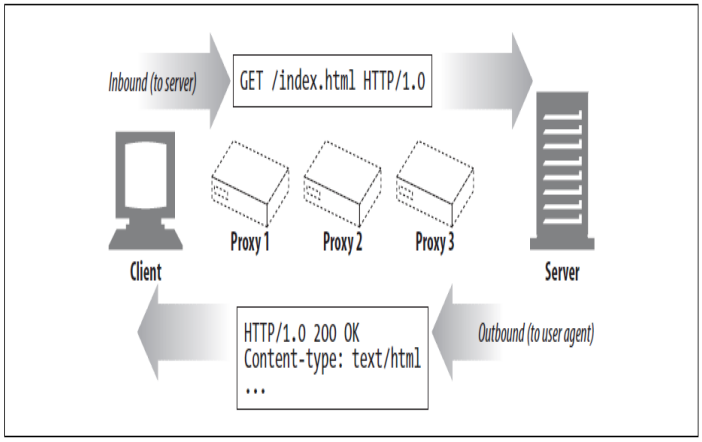


شکل ۱-۱ ارسال پیام به سرور و بازگشت به کاربر

پیام‌های HTTP همانند رودخانه جریان دارند. همه پیام‌ها در پایین دست، صرف نظر از اینکه پیام‌های درخواست یا پیام‌های پاسخ هستند، جریان می‌یابد. پیام‌های HTTP ساده هستند که بلوکی از فرمت داده‌ها می‌باشند. هر پیام شام یک درخواست از یک مشتری با یک پاسخ از طرف سرور همراه است. این پیام‌ها شامل سه بخش؛ شروع خط توصیف پیام، بلوکی از سرصفحه و یک بدنه اختیاری حاوی داده می‌باشد.

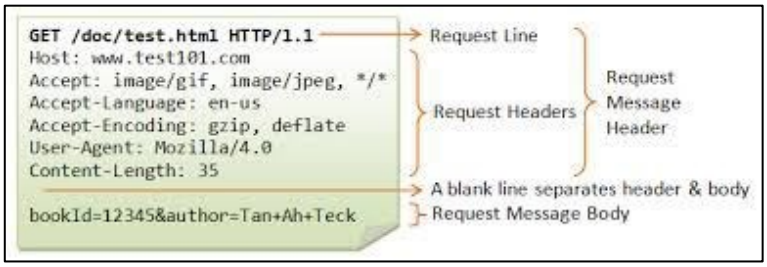
متدهای[۱] پروتکل HTTP

پروتکل انتقال ابرمتن روش‌هایی را برای درخواست تعریف کرده است که هر کدام از آن‌ها باعث انجام عمل خاص در سمت سرور می‌شوند. این روش، خط شروع درخواست را آغاز می‌کند و به سرور می‌گوید که چه کاری باید انجام دهد. به عنوان مثال، خط «GET /specials/saw-blade.gif HTTP/۱,۰» در متد GET انجام می‌گیرد.

مشخصات HTTP مجموعه‌ای از روش‌های درخواست مشترک را تعریف می‌کند. به عنوان مثال، روش GET یک سند از یک سرور دریافت می‌کند، روش POST داده‌ها پردازش شده را به سرور ارسال می‌کند و متد OPTION قابلیت‌های کلی وب

[۱] HTTP Methods

سرور یا قابلیت‌های یک وب سرور را برای منبع خاص را تعیین می‌کند. جدول ۱-۱ و شکل ۱-۲ شمای کلی از یک درخواست را نشان می‌دهد.



شکل ۱-۲ شمای کلی از یک Request

جدول ۱-۱ متدهای تعریف شده در پروتکل HTTP

نام متد	توضیح
GET	درخواست دریافت یک سند از سرور
HEAD	درخواست دریافت فقط سرآیند یک سند از سرور
POST	ارسال داده به سرور برای پردازش
PUT	درخواست ذخیره یک منبع بر روی سرور
TRACE	برگرداندن تقاضای همین پیام درخواست
OPTIONS	تقاضای داده در خصوص موارد خاص
DELETE	درخواست حذف یک سند از روی سرور

آنچه که از متدها در جدول بالا اشاره شده، این است که همه سرورها، تمام هفت روش گفته شده را اجرا نمی‌کنند. علاوه بر این، به دلیل اینکه HTTP به راحتی طراحی شده است، سرورهای دیگر می‌توانند متدهای درخواست خود را علاوه بر این‌ها اجرا کنند. این روش‌های اضافی، روش‌های توسعه یافته هستند، زیرا که ویژگی HTTP را گسترش می‌دهد.

