشتن یک سرور SMTP است که قابلیت فوریت یا immediacy مورد نیاز برنامه‌های real-time به آن افزوده شده است. واضح است که پیاده‌سازی چنین سیستمی بسیار مشکل و پیچیده است.

چند نمونه در دنیای واقعی^۵

- Facebook: فیسبوک یکی از شبکه‌های اجتماعی پیشرو در بحث real-time web است که برای پیاده‌سازی قابلیت‌های real-time خود از تکنیک long polling و WebSockets استفاده می‌کند. فیسبوک قابلیت چت و آگاه‌سازی real-time خود را اوایل سال ۲۰۰۸ ارائه کرد و در طول این سال‌ها آن را بهبود داده است.
- تویتر: تویتر برای ارتباط real-time از تکنولوژی خودش استفاده می‌کند.
- اسناد گوگل (Google Docs): گوگل نیز در این برنامه از تکنولوژی خودش استفاده می‌کند.
- Jabbr: یک سرویس اتاق گفتگو و بحث اینترنتی است که توسط مبدعان SignalR تحت ASP.NET نوشته شده و هم اکنون به محل خوبی برای گفتگوی برنامه‌نویسان تبدیل شده است. این برنامه این‌سورس بوده و می‌توانید سورس آن را به طور کامل از github دانلود کنید (<https://github.com/Jabbr>).
- ShootR: بازی آن‌لاین این‌سورسی است که با SignalR و ASP.NET نوشته شده است.

<https://github.com/ntaylorlormullen/shootr>

مایکروسافت و برنامه‌های real-time

تا این اواخر در NET، فناوری‌ای وجود نداشت که بتواند برای پیاده‌سازی چنین برنامه‌هایی راه‌حلی فراهم کند. البته فناوری‌هایی مانند WCF، وب سرویس یا سرویس‌های جدیدتر Web API وجود دارد که با آنها می‌توان سرویس‌هایی ایجاد کرد که کلاینت به طور دائم به سرور وصل باشد. اما هیچ یک از آنها به طور ویژه برای محیط‌های آسنکرون یا غیر همزمان که ارتباط آنی بین کاربرها را فراهم کند طراحی نشده است. اگرچه با فناوری‌های کنونی NET، می‌توان برنامه‌های real-time نوشت، اما چنین کاری حتی برای برنامه‌نویسان خبره و با تجربه هم راحت نیست و بیشتر مواقع سیستم‌هایی به دست می‌دهد که از نظر سرعت، کارایی و مقیاس‌پذیری بسیار مشکل دارند.

در این کتاب نحوه‌ی پیاده‌سازی برنامه‌های real-time را با استفاده از SignalR نشان می‌دهیم. SignalR فریم‌ورک جدید مایکروسافت است که به طور ویژه برای نوشتن برنامه‌های real-time نوشته شده و این کار را بینهایت ساده نموده و در عین حال فریم‌ورکی قابل اعتماد، قدرتمند، انعطاف‌پذیر و مقیاس‌پذیر است.

^۵ Pro ASP.NET SignalR, Apress, 2014

مروری بر محتوای کتاب

در این کتاب نخست مشکلات برنامه‌های real-time چند کاربره را مرور می‌کنیم که کمی پیش تعدادی از آنها را ذکر کردیم. سپس نگاه سریعی به نحوه‌ی کار پروتوکل HTTP و محدودیت‌های آن در پشتیبانی از برنامه‌های real-time انداخته و در ادامه مدل push را توضیح می‌دهیم که همان مدل ارتباط وارون سرور و کلاینت است. همچنین استانداردهای جاری و در دست توسعه‌ی W3C و IETF^۶ را به همراه تکنیک‌های مختلف پیاده‌سازی مدل push تحت پروتوکل HTTP بررسی خواهیم کرد. بدین ترتیب درک عمیقی نسبت به چالش‌های پیش رو در نوشتن برنامه‌هایی که می‌خواهند به داشتن قابلیت‌های فوریت و تعامل آنی با کاربر فخر کنند پیدا خواهیم کرد. این مساله به نوبه‌ی خود درک بهتری نسبت به SignalR و زیربنایی که SignalR بر سر آن بنا شده برای ما فراهم می‌کند.

