فصل سوم – آشنایی با معماری REST-API

REST API یک نسخه جدید و قابل انعطاف از سرویس های وب ارائه می کند. در این قسمت با مفاهیم مهم در این معماری و عملکرد API و مشتریان آن و نحوه تبادل اطلاعات بین این دو را شرح می دهیم.

امروزه سرویس های وب یکی از موضوعات مهم در حوزه وب و برنامه نویسی می باشد. که هر برنامه نویس مبتنی بر وب باید با نحوه پیاده سازی و استفاده از آنها آشنایی داشته باشد. وب سرویس ها امکان ارتباط بین برنامه ها را در بستر وب فراهم می کنند.

با کمک وب سرویس ها، نرم افزارهای پیاده سازی شده با زبان های برنامه نویسی مختلف و اجرا شده در سکوهای مختلف می توانند از طریق وب، خدمات موجود در نرم افزارهای دیگر را درخواست کرده و استفاده کنند.

۳-۱ تعریف REST API

معماری REST API یا RESTful API یک روش جدید و انعطاف پذیر از سرویس های وب در اختیار ما قرار می دهد که اخیراً در بیشتر جاها از جمله در ساخت اپلیکیشن های موبایل جای وب سرویس های سنتی را گرفته است.

به طور مشخص دو معماری برای ساخت و استفاده از سرویس های وب وجد دارد:

۱. وب سرویس های مبتنی بر پروتکل SOAP که از XML برای توصیف خود و فرمت داده های انتقالی استفاده می کنند. برای آشنایی با این نوع وب سرویس ها از بخش قبلی "وب سرویس های مبتنی بر SOAP" استفاده کنید.

۲. وب سرویس های مبتنی بر REST که فرمت خاصی برای توصیف یا انتقال داده ها تعیین نمی کنند هر چند بیشتر از JSON برای این کار استفاده می شود.

برای ساخت و استفاده از وب سرویس های مبتنی بر REST از معماری REST API استفاده می شود که مجموعه ای از رهنمودها و دستورالعمل ها برای ایجاد وب سرویس و ارتباط با آن می باشد. در ادامه به شرح این معماری می پردازیم.

REST مخفف واژگان Representational State Transfer است که از سال ۲۰۰۵ در وب شناخته شد. اگر خیلی ساده بخواهیم به این موضوع نگاه کنیم، REST عبارت است از راهکارها و روش‌هایی که با استفاده از آن‌ها می‌توان به رد و بدل داده ها از طریق شبکه پرداخت. به عبارت دیگر، REST راهی ساده به منظور سازماندهی تعاملات بین سیستم‌های جدا از یکدیگر می‌باشد.

در مقابل REST ، پروتکل SOAP که مخفف واژگان Simple Object Access Protocol است قرار دارد که از طریق آن می‌توان به رد و بدل دیتا از طریق شبکه در قالب وب سرویس های مختلفی با فرمت XML پرداخت.

APIهم مخفف واژگان Application Programming Interface است که شامل متدهایی برای ارتباط با سایر کتابخانه ها یا اپلیکیشن ها می باشد.
حال اگر این اصطلاحات در کنار یکدیگر قرار دهیم RESTful API ساخته می شود که منظور ساز وکارهایی برای ارتباط یک سیستم با سرویس ها یا سیستم های دیگر با استفاده از معماری خاص است. معماری REST دارای یکسری ویژگی‌ها است که مهمترین آن‌ها عبارتند از:

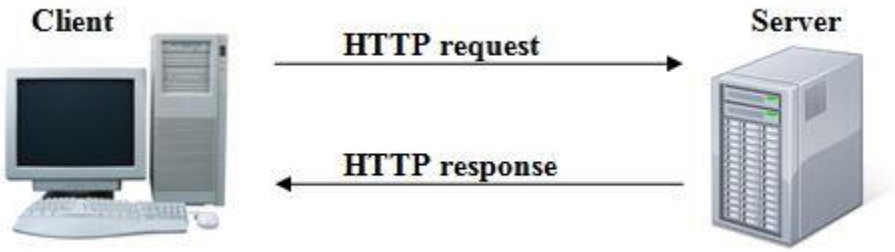
-ثبات و یکنواختی این معماری در همه جایAPI
-عدم برخورداری از session در سمت سرور
-به‌کارگیری از کدهای وضعیت HTTP
-استفاده از URL ها برای مشخص کردن مسیرهای مد نظر
-قراردادن کوئری ها در URL به جای قرادادن آنها در بخش header در ارتباط HTTP

