Json API

JSON که کوتاه شده عبارت "JavaScript Object Notation" است، یک استاندارد سبک مبتنی بر متن است که برای تبادل داده های قابل خواندن توسط انسان ایجاد شده است. قوانین استفاده شده توسط JSON برای برنامه نویسان شناخته شده هستند.

قالب اصلی JSON توسط Douglas Crockford مشخص شده است.

برای تبادل داده های قابل خواندن توسط انسان ایجاد شده است.

در زبان برنامه نویسی JavaScript توسعه داده شده است.

کابرد های json

فرمت JSON برای serialize کردن و انتقال داده های ساخت یافته از طریق شبکه استفاده می شود.

در درجه اول به منظور انتقال داده ها بین یک سرور و یک برنامه وب مورد استفاده قرار میگیرد.

وب سرویس ها و API ها از قالب JSON برای ارائه داده استفاده می کنند.

در برنامه های دسکتاپ می توان از آن برای ذخیره سازی تنظیمات و پیکربندی برنامه استفاده کرد.

ویژگی های json

خواندن و نوشتن JSON آسان است و یک فرمت سبک و مبتنی بر متن برای تبادل داده ها است و JSON مستقل از زبان است.

Authentication

Authentication یا احراز هویت به فرآیندی گفته می شود که در آن ارسال کننده یا دریافت کننده اطلاعات برای همدیگر اطلاعاتی را ارائه می کنند تا مطمئن شوند آنها همانی هستند که ادعا می کنند. به نوعی در احراز هویت بررسی می شود که شخص یا ... همانی هست که ادعا می کند. اگر ارسال کننده یا دریافت کننده اطلاعات نتوانند به درستی برای همدیگر احراز هویت شوند در این میان اعتمادی ایجاد نمی شود که آنها بتوانند با همدیگر تبادل اطلاعات داشته باشند. احراز هویت یا Authentication همانطور که گفتیم یک فرآیند است و این فرآیند هم می تواند بسیار بسیار ساده باشد و هم می تواند بسیار بسیار پیچیده و دشوار باشد.

ساده ترین راهکار احراز هویت که همگی ما از آن استفاده کرده ایم ساختار بسیار ساده یک کلید احراز هویت متنی است که ما آن را به عنوان پسورد یا رمز عبور می شناسیم و برای احراز هویت شدن در سیستم های مختلف از آن استفاده می کنیم. اما احراز هویت به تنهایی شامل فاکتورهای مختلفی است که برای بالا بردن سطح امنیتی آن ما این فاکتورها را به شکل زیر طبقه بندی کرده ایم ، هر چقدر شما از فاکتورهای بیشتری در احراز هویت استفاده کنید طبیعتا امنیت شما نیز بالاتر خواهد رفت :

فاکتور اول: چیزی که شما می دانید (What you know)

ساده ترین و البته ضعیف ترین فاکتور احراز هویت یا Authentication در امنیت اطلاعات چیزی است که شما می دانید . ساده ترین مثال آن را نیز همین رمز عبور یا پسوردی می توانیم تعبیر کنیم که از آن در طی روز ممکن است بارها استفاده کنیم. مهمترین نکته در

خصوص این فاکتور احراز هویت این است که اگر چیزی که شما می دانید به عنوان روش احراز هویت استفاده می شود ، بنابراین اگر این چیز را شخص دیگری هم بداند ، آن شخص می تواند به جای شما احراز هویت شود.

فاکتور دوم: چیزی که شما دارید (**What you have**)

چیزی که شما دارید یا **What you have** به این معناست که شما دارای یک دستگاه فیزیکی هستید که این دستگاه شما را احراز هویت می کند و شما نیازی به حفظ کردن چیزی ندارید و صرفا با داشتن این دستگاه احراز هویت خواهید شد ، از نمونه های این دستگاه ها می توانیم به کارت های هوشمند ، توکن های امنیتی و ... اشاره کنیم.

