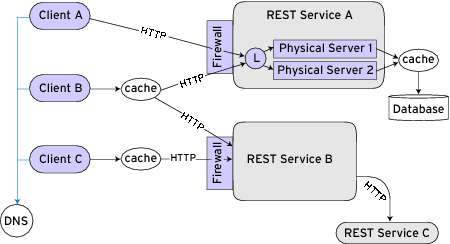
**برنامه‌ نویسی بر اساس اصول REST (رست) چیست؟**

**کلیات REST (رست)**

یکی از راه‌های ارائه خدمات و همکاری میان سیستم‌های کامپیوتری در بستر شبکه سرویس‌های وبREST (Representational state transfer) می‌باشد. در واقع REST یک معماری وب سرویس است که برای انتقال داده‌ها و اطلاعات، معمولا از [پروتکل HTTP](http://batisapp.ir/?p=373) استفاده می‌نماید. باید گفت که معماری REST (رست) یک اساس و رکن اصلی در ساختار امروزی وب به شمار می‌رود و در ابررسانه‌های توزیع شده کاربرد دارد. این معماری دارای رویکردی جهت ساده‌سازی سرویس‌های پیچیده وب است و هدف آن نیز توسعه مستقل از اجزا با استفاده از واسطه‌ها می‌باشد؛ این نوع از معماری جهت افزایش انعطاف‌پذیری، سادگی کارها، امکان مشاهده و نظارت و قابلیت حمل و اطمینان است. حیرت‌آور به نظر می‌آید که در دنیای وب، مرورگر کاربران از طریق راه‌های پیچیده با سرویس‌دهندگان ارتباط برقرار نماید، بدون آن‌که کاربران هیچگونه اطلاعی نسبت به منابع سرویس‌دهنده داشته باشند. در این ارتباط یکی از محدودیت‌های اصلی آن است که سرویس‌دهنده و سرویس‌گیرنده هر دو باید رسانه مشترکی را بپذیرند و از آن استفاده نمایند، بطور مثال می‌توان به زبان نشانه‌گذاری HTML اشاره نمود.

[](http://batisapp.ir/wp-content/uploads/2017/03/rest.gif)rest

متاسفانه در بسیاری از منابع، شما مفاهیم SOAP و REST را به عنوان رقیب مشاهده می‌نمایید، این در حالی است که SOAP یک پروتکل بوده و مقایسه آن با REST که یک معماری تلقی می‌شود از نظر تئوریک صحیح نمی‌باشد، و شاید بتوان برای مقایسه از SOA و RPC یاد کرد. اما بطور کلی اگر بخواهیم مقایسه‌ای داشته باشیم باید گفت که در این معماری برای انجام اتصال‌ها و دریافت ، ایجاد، بروزرسانی و یا حذف اطلاعات (CRUD) یک منبع، از فراخوانی یک آدرس تحت پروتکل HTTP استفاده می‌شود که به آن اصلاحا HTTP Request و پاسخ سرویس‌دهنده نیز معمولا در قالب زبان‌های نشانه‌گذاری همچون XML، HTML و یا JSON  بصورت یک HTTP Response ارسال می‌گردد. لازم به ذکر است که در یک سیستم که بر پایه معماری REST فعالیت می‌کنید، تمامی مفاهیمی که با آن سر و کار داریم به عنوان یک منبع در نظر گرفته می‌شود و از طریق یک URL قابل دسترس است؛ باید گفت که آدرس URL صرفا یک آدرس فیزیکی نبوده و در اکثر اوقات آدرسی منطقی است. به طور مثال اگر بخواهیم یک کتاب با شناسه ۱۰ را درخواست کنیم، کافی‌ است تا آدرس منطقی http://BatisApp/api/books/10 را فراخوانی نماییم.

**مختصری از تاریخچه REST**

REST در سال ۲۰۰۰ به بصورت پایان‌نامه دوره دکتری آقای Roy Fielding تحت عنوان Architectural Styles and the Design of Network based Software Architectures از دانشگاه Irvine کالیفرنیا معرفی گردید. آقای Fielding این سبک معماری را بصورت موازی با توسعه HTTP 1.1 و بر پایه نسخه موجود HTTP 1.0 طراحی نمود. لازم به ذکر است که ایشان از مولفان اصلی پروتکل HTTP نیز بوده‌اند.