متد GET

این روش رایج‌ترین متد است. معمولاً برای درخواست ارسال یک منبع به سرور استفاده می‌شود. پروتکل HTTP ۱,۱ نیاز به سرور برای اجرای این روش دارد. شکل ۱-۳ یک نمونه از یک کلاینت که درخواست HTTP را با روش GET ایجاد می‌کند.

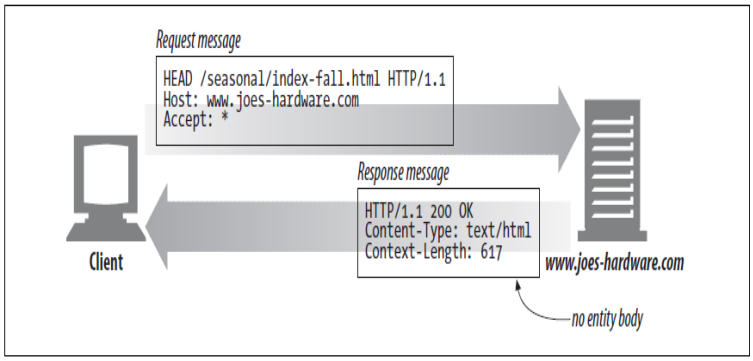
شکل ۱-۳ مثالی از متد GET

متد HEAD

روش HEAD دقیقاً مانند روش GET رفتار می‌کند، اما سرور تنها هدرهای پاسخ را باز می‌گرداند. این اجازه را می‌دهد که یک کلاینت هدرهای یک منبع را در واقع بدون نیاز به خود منبع دریافت کند. با استفاده از متد HEAD می‌توان:

- در مورد یک منبع (به عنوان مثال، نوع آن را تعیین کنید) بدون اینکه آن را دریافت کنید.
- اگر یک شی وجود دارد، با نگاه کردن به کد وضعیت آن پاسخ می‌دهد.
- اگر منبع با اصلاح همراه بود، با نگاه کردن به هدر آن، تست می‌شود.

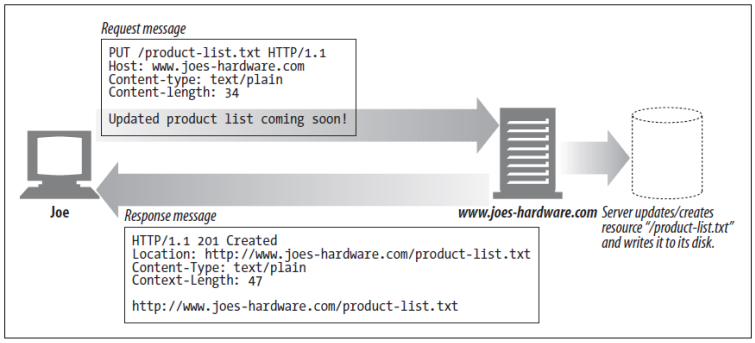
توسعه دهندگان سرور باید این اطمینان را حاصل نمایند که هدرها دقیقاً همان چیزی هستند که درخواست GET به آنها برمی‌گردد. شکل ۱-۴ متد HEAD را در عمل نشان می‌دهد.



شکل ۱-۴ مثالی از متد HEAD

متد PUT

روش PUT اسناد را بر روی یک سرور ذخیره می‌کند، در صورتی که برعکس GET می‌باشد که اسناد را از روی سرور می‌خواند. برخی از سیستم‌های منتشر شده به شما این اجازه را می‌دهند که صفحات وب را ایجاد کرده و آنها را مستقیماً در وب سرور با استفاده از PUT نصب کنید (با توجه به مثال شکل ۱-۵).