پس از بررسی مفاهیم زیر بنایی به طور رسمی SignalR را معرفی می‌کنیم، قابلیت‌هایش را توضیح داده و جایگاهش را در میان دیگر فناوری‌های وبی میکروسافت بررسی می‌کنیم. به دنبال آن سطوح انتزاعی SignalR را توضیح می‌دهیم که SignalR آنها را تحت پروتوکل HTTP فراهم کرده و با استفاده از آنها می‌توانیم بدون درگیری با جزئیات سطح پایین پروتوکل HTTP به قابلیت‌های خود برنامه تمرکز کنیم. همچنین از فرصت استفاده کرده و درباره‌ی OWIN و katana صحبت می‌کنیم. دو فریم‌ورک جدید میکروسافت که به سرعت دارند در میان جامعه‌ی برنامه‌نویسان وب جا می‌افتند.

در ادامه به طور عمقی تکنیک‌های SignalR را برای ایجاد برنامه‌های تعاملی چند کاربره‌ی real-time بررسی خواهیم کرد و نحوه‌ی استفاده از آنها را یاد خواهیم گرفت. برای این کار از مثال‌های مختلفی استفاده می‌کنیم تا بتوانیم به طور عملی زیر بنای SignalR را درک کرده و ببینیم چگونه می‌توانیم از SignalR در پروژه‌های واقعی استفاده کنیم.

با وجودی که به نظر می‌رسد SignalR برای وب طراحی شده، اما این فناوری فراتر از وب است و با آن می‌توان برای هر نوع برنامه‌ای سرویس real-time ایجاد کرد و آن را توسط هر نوع کلاینت و هر سیستمی استفاده نمود. در واقع SignalR به طور کامل مستقل از وب است. نشان می‌دهیم SignalR چگونه چنین طراحی شده است و مثال‌های مختلف استفاده از SignalR را در انواع مختلف برنامه‌های NET. نشان خواهیم داد.

یکی از چیزهای مهم دیگری که توضیح خواهیم داد و چندین صفحه را هم به آن اختصاص داده‌ایم، نحوه‌ی انتشار و مقیاس‌پذیر کردن برنامه‌های SignalR است. ابزارهای از پیش آماده‌ی SignalR را در این زمینه بررسی کرده و برای سناریوهایی که ابزارهای توکار SignalR در آنها کافی نیست راه‌حل‌های دیگری ارائه خواهیم کرد. همچنین تکنیک‌های دیگری معرفی می‌کنیم که برای سناریوهای با همروندی بسیار بالا و نظارت بر سرورها و بهبود کارایی آنها ابداع شده‌اند.

در نهایت به جنبه‌های پیشرفته‌ی برنامه‌نویسی SignalR می‌پردازیم. این موضوعات پیشرفته بیش از عمیق تری نسبت به نحوه‌ی کار این فریم‌ورک، امنیت آن، نحوه‌ی ایجاد مولفه‌های تزریق‌پذیر توسط آن، امکان توسعه‌ی خود این فناوری، چگونگی اجرای آزمون واحد و چندین موضوع جالب دیگر برای ما فراهم خواهد کرد.

فصل ۲. HTTP و مدل Push

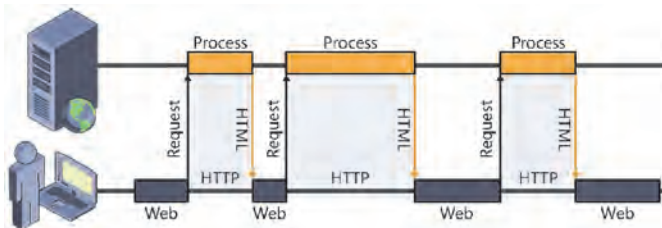
HTTP یا پروتوکل انتقال فرا متن (HyperText Transfer Protocol) قرارداد یا زبانی است که کلاینت‌ها و سرورهای برنامه‌های وب توسط آن با یکدیگر صحبت می‌کنند. این پروتوکل در سال ۱۹۹۶ ابداع شد^۱ و مدل ساده‌ای داشت، به گونه‌ای که موفقیت وب تا حدی مرهون سادگی و چندکارگی پروتوکل HTTP است. اگرچه استفاده از پروتوکل HTTP در سناریوهای معمولی وب ساده است، اما سناریوهای دیگری مانند سرویس‌های real-time وجود دارد که این پروتوکل در آنها به روشنی محدودیت دارد.