**خصوصیات معماری REST**

* **سیستم شبکه‌ای دارای معماری Client – Server**
* **Stateless یا بدون وضعیت بودن:** به این معنا است که سرویس‌دهنده هیچ گونه اطلاعاتی را از وضعیت سرویس گیرنده در خود ذخیره ننماید. یعنی تمامی درخواست‌ها بایستی مستقل از یکدیگر باشند؛ در صورتی که نیاز به حالت قبلی باشد، سرویس گیرنده باید آن را در قالب درخواست جاری مجددا ارسال نماید.
* **Cacheable یا قابلیت Cache:** به این معناست که پاسخ‌های ارسالی از طرف سرویس‌دهنده باید قابلیت این را داشته باشند که در مرورگر سرویس‌گیرنده Cache شود؛ این قابلیت و زمان اعتبار معمولا در سرآیند پاسخ HTTP بصورت صریح و یا ضمنی ذکر می‌شوند و نیازی به ساز و کار جدید ندارد.
* **Layered System یا سیستم لایه‌بندی شده:** بدین معنا است که سیستم‌هایی که در بین راه سرویس‌گیرنده و سرویس‌دهنده قرار دارند بتوانند بصورت شفاف عمل نمایند؛ بطور مثال ممکن است زمانی که یک سرویس‌گیرنده درخواست منبعی را می‌نماید، منبع از طریق یک سرویس‌دهنده میانی که قبلا آن منبع را Cache نموده برای سرویس‌گیرنده ارسال شود. شاید جا داشته باشد که در اینجا بگوییم اگر قابلیت Stateless بودن وجود نداشت، اجرا مثال فوق بسیار دشوار بود.
* **Code on demand یا قابلیت کد درصورت نیاز:** این قابلیت اجباری نبوده، و این امکان را می‌دهد تا سرویس‌دهنده با ارسال کدی به سیستم سرویس‌گیرنده، دسترسی‌های آن را کاربر را سفارشی نماید.
* **دارای واسط یکنواخت یا Uniform Interface باشد.**

**به سیستم‌هایی که دارای قابلیت‌های بالا هستند و از REST استفاده می‌نمایند به اصطلاح RESTful گفته می‌شود.**

گفته‌ها حاکی از آن است که استاندارد جامع و مرجعی همچون استانداردهای قرار داده شده برای SOAP برای REST وجود ندارد و دلیل آن هم این است که REST یک معماری بوده اما SOAP یک پروتکل می‌باشد؛ اما باید گفت که برای پیاده‌سازی REST از استانداردهایی همچون HTTP، JSON، XML و … استفاده می‌شود.

**مقدمه**

در واقع REST یک پروتکل مشخص نیست،‌ بلکه یک سبک معماری است که کاربرد آن در معماری وب باعث شهرت اخیر آن شده است. این سبک در سال ۲۰۰۰ توسط Roy Feilding در پایان‌نامه‌ی دکترایش معرفی شد، اما تا سال‌ها استفاده از رواج نداشت. وب‌سرویس‌هایی که مطابق با این سبک معماری طراحی می‌شوند اصطلاحاً RESTful نامیده می‌شوند.

شاخص رعایت سبک معماری REST قواعدی است که روی مولفه‌ها، داده‌ها و نحوه‌ی تعامل آن‌ها قرار داده می‌شود. قواعد رسمی REST را می‌توان در دسته‌بندی زیر خلاصه کرد:

1. سرور و کلاینت

یک رابط[۱] یکپارچه میان کلاینت‌‌ها و سرور وجود دارد. این جدایی باعث می‌شود که مثلاً کلاینت‌ها دغدغه‌ی نگه‌داری داده‌ها را نداشته باشند و این مسئله کاملاً در حوزه‌ی سرور باقی می‌ماند. از طرف دیگر سرور هم هیچ دغدغه‌ی در مورد حالت[۲] کلاینت نباید داشته باشد. بدین ترتیب هم کلاینت‌ها ترابرپذیرتر[۳] می‌شوند و هم سرورها ساده‌تر و مقیاس‌پذیر تر می‌شوند. سرورها و کلاینت‌ها می‌توانند بازنویسی شوند و یا به صورت مستقل توسعه داده شوند به شرط اینکه رابط میان آن‌ها تغییری نکند (.

1. بدون حالت

سرورهای REST باید بدون حالت[۴] باشند، به عبارت دیگر با هر بار فراخوانی سرویس، تمام اطلاعات لازم برای اجرای آن سرویس باید فرستاده شود و ایجاد رویه‌های فراخوانی چندمرحله‌ای (که با نگه‌داری حالت کلاینت همراه هستند) مجاز نیست. البته این امکان وجود دارد که حالت کلاینت توسط سرویس نگه‌داری نشود و به مولفه‌ای پایا همچون پایگاه داده منتقل شود.

