## django rest framework ابزاری قدرتمند و انعطاف پذیر برای ساخت Web Api است. Api مخفف Application Programming Interface است. Apiها استانداردی برای تبادل اطلاعات بین دو یا چند ماشین هستند. ماشین ها میتوانند اطلاعاتی که توسط Api به اشتراک گذاشته میشوند را گرفته و به هر شکلی که مایل هستند استفاده کنند. همچنین در صورت نیاز Apiها میتوانند اجازه تغییر در اطلاعات را به ماشین ها دهند. با استفاده از فریمورک Dajngo Rest میتوانید به سادگی اقدام به ساخت Api کنید. Django Rest دارای مستندات فوق العاده قابل فهمی میباشد. یکی از نقاط قوت Django Rest سادگی در serialize کردن model objects میباشد.

================================================================================

api  
api مخفف Application Programming Interface است. api یک رابط برای اتصال چندین ماشین بکدیگر برای تبادل اطلاعات است. با استفاده از یک برنامه میتواند در چند پلتفرم مختلف کار کند.

================================================================================

response  
زمانی که درخواست از سمت کاربر به برنامه ارسال میشود برای پاسخ دادن به درخواست ها میتوان از کلاس Response استفاده کرد. نوع پاسخی که از طریق کلاس Response ارسال میکنید باید یک دیکشنری پایتونی باشد.

https://www.django-rest-framework.org/api-guide/responses/

متد response پارامترهای گوناگونی به عنوان پارامتر دریافت میکند که مهمترین آنها data میباشد که باید به صورت دیکشنری باشد و پاسخی که متد response ارسال میکند به صورت json میباشد   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

هنگامیکه میخواهید از طریق response به ارسال پاسخ بپردازید لازم است اگر از cbv استفاده میکنید از APIView ارث بری کنید یا اگر از fbv استفاده میکنید از دکوریتور api\_view@ استفاده کنید

[https://www.django-rest-framework.org/api-guide/views/#class-based-views](https://www.django-rest-framework.org/api-guide/views/" \l "class-based-views)

[https://www.django-rest-framework.org/api-guide/views/#function-based-views](https://www.django-rest-framework.org/api-guide/views/" \l "function-based-views)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

دکوریتور api\_view متدهایی که باید این فانکشن قبول کند را به عنوان ورودی دریافت میکند اگر چیزی قرار ندهید به صورت پیش فرض فقط get میباشد

==================================================

request  
با استفاده از آبجکت request میتوانید به پارامترهای ارسالی توسط کاربر دسترسی پیدا کنید. همچنین میتوانید به وضعیت کاربر, متد درخواستی یا نوع دیتای قابل قبول دسترسی داشته باشید

https://www.django-rest-framework.org/api-guide/requests/

این request همان requestی هست که ما به هر متدی میدهیم

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

نکته قابل توجه :

هنگامیکه از متد get استفاده میکنید لازم نیست که حتما (/) قرار دهید اما اگر از متد post استفاده میکنید باید حتما (/) قرار دهید (داخل postman)

==================================================

serializer  
زمانی که نیاز به ارسال اطلاعات دیتابیس داشته باشید باید اطلاعات خود را به json تبدیل کرده و ارسال کنید. برای تبدیل model objects به json میتوانید از serializer استفاده کنید. serializerها میتوانند با دو کلاس Serializer و ModelSerializer ایجاد شوند.

https://www.django-rest-framework.org/api-guide/serializers/

==================================================

هنگامیکه تعداده زیادی object رو به serializer ارسال میکنید لازم است که many=true را قرار دهید

ser\_data = PersonSerializer(persons, many=True)

اما اگر یک object رو ارسال میکردید دیگر لازم نبود عبارت بالا رو قرار دهید

اگر در serializer از serializer.Serializer ارث بری کردید لازم است برای هر فیلدی که میخواهید ارسال شود یک بازنویسی انجام شود

==================================================

ModelSerializer  
ModelSerializer نوع دیگری از serializer است که به صورت مستقیم میتواند اطلاعات داخل یک model را به json تبدیل کند. استفاده از کلاس ModelSerializer نسبت به کلاس Serializer مرسوم تر است