بنابراین، RESTful API ها به کمک توسعه‌دهندگان وب آمده‌اند تا فرایند توسعه ی وب، ایجاد تجربه ی کاربری بهتر، سهولت در استفاده از API ها و نقل و انتقال داده‌ها از طریق پروتکل HTTP را امکان‌پذیر سازند.
علاوه بر این، توجه داشته باشیم که REST بیش از آنکه پروتکل باشد، یکسری راهنما، اصول و قواعدی است که با استفاده از آن‌ها می‌توانیم به برقراری ارتباط مابین منابع مختلف بپردازیم.
برای درک بهتر این موضوع، فرض کنید، قصد دارید از API شبکه ی اجتماعی توییتر برای نشان دادن محبوب‌ترین توییت ها در وب‌سایت خود استفاده کنید. در چنین شرایطی، بدون آن که به سرور این سایت دسترسی داشته باشید، می توانید با استفاده از API موجود روی سرور توئیتر، داده های مورد نیاز خود را بدست آورده و در سایت خود نمایش دهید . معمولاً توسعه دهندگان API داده ها را با قالب JSON ارائه کرده و دریافت می کنند. هر چند اجباری به این کار ندارند.
امروزه برنامه های سنتی وب در حال حرکت به سمت سرویسی شدن هستند، بدین صورت که کلاینت ها تنها از طریق وب سرویس ها با سرور در تماس هستند. در چنین برنامه هایی منطق برنامه کاملا در سمت کلاینت پیاده سازی می شود و سرور دیگر هیچ نقشی جز فراهم کردن داده برای کلاینت ها برعهده ندارد.
نمونه رایجی از این گونه نگرش برنامه نویسی، برنامه های SPA که مخفف Single Page Application است می باشد. در اینگونه برنامه ها تمامی منطق برنامه در سمت کلاینت پیاده سازی می شود و تنها برای دریافت داده ها نیاز به ارتباط با سرور دارد.
این نگرش این امکان را فراهم می کند که به راحتی و با کمترین هزینه یک برنامه SPA را به یک برنامه دیگر، مثلا یک برنامه موبایل تبدیل کنید. تنها کاری که باید انجام دهید این است که یک کلاینت برای تلفن همراه بنویسید که از طریق همان API ها با سرور ارتباط برقرار کند.

۲-۳ روش کار پروتکل HTTP

قبل از پرداختن به جزئیات REST API ، باید با روش کار پروتکل HTTP آشنا شویم. منظور از پروتکل یکسری اصول و قواعدی است که مشخص می کند در هر ارتباط چه پیام‌هایی قابل انتقال هستند و ترتیب مبادله پیام ها و فرمت پیام ها را

تعیین می کند. HTTP به پروتکلی اشاره دارد که از آن طریق آن می‌توان در بستر وب به نقل و انتقال داده ها پرداخت. در HTTP یک سرویس گیرنده و یک سرویس دهنده داریم.
منظور از کلاینت یا مشتری، یک مرورگر یا هر برنامه کاربردی دیگری است که از طریق وب، منبعی را از سرور درخواست می کند. منبع (Resource) می‌تواند هر شیئی مثل تصویر، ویدئو، صفحه وب باشد. و منظور از سرور نرم افزاری روی کامپیوتر سرویس دهنده است که همواره در حال اجرا و آماده است تا درخواست های رسیده از سمت سرویس گیرنده ها را دریافت، پردازش کرده و پاسخ را به آنها برگرداند. همیشه ارتباط با درخواست یک منبع از سمت سرویس گیرنده شروع می شود. و سپس پیام هایی بین سرویس گیرنده و سرویس دهنده رد و بدل می شود.



پیام‌ها در پروتکل HTTP دارای دو بخش بدنه(body) و سرآیند(header) هستند. بدنه شامل داده هایی است که قصد انتقال آن از مرورگر خود برای سرور را دارید (مثل نام کاربری و رمزعبور که وارد فرم لاگین می‌کنید). سرآیند هم شامل یکسری متاداده مثل نوع داده های ارسالی (Content-Type)، نحوه کدگذاری متن (Encoding)، متدهای HTTP و غیره است که تعیین می کنند که داده های قرار گرفته در بدنه پیام به شکل می‌بایست مورد استفاده و پردازش قرار گیرند.
در صورتی که از گوگل کروم استفاده می‌کنید، با فشردن هم‌زمان کلیدهای Ctrl + Shift + J می‌توانید پنجره ی Chrom Developer Toolsرا باز کرده وارد تب Network شوید و همان‌طور که به گشت زنی در وب می پردازید، می‌توانید جزئیات سرآیند و بدنه پروتکل HTTP را رصد کنید.

