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 پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱٤، واحد ۱٦ **www.pendarepars.com**

تلفن: ٦٦٥٧٢٣٣٥ - تلفکس: ٦٦٩٢٦٥٧٨همراه: ۰۹۱۲۲٤٥۲۳٤۸ **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**

فهرست

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 کُدهایی درگیر می‌شدند که برنامه‌های آنها را کثیف و شلوغ می‌کرد. خوشبختانه مایکروسافت 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 مایکروسافت 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 کاربره‌ی 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 تک‌کاربرگی خشک و بی روح 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 تقریبا 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 سریع‌تر 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 خوش‌یُمن‌ترین 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 تولید کننده‌ی مرورگر، شرکت‌های تولید کننده‌ی 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 کُندی پیش می‌رود.

اما اینها تنها مشکلاتی هم نیست که هنگام نوشتن برنامه‌های آنی 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 Consurtium: http://www.w3.org

در چنین سناریویی سرور باید برای جلوگیری از گُم 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.
- توییتر: توییتر 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 اپن‌سورس بوده و می‌توانید سورس آن را به طور کامل از github دانلود کنید (https://github.com/JabbR).
- ShootR: بازی آن‌لاین اُپن‌سورسی است که با SignalR و ASP.NET نوشته شده است.
  https://github.com/ntaylormullen/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 فریم‌وُرکی 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 مایکروسافت 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 مایکروسافت 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 چطور 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 بینش عمیق‌تری 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.

---

[۶] سازمان Internet Engineering Task Force به آدرس http://www.ietf.org

فصل ۲. HTTP و مدل Push

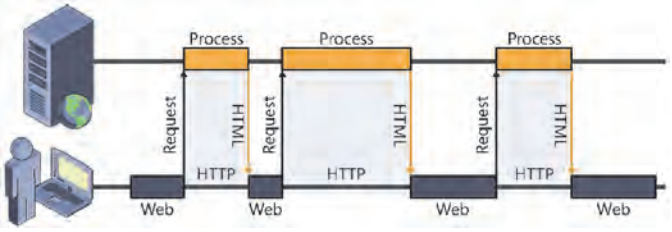
HTTP یا پروتوکل انتقال فرا متن (HyperText Transfer Protocol) قرارداد یا زبانی است که کلاینت‌ها و سرورهای برنامه‌های وب توسط آن با یکدیگر صحبت می‌کنند. این پروتوکل در سال ۱۹۹۶ ابداع شد[۱] و مدل ساده‌ای داشت، به گونه‌ای که موفقیت وب تا حدی مرهون سادگی و چندکارگی پروتوکل HTTP است. اگرچه استفاده از پروتوکل 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 پروتوکل HTTP

پروتوکل 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.

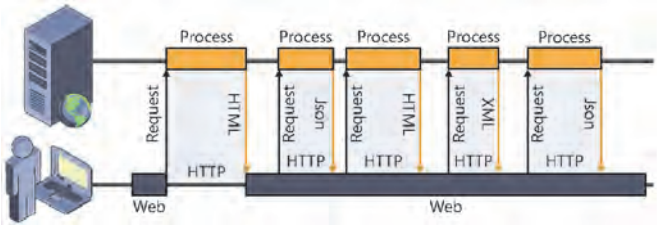
- **اتصال HTTP یک بار مصرف است**

این بیان دیگری از محدودیت بالایی است. یعنی هر اتصال تنها برای یک درخواست قابل استفاده است.

همان گونه که می‌بینید سه مورد آخر تقریبا 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 پروتوکل 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 کاملا 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 کاملا 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 دقیقا 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.

---

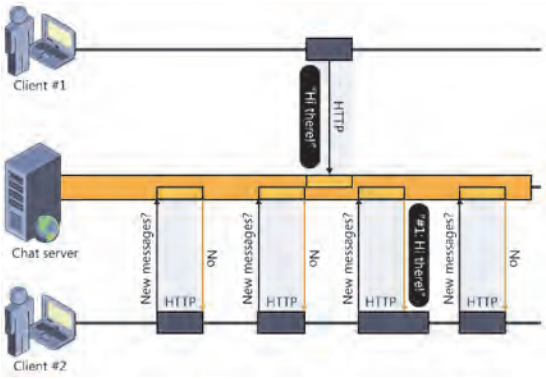
[۲] peer-to-peer

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، اتصالی 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 هربار به او سر بزند 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 بی‌انتها ادامه دارد. و اگر پای منبع کُندی مانند دیتابیس و فایل به میان کشیده شود اوضاع بدتر می‌شود. سرور باید با سرعت هرچه تمام‌تر اطلاعات را بازیابی کند تا کلاینت‌ها با تاخیر 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 الزاما 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.