مدل pull

پروتوکل HTTP به سبک درخواست-پاسخ (request-response) کار می‌کند. هنگامی که کلاینت می‌خواهد به منبعی که توسط سرور ارائه شده دسترسی پیدا کند، اتصالی به سرور ایجاد کرده و با استفاده از قرارداد HTTP، منبع مورد نیاز را درخواست می‌کند. از آن سو سرور درخواست دریافت شده را بررسی کرده و منبع را بر می‌گرداند (منبع می‌تواند یک فایل یا خروجی حاصل از اجرای یک پردازش باشد). به دنبال آن سرور اتصال ایجاد شده را به طور یکجانبه قطع می‌کند. آغاز کننده‌ی عملیات نیز همیشه کلاینت است. به چنین مدلی مدل pull گفته می‌شود.

اگر کلاینت بخواهد با منبع دیگری کار کند این سیر از نو اجرا می‌شود: کلاینت اتصالی به سرور ایجاد می‌کند، درخواستش را ارسال کرده و منتظر می‌ماند، سرور درخواست را دریافت کرده، پاسخ آن را بر می‌گرداند و در نهایت اتصال توسط سرور قطع می‌شود. این سیر هر بار که صفحه‌ی وبی را در اینترنت مشاهده می‌کنیم به ازای خود صفحه و همه‌ی اجزای خارجی آن مانند تصاویر، کلیپ‌ها، اسکریپت‌ها، فایل‌های شیوه‌نامه و غیره اجرا می‌شود.

در شکل ۱-۲ سیر ارتباط HTTP نشان داده شده است. همان گونه که می‌بینید این سیر سنکرون یا همزمان است. یعنی پس از آن که کلاینت درخواستش را برای سرور ارسال کرد مجبور است تا زمان دریافت پاسخ صبر کند و تحت اتصال ایجاد شده کار دیگری نمی‌تواند انجام بدهد. البته می‌تواند اتصال دیگری ایجاد کرده و درخواست دیگری طی آن بفرستد، اما تحت هر اتصال باید تا زمان دریافت پاسخ صبر کند.



شکل ۱-۲. مکالمه‌ی یک مرورگر و یک وب سرور از طریق پروتوکل HTTP

^۱ Specification of HTTP 1.0: <http://www.w3.org/Protocols/HTTP/1.0/spec.html>

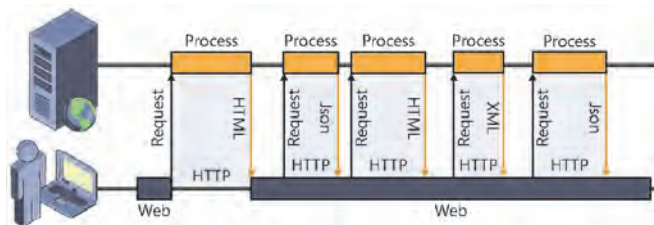
محدودیت‌های پروتکل HTTP

پروتکل HTTP با وجود سادگی‌اش محدودیت‌های بزرگی دارد:

- **بی‌وضعیت یا stateless است**
زیرا اتصال کلاینت و سرور دائمی نیست. پس از یک رفت و برگشت HTTP، ارتباط قطع می‌شود. از این رو بدون یک مکانیزم اضافی، نمی‌توان وضعیتی را به طور مشترک بین کلاینت و سرور حفظ کرد.
- **ارتباط ایجاد شده سنکرون است**
تا زمان دریافت پاسخ، تحت اتصال کنونی هیچ کار دیگری جز لغو کردن درخواست نمی‌شود کرد. البته همان گونه که گفته شد می‌توان درخواست دیگری به طور موازی به سمت سرور ارسال کرد، اما تحت هر اتصال جز صبر کردن تا زمان دریافت پاسخ کار دیگری نمی‌توان انجام داد.
- **اتصال ایجاد شده دائمی نیست**
اتصال HTTP بین سرور و کلاینت دائمی نیست. سرور پس از برگرداندن پاسخ، اتصال را قطع می‌کند.
- **هر اتصال تنها برای یک درخواست قابل استفاده است**
پس از ارسال درخواست، نمی‌توانید درخواست دیگری به دنبال آن بفرستید. هر منبع باید طی یک اتصال و درخواست جداگانه از سرور درخواست شود.
- **اتصال HTTP یک بار مصرف است**
این بیان دیگری از محدودیت بالایی است. یعنی هر اتصال تنها برای یک درخواست قابل استفاده است.
همان گونه که می‌بینید سه مورد آخر تقریباً هر سه یک چیز را به شکل مختلف بیان می‌کنند.