نوع محتوا content type

پیش از این گفتیم که هر پیام دارای دو بخش سرآیند و بدنه است که بخش بدنه پیام حاوی داده ها و بخش سرآیند حای تعاریف می باشدو با کمک عبارت Content-Type در سرآیند نوع داده های ارسالی تعیین می شود. به عنوان مثال برای ارسال داده ها از سرویس گیرنده به سرویس دهنده با فرمت json ، تعریف زیر در سرآیند قرار می گیرد:

```
Content-Type : application/json
```

و یا اگر پاسخ سرویس دهنده بصورت یک صفحه html است نوع داده آن بصورت زیر تعیین می شود:

```
Content-Type : text/html
```

همچنین اگر پاسخ یک فایل تصویر با فرمت png است، نوع محتوای پیام بصورت زیر خواهد بود:

```
Content-Type: image/png
```

ضمن اینکه بدنه پیام حاوی بایت های مربوط به تصویر بوده و سرویس گیرنده با توجه به بخش سرآیند و نوع محتوا تشخیص می دهد که بایت های دریافتی را باید به عنوان یک فایل تصویری مورد استفاده قرار دهد.

افعال HTTP

در اینجا باید با مفهومی تحت عنوان افعال یا متدهای HTTP آشنا می شوید. هر درخواست از جنس HTTP، حاوی متدی است که در بخش هدر قرار می‌گیرد. متدهای HTTP به سرویس دهنده می فهمانند که درخواست برای چه کاری انجام شده است. به عنوان مثال اگر تعریف زیر در بخش سرآیند قرار گیرد:

```
GET / HTTP/1.1
```

مشخص می کند که درخواست با متد GET برای گرفتن اطلاعات ارسال شده است. متدهای مهم مورد استفاده در درخواست ها عبارتند از:

GET– برای دریافت یک شیء

POST – برای ایجاد یک شیء

PUT -برای تغییردادن و جایگزین کردن یک شیء

DELETE -برای حذف یک شیء

متد GET ساده‌ترین و پرکاربردترین متد استفاده شده در پروتکل HTTP است. به عنوان مثال هنگام ورود به یک سایت، کلیک روی لینک های مختلف آن برای درخواست صفحات مختلف و دریافت ضمائم صفحه مثل تصاویر و غیره دائماً درخواست هایی با متد GET به سرویس دهنده فرستاده می شود.

اکثر وب سرویس ها فقط درخواست های با متد GET را می پذیرند، یعنی فقط درخواست برای گرفتن داده ها را پاسخ داده و درخواست ها برای ایجاد، تغییر یا حذف داده ها را قبول نمی کنند. بعضی وب سرویس ها درخواست ها با متدهای دیگر را فقط از سرویس گیرندگان معین و تنها پس از احراز هویت آنها ، قبول می کنند.

کدهای وضعیت

یکی تعریف دیگر که در سرآیند پاسخ های سرویس دهنده به سمت سرویس گیرنده قرار داده می شود. کد وضعیت پاسخ است که وضعیت پاسخ را مشخص می کند. اینکه درخواست با موفقیت پاسخ داده شده، چه عملی انجام شده یا چه مشکلی داشته است. تعریف زیر کد ۲۰۰ تعیین می کند که نشان دهنده اجرای موفق درخواست است:

```
HTTP/1.1 200
```

کدهای وضعیت اعداد صحیح ۳ رقمی هستند که با توجه به رقم سمت چپ به چند دسته تقسیم می شوند و در ادامه به مهمترین آنها اشاره می کنیم. برای دریافت لیست کامل کدهای وضعیت می توانید جستجوی ساده ای در اینترنت انجام دهید.

1xx - پاسخ موقت

این نوع کدها نشـان دهنده پاسـخی موقت و نیازمند انجام فعالیتی از طرف درخواسـت‌کننده برای ادامه می‌باشد. به عنوان مثال وضـعیت 102 نشـان می دهد که درخواسـت‌کننده باید به درخواسـت خود ادامه دهد. سرور زمانی این کد را نمایش می‌دهد که بخش اول درخواست را دریافت کرده و منتظر بقیه درخواست‌ها می‌باشد.

2xx - موفقیت آمیز

این کدهای وضعیت نشان دهنده موفقیت سرور در پردازش درخواست می‌باشد.