شکل ۱-۵ مثالی از متد PUT

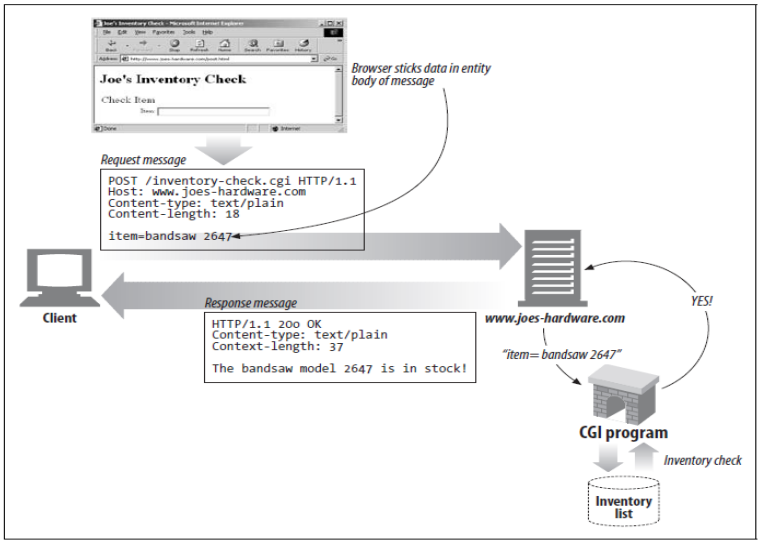
روش PUT بدین صورت است که سرور برای گرفتن درخواست از بدنه است یا برای ایجاد یک سند جدید به نام URL[۱] که درخواست شده، استفاده می‌شود و یا اگر این

[۱] Uniform Resource Locator (آدرس یک فایل در اینترنت)

URL در حال حاضر وجود دارد، از بدنه برای جایگزینی آن استفاده می‌کند. از آنجائیکه PUT این امکان را به شما می‌دهد محتویات را تغییر دهید، بسیاری از وب سرورها برای ورود به سیستم قبل از اینکه بتوانید PUT را اجرا کنید، باید رمز عبور را وارد کنید.

متد POST

این متد برای ارسال داده ورودی به سرور طراحی شده است. در این متد اغلب برای پشتیبانی از اشکال HTML مورد استفاده قرار می‌گیرد. داده‌ها از فایل‌هایی پر شده و به طور معمول به سرور ارسال می‌شود و سپس به جایی که لازم است به آن برسد ارسال می‌شود (به عنوان مثال؛ به یک برنامه دروازه سرور، که پس از آن پردازش می‌شود). شکل ۱-۶ یک کلاینت را که درخواست یک داده HTTP را به یک سرور ارسال می‌کند را نشان می‌دهد که با روش POST صورت گرفته است.

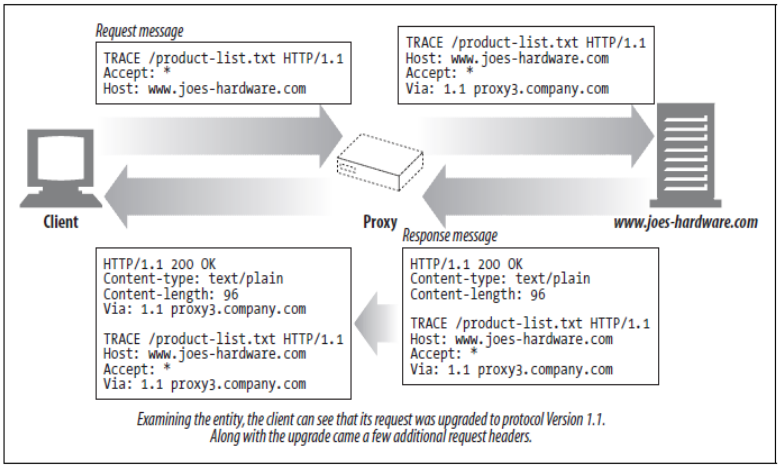


شکل ۱-۶ مثالی از متد POST

متد TRACE

هنگامی که یک کلاینت یک درخواست را ایجاد می‌کند، ممکن است این درخواست از طریق فایروال‌ها، پروکسی‌ها، دروازه‌ها و یا سایر برنامه‌ها مورد بررسی قرار گیرد. هر کدام از اینها دارای امکان تغییر درخواست اولیه HTTP هستند. در این متد سرور اطلاعات درخواست شده را به همان شکلی که دریافت کرده است به کلاینت باز می‌گرداند.