منطقی که برای فاکتور دوم احراز هویت وجود دارد این است که اگر شما یک کارت هوشمند در کنار خود دارید بنابراین مالک آن نیز هستید و مالک دسترسی های مربوط به آن نیز خواهید بود. تصور کنید برای ورود و خروج به یک اتاق سرور یا یک محوطه امنیتی شما باید یک کارت را در دستگاه مربوطه وارد کنید و فقط کارتی می تواند وارد این محیط شود که متعلق به سجاد عیدی است ، حالا اگر هر کسی این کارت را در اختیار داشته باشد می تواند سجاد عیدی باشد.

مشکل اصلی در اینجاست که سجاد عیدی ممکن است کارت مورد نظر را گم کند و یا کارت از او به سرقت رفته باشد و یا در شرایط بدتر خود سجاد عیدی کارت را به کسی داده باشد.

فاکتور سوم : چیزی که شما هستید (**What you are**)

کاربران ممکن است رمز عبور خود را فراموش کنند ، کاربران ممکن است کارت هوشمند خود را گم کنند اما قطعا فراموش نمی کنند که دست و بدن خود را همراه خود به اینور و آنور ببرند. در فاکتور سوم احراز هویت از اعضا و پارامترهای فیزیکی بدن انسان برای احراز هویت استفاده می شود که برای هر فردی در دنیا منحصر به فرد است . برای مثال در این نوع از احراز هویت از الگوی اثر انگشت ، الگوی صدای شخص ، الگوی مردمک و عنبیه چشم ، الگوی کف دست ، الگوی **DNA** و ... استفاده می شود.

مشکلاتی که در استفاده از این روش احراز هویت وجود دارد این است که ممکن است ما مواردی **false positive** و مواردی **false negative** داشته باشیم که به معنی وجود اشتباهات در سیستم احراز هویت است ، البته این مشکل بستگی به مکانیزمی دارد که شما استفاده می کنید ، من یک نرم افزار ساده احراز هویت توسط چهره داشتم که بعد از نصب بر روی سیستم و نگاه کردن به آن به سیستم وارد می شدم ، هیچ کس نمی توانست به غیر از من توسط این نرم افزار تشخیص داده شود تا اینکه خواهرزاده ۴ ساله من به نگاه کردن به وبکم موفق به ورود به سیستم شد !!!

یکی دیگر از مشکلاتی که برای پیاده سازی احراز هویت با فاکتور سوم وجود دارد این است که هزینه پیاده سازی این نوع مکانیزم بسیار بسیار بیشتر از هزینه های سایر سیستم های احراز هویتی در سطح کلان می باشد.

فاکتور چهارم: کاری که شما می کنید (**What you do**)

کاری که شما می کنید یا **What you do** در واقع یک نوع احراز هویت بیومتریک است که جزو زیرمجموعه های فاکتور سوم به حساب می آید ، در این روش احراز هویت رفتارهایی که شما انجام می دهید تجزیه و تحلیل می شود و بر حسب آنها تشخیص داده می شود که شخص مورد نظر همانی است که ادعا می کند یا خیر ، یکی از مرسوم ترین روش هایی که در این خصوص استفاده می شود سرعت تایپ کردن پسورد شما است ، فرض کنید شما پسورد خود را در حالت عادی در عرض ۱۵ ثانیه وارد می کنید و سیستم در صورتیکه شما در

بازه زمانی ۱۵ تا ۲۰ ثانیه پسورد خود را وارد کنید تشخیص می دهد که خود شما هستید ، حالا فرض کنید که سیستم تشخیص می دهد که شما پسورد را درست وارد کرده اید اما بر حسب عادتی که داشته اید نبوده است و پسورد حدود ۴۰ تا ۵۰ ثانیه طول کشیده است تا وارد شود ، حالا با اینکه پسورد درست است اما سیستم احراز هویت تشخیص می دهد که پسورد شما به سرقت رفته است و شخصی در حال خواندن و تایپ کردن پسورد است و به شخص مورد نظر اجازه **Login** نخواهد داد. البته احراز هویت به این روش چندان هم دقیق نیست و به همین دلیل بیشترین استفاده از این نوع مکانیزم های احراز هویتی در لابراتوارها است و در محیط واقعی چندان کاربردی ندارند.