بهبود وب با Ajax

یکی از بهبودهایی که برای وب ایجاد شد تکنیکی به نام Ajax یا Asynchronous Javascript and XML بود که با آن می‌توان نیاز غیرهمزمانی برنامه‌های وب را بر طرف کرد. با Ajax می‌توانیم در سمت کلاینت و پس از بارگذاری صفحه بدون ترک صفحه یا بلاک شدن اجرای آن، تحت پروتکل HTTP درخواستی برای سرور ارسال کرده و پاسخ آن را استفاده کنیم. سیر Ajax در شکل ۲-۲ نشان داده شده است.



شکل ۲-۲. سیر AJAX در یک صفحه‌ی وب

مزیت بزرگ Ajax که باعث پیدایش سایت‌ها و سرویس‌های تعاملی بسیار زنده‌ای مانند فیسبوک و جیمیل گردیده این است که عملیات آن آسنکرون یا غیر همزمان است. یعنی کاربر می‌تواند در حین ارسال درخواست‌های Ajax به کار با صفحه ادامه بدهد.

با این حال عملیات Ajax نیز تحت پروتوکل HTTP و طبق همان مدل request-response انجام می‌شود. یعنی صرفنظر از ماهیت آسنکرون Ajax، به هر حال باز هم همیشه کلاینت ارتباط را آغاز می‌کند، اتصال ایجاد شده یک طرفه و یک بار مصرف است و سرور پس از ارسال پاسخ بی‌درنگ و به طور یکجانبه اتصال را قطع می‌کند. اگر کلاینت به چیز دیگری نیاز داشته باشد باید درخواست دیگری به سمت سرور بفرستد.

محدودیت Ajax

Ajax تکنیک بسیار مفیدی است، اما سناریوهایی وجود دارد که به طور کلی در آنها ناتوان است. برای نمونه نوشتن برنامه‌های چت، ابزارهای وبی چند نفره، بازی‌های آن‌لاین چندکاربره یا سرویس‌های real-time، حتی با ماهیت آسنکرون Ajax چندان راحت نیست.

پاسخ کاملاً ساده است: مشکل به Ajax مربوط نمی‌شود. این HTTP است که برای ارتباط بلادرنگ و real-time طراحی نشده است. در حالی که پروتوکل‌های دیگری مانند پروتوکل محبوب IRC ارتباط کاملاً چابکی فراهم می‌کنند و با آنها می‌توان سرویس‌هایی پویاتر و تعاملی‌تری نسبت به مدل pull ایجاد کرد. در چنین پروتوکل‌هایی سرور هم می‌تواند تحت اتصال شکل گرفته هر زمان که بخواهد اطلاعاتی را برای کلاینت بفرستد، بدون آن که منتظر شود ابتدا کلاینت به او درخواست بدهد.

مدل push

برنامه‌های گفتگوی آنی (instant messaging)، ابزارهای تحت وب گروهی، بازی‌های آن‌لاین چند کاربره یا هر نوع سیستمی که در آن باید به محض تولید اطلاعات، آن را برای کاربران ارسال کنیم نمونه‌های کوچکی هستند که مدل سنتی pull در آنها فاقد کارایی است. در چنین برنامه‌هایی به جای این که سرور منتظر درخواست (یا سرکشی) کلاینت بماند، خودش باید بتواند اطلاعات را برای کلاینت بفرستد. این دقیقاً ایده‌ی مدل push یا server push است. این نام نیز به هیچ مولفه، تکنولوژی یا پروتوکلی اشاره نمی‌کند و تنها یک مفهوم است. یک جور ارتباط کلاینت و سرور است که در آن سرور آغازگر مبادله است.

البته ایده‌ی push چیز جدیدی نیست و در گذشته پروتوکل‌هایی با مدل push نیز وجود داشته است، مانند پروتوکل IRC که در سرویس‌های چت قدیمی به کار رفته یا پروتوکل SMTP که برای مدیریت ایمیل استفاده می‌شود. این پروتوکل‌ها پیش از آن که اصطلاح push یا server push ابداع شود ایجاد شده‌اند.