یک درخواست TRACE به «تشخیص حلقه» در سرور مقصد را آغاز می‌کند. سرور در بخش پایانی مسیر یک پاسخ TRACE را با پیام درخواست برمی‌گرداند که از بدنه آن را دریافت کرده است. یک کلاینت می‌تواند پیام اصلی خود را در این زنجیره مسیر درخواست تا پاسخ مشاهده کند که هرگونه برنامه کاربردی HTTP در آن مداخله کردند (با توجه به مثال شکل ۱-۷).



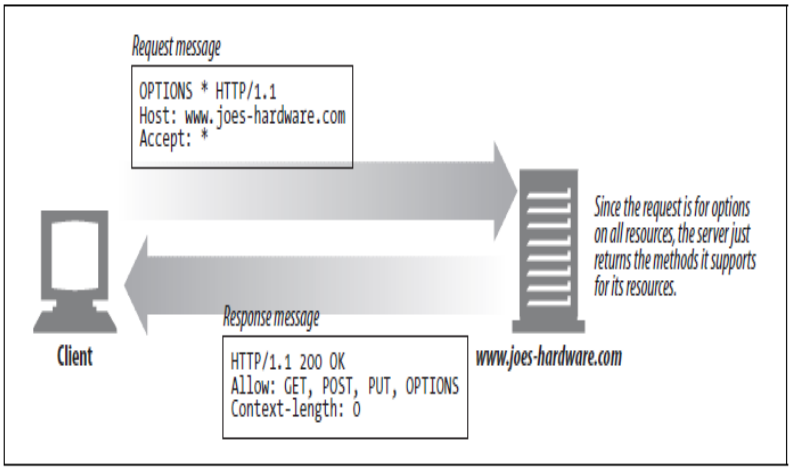
شکل ۱-۷ مثالی از متد TRACE

روش TRACE در درجه اول برای تشخیص استفاده می‌شود، یعنی این متد برای برای اشکال‌زدایی و بررسی سلامت سرور کاربرد دارد. این نیز یک ابزار خوب برای مشاهده کردن اثرات پروکسی‌ها و دیگر برنامه‌های کاربردی بر روی درخواست‌های شما است.

بسیاری از برنامه‌های HTTP بسته به آن متد، روش‌های متفاوتی را انجام می‌دهند. مثلا یک پروکسی ممکن است یک درخواست POST را مستقیماً به سرور منتقل کند، اما سعی می‌کند درخواست GET را به یک برنامه HTTP دیگر (مانند حافظه کش وب) ارسال کند. متد TRACE یک مکانیسمی را برای تشخیص روش‌ها ارائه نمی‌دهد. به طور کلی، برنامه‌ها تداخل ایجاد می‌کنند که چگونگی TRACE یک درخواست را پردازش می‌کند.

متد OPTIONS

این متد از سرور می‌خواهد که در مورد قابلیت‌های مختلف پشتیبانی شده وب سرور به ما بگوید. شما می‌توانید از یک سرور در مورد روش‌هایی که به طور کلی یا برای منابع خاصی پشتیبانی می‌شود، سوال کنید (بعضی از سرورها ممکن است عملیات خاصی را فقط بر روی انواع خاصی از اشیا پشتیبانی می‌کنند).