فاکتور پنجم: احراز هویت چند فاکتوری (**Multifactor Authentication**)

همانطور که از نامش هم پیداست یعنی از چندین فاکتور احراز هویت در احراز هویت افراد استفاده شود. برای مثال شما کارت بانکی که دارید دارای سیستم احراز هویت **Two Factor Authentication** یا احراز هویت دو فاکتوره است چون هم از شما رمز عبور پرسیده می شود و هم اینکه باید کارت بانکی را داشته باشید. یا اینکه شما کارت شناسایی دارید که بعد از وارد کردن کارت شناسایی در دستگاه مربوطه باید اثر انگشت شما نیز تایید شود و ایندو با هم تشکیل یک سیستم احراز هویت را می دهد.

اگر هر سه فاکتور با هم در یک سیستم احراز هویت استفاده شود باعث بالا رفتن حداکثری امنیت می شود اما این نکته را فراموش نکنیم که هر چقدر امنیت شما بالا برود بالطبع آن دسترسی پذیری و چه بسا عملیاتی بودن یا **Functionality** سیستم شما پایین می آید و امنیت دیگر به عنوان یک فاکتور مزاحم در نظر گرفته می شود تا یک فاکتور آرامش بخش ، یک مثال ساده را می زنم ، آنتی ویروس شما اگر زیادی به ویروس ها و فعالیت های سیستم شما گیر بدهد طبیعتا آن را خاموش می کنید چون دسترسی پذیری و عملکرد سیستم شما را دچار اختلال کرده است ، همین مورد برای **UAC** ویندوز هم صادق است.

Soap

مخفف **Simple Object Access Protocol** و یک پروتکل مبتنی بر **XML** برای رد و بدل کردن اطلاعات بین برنامه ها است. اطلاعات در **SOAP** به صورت پیام (**Message**) و از طریق پروتکل‌های موجود در اینترنت مانند **HTTP** منتقل می‌شود (**SOAP** در سایر پروتکل ها، مانند **SMTP** یا **MIME** نیز قابل استفاده است). به زبان ساده تر، **SOAP** یک پروتکل برای دستیابی به یک سرویس ارایه شده در وب (**Web Service**) است.

- وابسته به محیط پیاده سازی و اجرا نیست. (**Platform Independent**)
- یک پروتکل ارتباطی مبتنی بر **XML** است.
- از دیوارهای آتش (**Firewall**) گذر می‌کند ودیوارهای آتش مانع آنها نمی شوند (**Block** نمی‌شوند).
- برای ارسال پیام استفاده می‌شود.
- برای محیط اینترنت و شبکه طراحی شده اند.

یکی از مسایلی که در دهه اخیر از اهمیت خاصی برخوردار بوده، نحوه ارتباط برنامه های تحت اینترنت با یکدیگر بوده است. همانطور که می‌دانید برنامه های عادی از **RPC** که مخفف **Remote Procedure Call** یا فراخوانی روالهای از راه دور ، برای فراخوانی اشیاء **DCOM** یا **CORBA**، استفاده می‌کنند. اما مشکلی که در این نوع فراخوانی‌ها در بستر اینترنت وجود دارد، مسدود شدن این روشها در

Proxy Server ها و دیوارهای آتش (Firewall ها) است. در صورت استفاده از SOAP با این مشکل روبرو نخواهید بود. SOAP به راحتی شما را قادر خواهد کرد تا بین برنامه هایی که در بسترهای متفاوت طراحی شده اند و در بسترهای متفاوتی در حال سرویس دهی هستند، ارتباط برقرار کنید.