برای این که سرور بتواند کلاینت‌ها را بی‌درنگ از رویدادهای سمت خودش آگاه کند، در بهترین حالت باید اتصال نقطه به نقطه‌ی^۲ مستقیمی میان او و کلاینت وجود داشته باشد و اگر چنین اتصالی وجود ندارد خودش باید بتواند آن را ایجاد کند و اطلاعات را تحت آن برای کلاینت بفرستد.

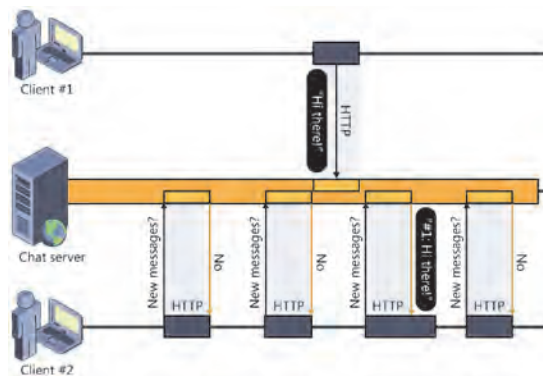
^۲ peer-to-peer

برای نمونه یک سرور چت می‌تواند IP کسانی را که وارد اتاق شده‌اند حفظ کرده و هنگامی که کسی در یک اتاق پیام می‌فرستد، بر اساس شماره IP حاضرین اتاق، اتصال به آنها ایجاد کرده و پیام را برای آنها بفرستد.

اما از نظر فنی چنین چیزی حتی در پروتوکل IRC میسر نیست. زیرا به دلایل امنیتی نمی‌توان به طور مستقیم به کلاینت متصل شد. حتی ممکن است سطوح میانی بین راه مانند دیواره‌های آتش، روترها یا پراکسی‌ها از برقراری چنین اتصالی جلوگیری کنند. از این رو همیشه کلاینت است که باید ارتباط را آغاز کند. با وجودی که سرور نمی‌تواند خودش به کلاینت متصل شود، اما یک راه وجود دارد. پس از آن که کلاینت درخواستی برای سرور می‌فرستد، سرور می‌تواند اتصال او را قطع نکند.

تکنیک سرکشی یا polling

اگر از برنامه‌نویسان وب بپرسید برای حل سناریویی که در آن سرور بخواهد تحت پروتوکل HTTP بدون دریافت درخواستی از سوی کلاینت اطلاعاتی برایش بفرستد چه راه حلی پیشنهاد می‌دهید نخستین راه ابتکاری که به ذهن‌شان می‌رسد تکنیک سرکشی یا polling است. در این تکنیک کلاینت به طور دوره‌ای به سرور سر می‌زند و از او می‌پرسد آیا چیزی برای من داری یا خیر. اگر سرور اطلاعاتی برای ارسال داشته باشد آن را بر گردانده و در غیر این صورت پاسخ خالی بر می‌گرداند. اتصال نیز طبق معمول توسط سرور به سرعت قطع می‌شود. این مدل در شکل ۳-۲ نشان داده شده است.



شکل ۳-۲. تکنیک polling که در یک برنامه‌ی چت استفاده شده است

این نشان می‌دهد برنامه‌نویسان وب از پیش با سیستم کاری وب آشنا هستند و می‌دانند سرور خودش نمی‌تواند چیزی برای کلاینت بفرستد. از این رو برای حل مساله آن را وارونه می‌کنند و می‌گویند حال که سرور نمی‌تواند با کلاینت ارتباط برقرار کند، کلاینت را مجبور می‌کنیم هر بار به او سر بزنند و از او بپرسد آیا چیزی داری به من بدهی.

نقاط قوت و ضعف تکنیک polling

مزیت بزرگ تکنیک polling سادگی پیاده‌سازی آن است. ضمن این که همه جا، در هر سناریویی، با هر مرورگر و سروری می‌توان آن را به کار برد. زیرا از هیچ چیزی جز قابلیت‌های استاندارد و از پیش موجود وب مانند پروتوکل HTTP و جاوااسکریپت استفاده نمی‌کند. اما نباید فراموش کرد که مدل استفاده شده باز هم pull است.