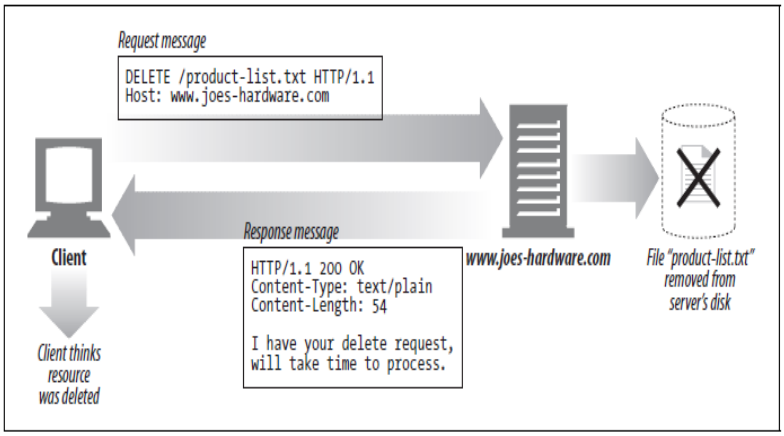


شکل ۱-۸ مثالی از متد OPTION

این متد بدین معنی است که برنامه‌های کاربردی مشتری برای تعیین چگونگی بهترین دسترسی به منابع مختلف، بدون نیاز به آنها دسترسی داشته باشد. شکل ۱-۸ یک سناریوی از درخواست با استفاده از روش OPTION را نشان می‌دهد.

متد DELETE

این متد تنها چیزی است که شما فکر می‌کنید منابع مشخص شده توسط URL درخواست شده از روی سرور می‌خواهد حذف شود. با این حال، در برنامه سرویس گیرنده این تضمین وجود ندارد که حذف انجام شود. به این دلیل است که HTTP به سرور اجازه می‌دهد تا درخواست با احراز هویت صادر کننده صورت گیرد و بدون اطلاع آن لغو گردد. شکل ۱-۹ مثالی از متد DELETE را ارائه می‌کند.



شکل ۱-۹ مثالی از متد DELETE

کدهای وضعیت

این کدهای وضعیت به سرور می‌گوید که چه کاری باید انجام شود، کدهای وضعیت به مشتری اطلاع می‌دهد که چه اتفاقی افتاده می‌افتد. کدهای وضعیت در خط شروع

پاسخ قرار دارند. به عنوان مثال؛ در خط «HTTP/1.0 200 OK»، کد وضعیت ۲۰۰ است.

وقتی مشتریان پیام‌های درخواست را به سرور HTTP ارسال می‌کنند، چیزهای زیادی می‌تواند اتفاق بیافتد. اگر خوش‌شانس باشید، درخواست با موفقیت انجام خواهد شد. سرور ممکن است به شما بگوید که منابع مورد نظر شما یافت نشد که شما مجوز دسترسی به منابع و یا شاید این منبع را ندارید که جایی دیگر جابه‌جا شده است.

کدهای وضعیت در خط شروع هر پیام پاسخ قرار می‌گیرد. هر دو وضعیت عددی و متنی قابل خواندن هستند که کد عددی خطا برنامه‌ها را به آسانی پردازش می‌کند، در حالی که دلیل عبارت این است به راحتی توسط انسان قابل درک است.

کدهای مختلف وضعیت با کدهای سه رقمی به کلاس‌هایی طبقه بندی می‌شوند. کدهای وضعیت بین ۲۰۰ تا ۲۹۹ نشان دهنده موفقیت است. کد بین ۳۰۰ تا ۳۹۹ نشان می‌دهد که منبع منتقل می‌شود. کد بین ۴۰۰ تا ۴۹۹ به این معنی است که مشتری درخواست اشتباهی کرده است. کدهای بین ۵۰۰ تا ۵۹۹ چیزی است که بر روی سرور قرار دارد. جدول ۱-۲ طبقه بندی وضعیت کد را نشان می‌دهد.