200	سرور با موفقیت درخواست را محاسبه کرده‌است. و این به این معنی می‌باشد که سرور صفحه درخواست شده را فراهم کرده‌است.
201	درخواست موفقیت آمیز بوده و سرور یک منبع جدید ایجاد کرده‌است.
202	سرور درخواست را پذیرفته، اما هنوز آن را محاسبه نکرده‌است.
203	سرور با موفقیت درخواست را محاسبه کرده، اما اطلاعاتی را نمایش می‌دهد که ممکن است مربوط به منبع دیگری باشند.
204	سرور با موفقیت درخواست را محاسبه کرده، اما هیچ محتوایی را نمایش نمی‌دهد.

4xx – خطای سرویس گیرنده

این نوع کدهای وضعیت نشان می‌دهند که احتمالاً خطایی در درخواست رخ داده که باعث شده سرویس دهنده نتواند به آن پاسخ دهد.

400	سرور قادر به تشخیص نحو (Syntax) درخواست نمی‌باشد.
401	درخواست نیازمند تصدیق می‌باشد. سرور ممکن است این پاسخ را برای لاگین یک صفحه نمایش دهد.
403	درخواست معتبر است، اما سرور قادر به انجام عملیات نیست. کاربر ممکن است مجوزهای لازم برای یک منبع را نداشته باشد یا ممکن است نیاز به حساب کاربری خاصی باشد.
404	سرور قادر به پیدا کردن صفحه درخواست شده نمی‌باشد. برای مثال اگر برای صفحه‌ای که در سرور وجود ندارد درخواست شود، سرور اغلب این کد را نمایش می‌دهد.

۳-۳ آشنایی قالب JSON

در حال حاضر JSON (یا JavaScript Object Notation) اصلی ترین قالب برای تبادل اطلاعات بین سرویس گیرنده و API در پروتکل REST می باشد و نیز اکثر زبان های برنامه نویسی از جمله جاوا، جاوااسکریپت و php ، دستوراتی برای تبدیل اطلاعات به فرمت JSON یا تفسیر اطلاعات با قالب JSON و تبدیل آن به اشیا را دارند.
JSON یک قالب قابل خواندن برای ساختار بندی ، ذخیره و تبادل داده ها می باشد که می تواند برای ذخیره کردن رکوردهای داده بصورت قابل خواندن و تمییز در یک فایل متنی یا به عنوان یک قالب برای ارسال یا دریافت داده های بین دو سیستم استفاده شود. JSON برای انسان ها به راحتی قابل خواندن و نوشتن بوده و تفسیر و تولید آن برای ماشین ها هم ساده است.
JSON دارای شکل ساده و قابل فهم است و با مطالعه چند مثال زیر به راحتی با آن آشنا شده و می توانید از آن استفاده کنید.
مثال ۱ : برای نگهداری یک داده به فرمت JSON می نویسیم:

```
{"name":"ali" }
```

نمونه مثال بالا مشخص میکند که داده "john" یک نام است.
مثال ۲: برای نگهداری اطلاعات یک شیء به فرمت JSON می نویسیم:

```
{"name":"ali", "age":20, "city":"tehran"}
```

مثال ۳: برای نگهداری یک آرایه از اشیا بصورت زیر از فرمت **JSON** می نویسیم:

```
[{ "name":"ali" , "age":20, "city":"tehran"},
 { "name":"sara", "age":17, "city":"yazd"},
 { "name":"mina", "age":21, "city":"birjand"} ]
```

همچنین هر کدام از ویژگی های شی **JSON** می توانند خود حاوی یک شیء دیگر یا آرایه ای از اشیا باشند. به مثال زیر دقت کنید که یک شیء دارای ویژگی **count** و **data** داریم که ویژگی **data** خود یک آرایه از اشیا می باشد:

```
{"count":"3",
 "data":[{ "name":"ali" , "age":20, "city":"tehran"},
         { "name":"sara", "age":17, "city":"yazd"},
         { "name":"mina", "age":21, "city":"birjand"} ] }
```