Rest

REST مخفف Representational State Transfer می‌باشد. REST فقط یک سری از دستور العمل‌ها و سبک‌های معماری است که برای انتقال داده‌ها استفاده می‌شوند. این عموما در مورد اپلیکیشن‌های تحت وب کاربرد دارد؛ ولی می‌تواند داده‌ها را به سایر برنامه‌ها نیز ارسال کندخوب ما در این جا با شرح مختصری از وب سرویس REST چیست؟ آشنا خواهیم شد.

RESTful روشی برای ایجاد، خواندن، آپدیت نمودن و یا حذف اطلاعات بر روی سروری است که از HTTP call های ساده استفاده می کنند. در واقع REST یک مدل طراحی برای برنامه های شبکه ای می باشد که ارتباط بین دو سیستم (client-server) را توسط یک پروتکل (مانند http، smtp، ftp و ...) ایجاد می کند. برنامه های بر پایه این روش/معماری، ReSTful application نامیده می شوند، چرا که فقط با request های CRUD (مخفف create update read delete) پروتکل واسط، با هدف تعامل برقرار می کنند.

توسعه دهندگان وب به صورت مکرر در مورد اصول REST و ساختار داده RESTful بحث می‌کنند. چراکه یکی از جنبه‌های حیاتی توسعه وب مدرن است؛ ولی بعضی اوقات این کار فوق العاده گیج کننده می شود.REST به خودی خود یک تکنولوژی نیست ولی می توان گفت روشی است برای ایجاد API هایی با اصول سازماندهی مشخص.

اگر بخواهم جمع بندی کنم، RESTful API ها در واقع API هایی هستند که از معماری REST تبعیت می کنند.

RESTمخفف Representational State Transfer می باشد.

یک معماری وب سرویس است.

از HTTP برای انتقال اطلاعات میان کلاینت و سرور استفاده میکند.

کار کردن با REST بسیار ساده تر از وب سرویس های پیچیده ای مانند SOAP می باشد.

یک سرویس به اصطلاح RESTful عموما بر روی پروتکل HTTP و تمام افعال استاندارد این پروتکل را که توسط مرورگرهای وب قابل درک هستند کار میکند مانند (GET, POST, PUT, DELETE)

OAuth

پروتکل OAuth یک استاندارد فنی برای مجوز دادن به کاربران است. در واقع OAuth پروتکلی برای انتقال مجوز از یک سرویس به سرویسی دیگر بدون به اشتراک گذاشتن اعتبار واقعی کاربر مانند نام کاربری و رمز عبور است. با استفاده از پروتکل OAuth یک کاربر می تواند وارد یک پلتفرم شده و سپس مجاز به انجام اقدامات و مشاهده داده ها در پلتفرمی دیگر باشد.

OAuth انتقال مجوز از برنامه ای به برنامه دیگر را صرف نظر از اینکه چه برنامه ای هستند ممکن می کند. OAuth یکی از رایج ترین روش هایی است که برای انتقال مجوز از یک سرویس احراز هویت یکپارچه (SSO) به یک سامانه دیگر استفاده می شود. پروتکل های دیگر نیز می توانند این کار را انجام دهند اگرچه پروتکل OAuth یک از پرکاربردترین آنها است.

تصور کنید هنگامی که صاحب خانه در خانه نیست بازدید کننده ای به خانه می آید و مالک به جای ارسال کلید برای بازدید کننده کدی موقتی می فرستد تا وارد صندوقی شود که کلید داخل آن است. OAuth نیز به روشی مشابه کار می کند. در OAuth یک سامانه به سامانه دیگر ، به جای ارسال اعتبارنامه کاربر برای اجازه دسترسی به وی، یک توکن دسترسی (authorization token) ارسال می کند.

فرض کنید شخصی می خواهد به برنامه ذخیره فایل ابری سازمان خود دسترسی پیدا کند. فرد قبلا وارد SSO سازمان شده است اما هنوز در آن روز به برنامه ذخیره فایل دسترسی پیدا نکرده است.هنگامی که برنامه ذخیره فایل را باز می کند به جای اینکه به راحتی اجازه ورود به او داده شود، درخواست مجوز برای او از SSOگرفته می شود.