جدول ۱-۲ طبقه بندی کد وضعیت

طبقه	محدوده تعریف شده	محدوده کل
اطلاعاتی[۱]	۱۰۰-۱۰۱	۱۰۰-۱۹۹
موفقیت‌آمیز[۲]	۲۰۰-۲۰۶	۲۰۰-۲۹۹
انتقال[۳]	۳۰۰-۳۰۵	۳۰۰-۳۹۹
خطای مشتری[۴]	۴۰۰-۴۱۵	۴۰۰-۴۹۹
خطای سرور[۵]	۵۰۰-۵۰۵	۵۰۰-۵۹۹

۱ Informational
۲ Successful
۳ Redirection
۴ Client error
۵ Server error

نسخه‌های فعلی HTTP فقط چند کد برای هر دسته وضعیت را تعریف می‌کند. همانطور که پروتکل تکامل یافته است، کدهای وضعیت بیشتر به صورت رسمی در مشخصات HTTP تعریف خواهند شد. اگر شما یک کد وضعیت دریافت کنید که تشخیص ندادید، احتمالاً آن را به عنوان یک فرمت توسعه یافته به پروتکل جاری تعریف می‌کند. شما باید با آن به عنوان عضوی از کلاس قرار گرفته رفتار کنید. بعنوان مثال؛ اگر شما کد وضعیت ۵۱۵ را دریافت کنید (که این کد خارج از محدوده تعریف شده لیست کدهای ۵XX در جدول ۱-۲ نشان داده شده است)، شما باید پاسخ را به عنوان یک خطای سرور که در کلاس کلی از پیام‌های ۵XX است، تلقی کنید.

جدول ۱-۳ متداول‌ترین کدهای وضعیت

کد وضعیت	دلیل عبارت	توضیح
۲۰۰	مجاز	موفقیت! هر اطلاعات درخواست شده در بدنه پاسخ می‌باشد.
۴۰۱	غیرمجاز	شما باید یک نام کاربری و رمز عبور را وارد کنید.
۴۰۴	-	سرور نمی‌تواند یک منبع برای URL درخواست شده پیدا کند.

جدول ۱-۳ لیستی از متداول‌ترین کدهای وضعیت را نشان می‌دهد که توضیحات کدهای وضعیت فعلی HTTP خارج از محدوده این کتاب می‌باشد.

سرآیند[۱]

سرآیندها و متدها با هم کار می‌کنند، تا تعیین کنند که کلاینت‌ها و سرورها چه کاری انجام دهند. این بخش اهداف سرآیندها استاندارد HTTP را معرفی می‌کند. سرآیندهایی هستند که برای هر نوع پیام خاص و سرآیندهایی برای اهداف کلی می‌باشند. اطلاعاتی که در هر دو پیام درخواست و پاسخ ارائه می‌شود. سرآیندها به پنج کلاس زیر تقسیم می‌شوند:

[۱] Headers

- سرآیند عمومی
- سرآیند درخواست
- سرآیند پاسخ
- سرآیند موجودیت
- سرآیند توسعه

سرآیند درخواست[1]

هدر درخواست سرآیندی است که فقط در پیام درخواست قابل درک می‌باشد. آنها اطلاعاتی در مورد اینکه چه کسی یا چه درخواستی را ارسال می‌کند، جایی که درخواست ایجاد شده است، یا آنچه که ترجیحات و قابلیت مشتری است را ارائه می‌دهد. سرورها می‌تواند اطلاعاتی را که سرآیند درخواست در مورد مشتری می‌دهد، استفاده کند. تا سعی کند پاسخ بهتر را به مشتری دهد. جدول ۱-۴ سرآیندهای درخواست را نشان می‌دهد.