۳-٤ نحوه کار REST API

همان گونه که پیش تر گفته شد، REST API یک مجموعه از رهنمودها برای ایجاد و ارائه خدمات در وب و درخواست و استفاده از آنها در بستر وب و پروتکل HTTP می باشد.
در نگاه جزئی تر می توان یک مجموعه از منابع مثل فایل ها و بانک اطلاعاتی روی سرویس دهنده را در نظر گرفت و کنار آن ها برنامه ای دارای تعدادی تابع پیاده سازی شده با یک زبان برنامه نویسی که به این منابع دسترسی دارند. به این توابع API های تحت وب می گویند چرا که از طریق وب قابل فراخوانی هستند.
برنامه های سرویس گیرنده یا همان مشتریان می توانند با هر زبان برنامه نویسی پیاده سازی شده یا روی هر سخت افزار و سیستم عامل اجرا شده و درخواست هایی برای اجرای هر کدام از توابع به سرور ارسال کنند. این درخواست ها منجر به فراخوانی یکی از توابع شده می تواند برای دریافت یک یا چند منبع، ایجاد ، حذف یا بروزرسانی منابع باشد.
همه درخواست ها (یا فراخوانی‌ها) بدون حالت (state-less) هستند و نمی‌توان هیچ چیزی را بین درخواست های مختلف حفظ کرد.
هر درخواست دارای سرآیند و بدنه است. در سرآیند با توجه به نوع درخواست یکی از متدهای GET ، PUT ، POST یا DELETE تعیین شده است همچنین نوع محتوای ارسالی به یکی از شکل های زیر تعیین می شود:

نوع محتوا	مقدار Content-type
داده ها با فرمت json	application/json
داده ها با فرمت xml	application/xml
مجموعه ای از جفت متغیر و مقدار	application/x-www-form-urlencoded
چند بخشی	multipart/form-data

بدنه تابع شامل داده ها یا اشیای مورد نیاز برای اجرای تابع می باشد. به عنوان مثال اگر در هدر درخواست ، داشته باشیم :

```
Content-Type: application/x-www-form-urlencoded
```

بدنه درخواست باید دارای تعدادی متغیر مساوی مقدار که با علامت & از هم جدا شده اند مثل زیر باشد:

```
name=ali&age=21&city=birjand
```

یا در صورتی که نوع محتوا در هدر درخواست بصورت زیر باشد:

```
Content-Type:application/json
```

بدنه درخواست باید متنی با فرمت جی سان مثل زیر باشد:

```
{"name":"ali", "age":20, "city":"tehran"}
```

در هر کدام از درخواست ها باید نام تابع مورد نظر، مشخص شود. نام تابع می تواند در سرآیند یا بدنه پیام تعیین شود. هر چند توصیه شده که نام تابع در ادامه URL نوشته شود. به عنوان مثال URL زیر برای فراخوانی تابع addPerson از API به آدرس http://www.codenevisan.com/api/v1 درخواست گردد:

```
http://www.codenevisan.com/api/v1/addPerson
```

سرویس دهنده پس از دریافت درخواست، تابع درخواستی را فراخوانی کرده و داده های دریافتی را تحویل آن می دهد. این تابع می تواند در صورت نیاز با اتصال به بانک اطلاعاتی یا دسترسی به دیگر منابع موجود روی سرور، درخواست را انجام داده و نتایج را تولید نماید.

نتایج می تواند یک پیغام موفقیت یا خطا یا نوع داده پیچیده تر مثل یک یا چند رکورد داده باشد که به سمت سرویس گیرنده گسیل خواهند شد. برای ارسال داده ای پیچیده می توان از قالب JSON یا هر قالب دیگری استفاده کرد.

همان گونه که قبلاً بیان شد، در سمت سرویس گیرنده نیز پاسخ دارای دو بخش سرآیند و بدنه است. در بخش سرآیند باید کد وضعیت و نوع محتوای ارسالی تعیین شده و داده ها با فرمت مناسب در بدنه پیام قرار گیرند. به عنوان برای ارسال یک رکورد، می تواند از قالب JSON استفاده کند. سرآیند پیام بصورت زیر خواهد بود:

```
HTTP/1.1 200
Content-Type : application/json
```

همچنین رشته زیر را در خروجی چاپ خواهد کرد تا به سمت سرویس گیرنده فرستاده شود:

```
{"name":"ali", "age":20, "city":"tehran"}
```

در مثال فوق، سرویس گیرنده با دریافت پاسخ ابتدا هدر را بررسی کرده و از اجرای موفق درخواست خود اطمینان پیدا می کند و تشخیص می دهد که بدنه پاسخ دارای داده هایی با فرمت جی سان است، داده ها را از بدنه برداشته و پس از تفسیر آن رکورد را بدست آورده و استفاده می کند.

در این قسمت مفاهیم و اصطلاحات مهم در معماری REST-API ، نحوه کار سرویس گیرنده و سرویس دهنده و شکل پیام های مبادله شده بین آنها را در معماری REST –API شرح دادیم. در بخش بعد یک API و کلاینت ساده با زمان PHP پیاده سازی کرده و این فرایندها را در عمل استفاده خواهیم کرد.