در پاسخ ، SSO توکن دسترسی OAuth را به برنامه ارسال می کند. این توکن شامل اطلاعاتی در مورد حق امتیازات وی در برنامه است. همچنین این توکن محدودیت زمانی خواهد داشت. بعد از مدت زمان مشخصی توکن منقضی شده و باید مجددا وارد SSO شود.توکن های OAuth معمولا با استفاده از HTTPS ارسال می شوند به این معنی که رمزگذاری شده اند. آن ها در لایه ۷ از مدل OSI مورد استفاده قرار می گیرند.

OAuth هم برای اجازه دادن به کاربران و هم برای اجازه دسترسی جزئی سامانه ای به سامانه دیگر مورد استفاده قرار می گیرد. یکی از مواردی که کاربران با آن مواجه می شوند، اجازه دسترسی یک سامانه به یک پلتفرم شبکه اجتماعی یا یک حساب آنلاین دیگر است. حساب های کاربری گوگل می توانند با بسیاری از برنامه های مصرف کننده مختلف مانند پلتفرم های وبلاگ نویسی، وب سایت های خبری و بازی های آنلاین مختلف ادغام شوند. در این موارد از پروتکل OAuth در پشت صحنه استفاده می شود تا این برنامه های خارجی بتوانند به داده های لازم از گوگل دسترسی پیدا کنند.

JWT

برای احراز هوییت روی API ها ما متاسفانه و خوشبختانه نمیتونیم از دو روش دیگه یعنی Session Cookie استفاده کنیم . توی سیستم SOAP که دیتا ها با فرمت XML رد و بدل میشن؛ ما میتونیم با گرفتن Username, Password تو هر درخواست؛ احراز هوییت رو انجام بدیم.

ولی توی سیستم JWT هم میشه همین کار رو انجام داد ولی یکم دچار پیچیدگی وسردرگمی می شیم ولی خوشبختانه یه مشت گیک نشستن تو آفتاب فکر کردن و رسیدن به فرستادن Username, Password روی هدر درخواست. ولی باز یه مشکلی بود؛اگر شما هم به هدر های دیفالت HTTP آشنا باشید؛ به فکر هدر Authorization می افتید؛ پس همون گیکا دیدن نمیشه دوتافیلدو توی این هدر فرستاد پس اومدن توکن و اختراع کردن و اونو توی این هدر فرستادن که اتفاقا خیلی کار برنامه نویسا رو ساده کردن.

وقتی کتابخونه JWT رو به پروژتون اضافه میکنید؛ چند تا متد در اختیارتون میذاره که چندتا از متد های این کتابخونه رو با هم مرور میکنیم:

sign: کارش اینه که **Header** و **Payload** و **SecretKey** و میگیره و **JWT** میده

verify: توکن و **SecretKey** رو میگیره و میگه که توکن درست هستش یا نه

decode: خوندن اطلاعات توی توکن

احراز هویت

سرویس‌های وب فراموش کار هستند یعنی به خاطر نمی آورند چه کسی درخواست را برای آنها فرستاده است. به همین خاطر هر کدام از مشتریان علاوه بر درخواست خود شناسه ای برای احراز هویت خود نیز ارسال می کنند. در سمت سرور ابتدا شناسه از درون درخواست استخراج می شود تا با استفاده از آن هویت کاربر تعیین شود آنگاه درخواست کاربر پاسخ داده می شود. مشتریان به دو روش می توانند خود را به سرور معرفی کنند ۱- روش سنتی مبتنی بر نشست و ۲- مبتنی بر توکن.

در روش مبتنی بر نشست زمانی که کاربر نام کاربری و رمز عبور خود را برای سرور ارسال می کند پس از تایید هویت کاربر کد شناسه برای مشتری ارسال می شود که این کد شناسه توسط مرورگر ها در فایل هایی به نام کوکی ذخیره می شوند. هر زمانی که کاربر درخواستی را برای سرور می فرستد فایل کوکی را نیز به همراه درخواست ارسال می کند در سمت سرور با استفاده از این کوکی هویت کاربر مشخص می شود و پس از مشخص شدن هویت عملیات مورد نظر انجام می شود. پاسخی به مشتری برگردانده می شود.