با این وجود گاهی هزینه‌ی تکنیک polling بسیار زیاد می‌شود. زیرا ایجاد اتصال به سمت سرور و قطع و وصل مکرر و همزمان آن توسط هزاران کلاینت، از نظر مصرف منابع (پهنای باند و پردازش سمت سرور)، هزینه‌ی زیادی برای سرور و کلاینت ایجاد می‌کند و این هزینه با زیاد شدن تعداد کلاینت‌ها بیشتر و بیشتر می‌شود.

تصور کنید روی سروری برنامه‌ی وبی بر پا شود که در آن اطلاعاتی از صفحه به طور آنی به روز می‌شود. سپس هزاران کاربر بخواهند از آن استفاده کنند. هر بار که صفحه‌های این کاربرها سرور را چک می‌کند، هزاران اتصال به سرور شکل گرفته، هزاران درخواست ارسال شده و سپس اتصال‌ها قطع می‌شود. سرور نیز باید به همه‌ی درخواست‌ها سرویس بدهد، چه پاسخی داشته باشد چه نداشته باشد. و این سیر بی‌انتهای ادامه دارد. و اگر پای منیع‌کننده‌ی مانند دیتابیس و فایل به میان کشیده شود اوضاع بدتر می‌شود. سرور باید با سرعت هرچه تمام‌تر اطلاعات را بازیابی کند تا کلاینت‌ها با تأخیر مواجه نشوند.

چیزی که کار را باز هم مشکل می‌کند دستکاری داده‌های برنامه است. زیرا در برنامه‌های همروند و چند کاربره برای تضمین جامعیت داده‌ها هنگام دسترسی به داده‌های مشترک الزاماً باید از ساختار قفل و اجرای ترتیبی استفاده کرد. از این رو در حین این که سرور به درخواست یکی از کلاینت‌ها برای دستکاری اطلاعات پاسخ می‌دهد درخواست‌های دیگر کلاینت‌ها برای دستکاری همین اطلاعات توسط سرویسی که مدیریت داده‌های مشترک را بر عهده دارد (مانند DBMS) بلاک می‌شود. در نتیجه در ترافیک بالا سرور به وضوح به نفس زدن می‌افتد.

البته این مساله ارتباطی به مدل pull یا push ندارد و اجرای ترتیبی در چنین شرایطی الزامی است. اما تصورش را بکنید با آن همه باری که روی دوش سرور است بخواهیم با تکنیک polling هزاران هزار درخواست دیگر را هم به سرور تحمیل کنیم.

بهبود تکنیک polling

برای رفع مشکلات تکنیک polling یا حداقل کاهش دادن آن تکنیک‌های مختلفی ابداع شد. یکی از آنها «بازه‌های سرکشی مطابقت داده شده» یا adaptive intervals بود. یعنی کلاینت بازه‌ی زمانی سرکشی را به تناسب بار سرور یا زمانی که سرور قصد داشته باشد عملیات سنگینی اجرا کرده و نتواند به سرعت به درخواست‌های سرکشی رسیدگی کند، کم و زیاد کند. یعنی وقتی سرور زیر بار قرار دارد و به سرعت نمی‌تواند پاسخ بدهد، بازه‌ی زمانی سرکشی را طولانی کرده و دیرتر درخواست بدهیم تا بار اضافی به سرور تحمیل نکنیم (اجازه بدهیم بار سرور کم شود). این تکنیک بسیار مفید است و در برخی سناریوها به طور چشمگیری می‌تواند مصرف منابع را کاهش بدهد.

یک راه دیگر هم این است که درخواست‌های سرکشی را تنها زمانی ارسال کنیم که برگه یا صفحه‌ی جاری فعال بوده و کاربر نیز اشاره‌گر ماوس را در صفحه حرکت بدهد. اگر صفحه‌ی فعلی فعال نباشد یا مدتی سپری شده و اشاره‌گر ماوس حرکت نکند می‌توانیم احتمال بدهیم کاربر از پشت کامپیوتر بلند شده است. در این شرایط نیازی نیست سرور را با درخواست‌های بی‌دلیل بمباران کنیم. زمانی موتور ماشین سرکشی را دوباره روشن می‌کنیم که صفحه‌ی جاری فعال شود یا اشاره‌گر ماوس را حرکت کند.