جدول ۱-۴ سرآیندهای اطلاعاتی درخواست

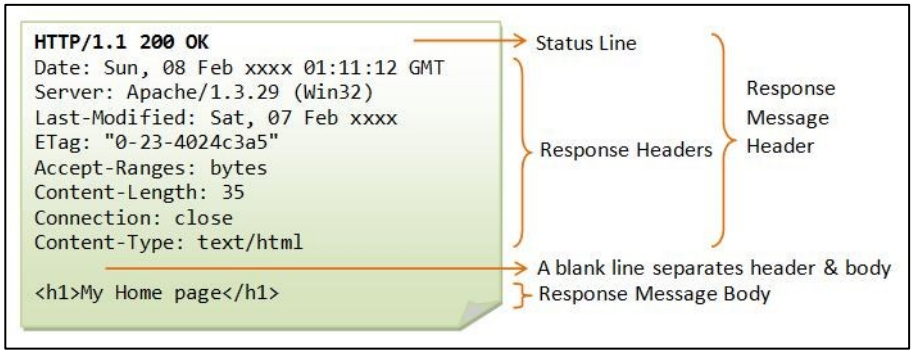
سرآیند	توضیح
HOST	نام میزبان و پورت سروری که درخواست را ارسال کرد، می‌دهد
User Agent	اطلاعاتی درباره مرورگر و سیستم عامل مشتری را می‌دهد
Accept	به سرور می‌گوید چه نوع رسانه‌ای برای ارسال مناسب است
Accept – Charset	به سرور می‌گوید کدام کارکترها برای ارسال مناسب است
Accept – Encoding	به سرور می‌گوید کدام کدینگ برای ارسال مناسب است
Accept – Language	به سرور می‌گوید که چه زبانی برای ارسال مناسب است
Authorization	شامل داده‌هایی است که مشتری برای تائید هویت خود به سرور می‌فرستد
Cookie	توسط مشتریان مورد استفاده قرار می‌گیرد تا یک توکن به سرور منتقل کند. هدر یک هشدار امنیتی درست نیست، اما پیامدهای امنیتی را دارد.

[1] Request Headers

سرآیند پاسخ[1]

پیام‌های پاسخ دارای مجموعه‌ای از هدرهای پاسخ هستند. هدرهای پاسخ برای مشتریان اطلاعات اضافی را فراهم می‌کنند. مانند اینکه چه پیامی را ارسال کرده، قابلیت پاسخ‌گویی را دارد، یا حتی دستورالعمل‌های ویژه‌ای در رابطه با پاسخ را ارائه می‌دهند. این سرآیندها برای پاسخگویی به مشتری کمک می‌کنند و درخواست‌های بهتری را در آینده ایجاد می‌کنند. هدرهای محتوا در سرآیند پاسخ اطلاعات خاصی در مورد محتوای موجودیت ارائه می‌دهند که نوع، اندازه و سایر اطلاعات مفید را برای پردازش می‌باشد.

جدول ۱-۵ لیست سرآیندهای پاسخ و شکل ۱-۱۰ شمای کلی از Response را نشان می‌دهد.



شکل ۱-۱۰ شمای کلی از Response

[1] Response Headers

جدول ۱-۵ سرآیندهای اطلاعاتی پاسخ

سرآیند	توضیح
Server	اطلاعاتی در مورد نام و نسخه نرم افزار سرور
Accept-Ranges	نوع محدوده‌هایی که یک سرور برای منبع می‌پذیرد
Set-Cookie	نشانه در سمت مشتری که سرور می‌تواند برای شناسایی مشتری استفاده کند.
Location	راهنمایی به مشتری برای انتقال به آدرسی دیگر که درخواستش کرده
Content-Encoding	کدگذاری داده‌های ارسالی
Content-Language	زبان محتویات صفحه
Content-Length	طول یا اندازه داده‌ها
Content-Type	نوع داده‌های ارسالی

حافظه‌های کش HTTP[۱]

پروتکل HTTP معمولاً برای سیستم‌های اطلاعاتی توزیع شده استفاده می‌شود، که عملکرد آن با استفاده از حافظه‌های کش پاسخ بهبود می‌یابد. پروتکل HTTP ۱,۱ شامل تعدادی از عناصر برای انجام کار ذخیره‌سازی است. هدف حافظه کش در HTTP۱,۱ رفع کردن بسیاری از درخواست‌هایی است که نیاز می‌باشد و همچنین برطرف کردن ارسال کامل پاسخ‌هایی در موارد بسیار که وجود دارد.

مکانیزم اصلی حافظه کش در HTTP۱,۱ شامل دستورالعمل‌هایی ضمنی برای حافظه ذخیره سازی می‌باشد که سرور در آن زمان انقضاء و اعتبارسنجی را تعیین می‌کند. که در این پروتکل از هدر کش کنترل[۲] برای این منظور استفاده می‌شود.