1. رابط یکپارچه

یکپارچه بودن رابط‌ها یکی از قواعد اساسی هر سرویس طراحی شده با سبک REST است. یکپارچه بودن سادگی و Decoupling در معماری ایجاد می‌کند. این قاعده شامل موارد زیر است:

1. شناسایی منابع: منابع به صورت مستقل قابل آدرس‌دهی باشند. منابع می‌توانند نمایش‌های مختلفی داشته باشند که از طریق سرویس قابل دسترسی هستند.
2. امکان تغییر منابع: کلاینتی که آدرس یک منبع و یک نمایش از آن را دارد، سرویس‌هایی برای تغییر یا پاک کردن آن منبع را دارد. البته ممکن است دسترسی لازم را نداشته باشد.
3. خود توصیف‌گری پاسخ‌ها: پاسخ سرویس‌ها باید اطلاعات کافی جهت پردازش پاسخ را دارا باشد، مثلاً قالب پاسخ که می‌تواند چیزی شبیه MIME type باشد و مشخصات Caching پاسخ.
4. تعیین وضعیت کلاینت با Hypermedia‌[۵]: کلاینت‌ها باید وضعیت خود را توسط Hyperlink های موجود در پاسخ‌های فعلی تغییر دهند. کلاینت نمی‌تواند فرض کند که Hyperlink ای به جز آن‌چه در پاسخ‌های دیگر از سرور گرفته مجاز هستند. طبعیتاً نقطه‌ی ورود نرم‌افزار از این قاعده مستثنا هستند.
5. قابل Cache بودن

مانند شبکه جهانی وب، کلاینت‌ها می‌توانند بدون نگرانی، پاسخ سرور را با توجه به مشخصات پاسخ، Cache کنند. بنابراین پاسخ سرویس‌ها باید به صورت صریح یا ضمنی ذاتاً قابل Cache بودن یا نبودن پاسخ و زمان آخرین تغییر آن را بیان کند و مکانیزمی برای بررسی تازه بودن پاسخ‌ها ارائه کند. مناسب بودن روش Cache می‌تواند بعضی تعاملات کلاینت و سرور را کاهش دهد و مقیاس‌پذیری و کارایی را بهبود بدهد.

1. لایه‌ای بودن

یک کلاینت نمی‌تواند در حالت عادی تشخیص دهد که به صورت مستقیم به سرور نهایی متصل است یا به سرورهای میانی متصل شده است. سرورهای میانی می‌تواند با استفاده از Load-balancing و مکانیزم‌های Caching مقیاس‌پذیری را بهبود دهند. همچنین می‌توانند رویه‌های امنیت روی دسترسی‌ها اعمال کنند.

1. ارسال کد به کلاینت

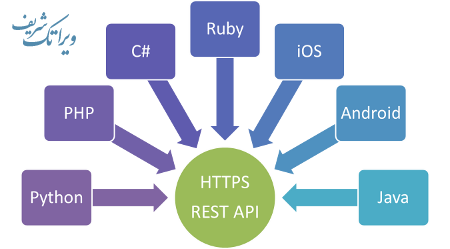
سرورها می‌توانند با ارسال کد اجرایی سمت کلاینت به آن‌ها ویژگی‌ها و قابلیت‌های آن را افزایش بدهند. به عنوان نمونه سرور می‌تواند کد جاوا اسکریپت برای کلاینت بفرستد که در مرورگر اجرا می‌شود. البته این مورد اجباری نیست و در بعضی کاربردها هم اصلا وجود ندارد.

**کاربرد REST در طراحی وب‌سرویس‌ها**

مهم‌ترین کاربرد سبک معماری REST در طراحی وب‌سرویس‌های مبتنی بر HTTP بوده است. این کابرد خود الگوی رایج مشخصی دارد که در ادامه شرح می‌دهیم.

* در این الگو آدرس همه‌ی وب سرویس‌ها با URL پایه‌ای همچون [http://viratech.ir/resources/](http://example.com/resources/) شروع می‌شود.
* قالب انتقال می‌تواند یکی از قالب‌های استاندارد Internet media type باشد، که معمولاً JSON است اما انواع دیگری همچون XML، Atom و YAML نیز می‌تواند باشد.
* از توابع استاندارد HTTP شامل POST، GET، DELETE، PUT و الگوی مشخص در URL برای طراحی سرویس‌ها به صورت یکپارچه استفاده می‌شود.
* Hypertext برای تغییر وضعیت و اشاره به منابع مرتبط استفاده می‌شود.
* هر وب‌سرویس راه دسترسی یا تغییر یک «منبع» است.

سرویس‌های REST مستقل از کلاینت می‌توانند عمل کنند. حتی کلاینت می‌تواند سرورهای دیگری باشند که قصد تعامل با سرویس‌دهنده را دارند (شکل ۱).



شکل ۱: سرویس‌های REST مستقل از کلاینت طراحی می‌شوند. کلاینت‌ها می‌توانند کلاینت‌های وب، موبایل یا سرویس‌دهنده‌های دیگر باشند

به عنوان نمونه رابط یکپارچه روی منبع فرضی مشتریان به همراه مثال‌هایی از هر تابع رابط در جدول زیر آمده است.