هر زمانی که کاربر در خواست لوگین به سرور ارسال می کند پس از تایید هویت کاربر رکوردی شامل اطلاعات هویتی کاربر در سرور ایجاد می شود که اصطلاحا نشست می گویند. شناسه این رکورد توسط سرور به مرورگر ارسال می شود که در کوکی ها ذخیره می شود. این شناسه با هر درخواستی به سمت سرور فرستاده می شود. با استفاده از این شناسه رکورد نشست بازیابی می شود. پس در هر لحظه حداقل به تعداد کاربران آنلاین در سرور متغییر نشست وجود دارد.

توکن رشته ای رمز نگاری شده حاوی اطلاعات هویتی کاربر است. برای تولید توکن دو تابع نیاز داریم تابع اول با استفاده از یک کلید خصوصی که بر روی سرور است و اطلاعات هویتی کاربر رشته رمز شده را تولید می کند. قاعدتا نباید به آسانی و بدون داشتن کلید قابل بازیابی باشد. تابع دوم برای انجام عملیاتی معکوس عملیات تابع **f** است این تابع توکن دریافتی از کاربر را با استفاده از کلید دریافت می کند و داده هویتی کاربر را بازیابی می کند.

در روش احراز هویت مبتنی بر توکن زمانی که هویت کاربر تایید شد در سرور توکنی تولید می شود و و آن را برای کاربر ارسال می کند هر زمانی که مشتری درخواستی برای سرور بفرستد باید توکن را نیز ارسال کند دو تفاوت بین نشست و توکن وجود دارد اولین و مهمترین تفاوت این که در سمت سرور هیچ اطلاعاتی در مورد ارتباط برقرار شده با کاربر ذخیره نمی شود. دومین تفاوت این است که توکن معمولا در هدر (سرآیند) پاسخ ارسال می شود و برای نگه داری لازم نیست در کوکی ها ذخیره شود.

قابلیت مقیاس پذیری:

همانطور که بیان شد در روش مبتنی بر نشست هر کاربری که لاگین می کند متغییری در سمت سرور باید ایجاد شود. به نظر می رسد که ایجاد این متغییر ها موجب بروز مشکل خاصی نشود و ظاهرا برای کاربردهای کوچک و متوسط همین گونه است. مشکل خاصی پیش نمی آید. مثال ۱۰۰۰ ارتباط انلاین به راحتی توسط وب سرور ها مدیریت می شوند و مشکل خاصی ایجاد نمی شود. اما تصور کنید برای نرم افزارهایی که میلیون ها یا میلیاردها کاربر دارند همانند جیمیل. شما شاید در بیش از چندین سیستم مختلف حساب جمیل خود را

وارد کنید. حال اگر سرور بخواهد برای هر دستگاهی که به آن لوگین کرده است متغییری را در حافظه خود نگه دارد عملا شدنی نیست یا زمان پردازش طولانی در پی دارد.

امنیت:

مشکل دیگر احراز هویت مبنتی بر نشست مشکلی است که مرورگر ها با کوکی دارد از آنجایی که نگه داری کوکی می تواند موجب بروز مشکل شود بعضی از مرورگر ها دسترسی کوکی ها را غیر فعال می کنند. بسیاری از دستگاه ها و نرم افزارها مکانیزمی برای نگه داری کوکی ندارند. کوکی ها قابل اشتراک نیستند و ...

قابلیت به اشتراک گذاشتن توکن:

یک توکن می تواند توسط چندین دستگاه به اشتراک گذاشته شود مثال جیمیل از یک توکن برای دستگاه های مختلف شما برای شناسایی شما استفاده می کند تا زمانی که اطلاعات پروفایل کاربری شما تغییر کند. زمانی که این اطلاعات تغییر کرد توکن نامتعبر است و کاربر باید دوباره لوگین کند.