هدر Cache-Control اجازه می‌دهد که یک مشتری یا سرور، بتواند انواع دستورات را در هر درخواست یا پاسخ ارسال کند. دستورالعمل‌های کش در یک لیست جدا شده با کاما مشخص می‌شوند. جدول ۱-۶ دستورالعمل‌های مربوط به درخواست کش که می‌تواند یک مشتری در درخواست HTTP مورد استفاده قرار گیرد را نشان می‌دهد.

---

۱ HTTP - Caching

۲ Cache-Control

جدول ۱-۶ مثال‌هایی از درخواست کش توسط مشتری در درخواست HTTP

دستور درخواست کش	توضیح
no-cache	یک حافظه کش نباید از پاسخ برای پاسخگویی درخواست‌های بدون تائید موفقیت مجدد با سرور اصلی استفاده کند.
no-store	حافظه کش نباید هیچ چیزی در مورد درخواست مشتری یا پاسخ سرور را بدهد.
no-transform	موجودیت بدنه را تبدیل نمی‌کند.
only-if-cached	داده‌های جدید را بازیابی نمی‌کند. حافظه کش می‌تواند فقط سندی را ارسال کند که در کش وجود دارد و نباید با سرور اصلی ارتباط برقرار کند تا ببیند کپی جدید وجود دارد.

جدول ۱-۷ دستورالعمل‌های مربوط به پاسخ کش که می‌تواند توسط سرور در پاسخ HTTP مورد استفاده قرار گیرد را نشان می‌دهد.

جدول ۱-۷ مثال‌هایی از پاسخ کش توسط سرور در پاسخ HTTP

دستور درخواست کش	توضیح
public	نشان می‌دهد که پاسخ ممکن است توسط هر حافظه کش ذخیره شود.
private	نشان می‌دهد که تمام یا بخشی از پیام پاسخ برای یک کاربر تنها در نظر گرفته شده و نباید توسط یک حافظه مشترک کش ذخیره شود.
no-cache	یک حافظه کش نباید از پاسخ برای پاسخگویی درخواست‌های بدون تائید موفقیت مجدد با سرور اصلی استفاده کند.
must-revalidate	حافظه کش باید قبل از استفاده از اسناد دائمی، تائید شود و نباید از آنها استفاده شود.

امنیت HTTP[1]

مردم برای انجام امور جدی معاملات از وب استفاده می‌کنند. مردم بدون امنیت قوی، احساس راحتی نمی‌کنند که بعنوان مثال؛ خرید آنلاین و انجام امور بانکی را می‌توان

[1] HTTP – Security

نام برد. شرکت‌ها بدون اینکه قادر به محدود کردن دسترسی‌ها باشند، نمی‌توانند اسناد مهم را در سرورهای وب قرار دهند. وب نیاز به فرم امن از HTTP دارد. یک نسخه امن از HTTP نیاز به کارآمدی بالا، قابلیت حمل، آسان برای مدیریت و سازگار با جهانی که در حال تغییر است، می‌باشد. همچنین این پروتکل باید نیازهای اجتماعی و دولتی را تامین کند. برای تکنولوژی HTTP نیاز داریم که:

- تایید اعتبار سرور (مشتریان می‌دانند که با سرور واقعی در ارتباط هستند و فریب نمی‌خورند).
- تائید هویت مشتری (سرورها می‌دانند که آنها با کاربر واقعی در ارتباط هستند و جعلی نیست).
- یکپارچگی (مشتریان و سرورها مطمئن‌اند اطلاعات خود را تغییر می‌دهند).
- رمزگذاری (مشتریان و سرورها به صورت خصوصی بدون ترس و استراق سمع در ارتباط هستند).
- کارآیی (یک الگوریتم به مقرون به صرفه برای مشتریان و سرورها مورد استفاده قرار می‌گیرد)
- پروتکل‌ها تقریباً توسط تمام مشتریان و سرورها پشتیبانی می‌شود.
- سازگاری (پشتیبانی از بهترین روش‌هاای شناخته شده در روز).
- مقیاس پذیری مدیریتی (ارتباط امن برای هر کسی و هرجا).