[https://www.django-rest-framework.org/api-guide/serializers/#modelserializer](https://www.django-rest-framework.org/api-guide/serializers/" \l "modelserializer)

در ModelSeerializer میتوانید تمام فیلدی هارو به وسیله\_\_all\_\_ ارسال کنید اما جنگو پیشنهاد میکند به صورت صریح هر فیلدی را بنویسید

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

نکته ای که وجود دارد این است که بعضی از مواقع نشان دادن ایمیل افراد کار مناسبی نمیباشد یکی از راه کارها حذف email از fields میباشد اما موقع ساخت person جدید به مشکل میخوریم یک راه مناسب استفاده از optionهای ModelSerializer میباشد که در این مورد write\_only میباشد :

[https://www.django-rest-framework.org/api-guide/fields/#core-arguments](https://www.django-rest-framework.org/api-guide/fields/" \l "core-arguments)

نکته ی مهمی که وجود دارد که اگر در کلاسی که از ModelSerializer ارث بری میکند فیلد جدیدی ساخته شود باید در همان فیلد optionها ساخته شود

out\_of\_iran = serializers.NullBooleanField(required=False, default=False)

اما اگر فیلدی ساخته نشد و به همان فیلدهایی که در fields قرار گرفته بود option اضافه شد از   
extra\_kwargs استفاده میکنیم :

fields = ('id', 'name', 'age', 'email', 'car')  
extra\_kwargs = {'email':{'write\_only':True}}

=======================================================

creating objects

در این قسمت با نحوه ذخیره اطلاعات ارسالی کاربر در دیتابیس آشنا میشوید. برای اینکار اطلاعات ارسالی کاربر را به serializer ارسال میکنیم و با متد is\_valid از معتبر بودن اطلاعات مطمئن میشویم. در این حالت  
django rest framework اطلاعات معتبر را در یک دیکشنری با عنوان validated\_data ذخیره میکند که میتوان از این اطلاعات استفاده کرد.

به دلیل اینکه از modelsrializer استفاده میکنیم دیگر لازم نیست به صورت زیر عمل کنیم :

person(name=info.validate\_data[‘name’], age=info.validate\_data[‘age’], email=info.validate\_data[‘email’]).save

تنها کافیست info را save کنیم :

info.save()

================================================================================

status codes

در این قسمت با status codes در django rest framework آشنا میشوید. status codeها مشخص کننده نوع پاسخی هستند که شما به کاربر ارسال میکنید. status codeها به پنج دسته بندی تقسیم میشوند. status codeها را همراه اطلاعات در کلاس Response ارسال میکنند

https://www.django-rest-framework.org/api-guide/status-codes/

================================================================================

authentication  
در این قسمت با authentication در django rest framework آشنا میشوید. عمل authentication برای احراز هویت کاربرها بکار میرود. باید دقت کنید که authentication فقط برای شناسایی کاربر استفاده میشود و هیچگونه محدودیتی برای دسترسی کاربر ایجاد نخواهد کرد. به طور کلی چهار نوع احراز هویت وجود دارد که در این قسمت با نوع TokenAuthentication آشنا میشوید. همانطور که مشخص است TokenAuthentication برای احراز هویت کاربر براساس token است. هر کاربر رشته ای از کاراکترها به هم ریخته را خواهد داشت که براساس آن احراز هویت خواهد شد

https://www.django-rest-framework.org/api-guide/authentication/

کاربری که authenticate کرده باشد میتوان به وسیله request.user به اطلاعات آن دسترسی پیدا کرد و کاربری که authenticate نکرده باشد به عنوان AnonymousUser در نظر گرفته میشود (میتوانیم به وسیله request.auth به اطلاعات بیشتری نظیر token کاربر دسترسی داشته باشیم)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

اگر در فایل settings.py عبارات REST\_FRAMEWORK را تعریف کردید و داخل آنDEFAULT\_AUTHENTICATION\_CLASSES را مقدار دهی کردید به صورت سراسری authentication رو تمام viewها اعمال میشود اما میتوانید برای هر view که دارید authentication متفاوت تعریف کنید

[https://www.django-rest-framework.org/api-guide/authentication/#setting-the-authentication-scheme](https://www.django-rest-framework.org/api-guide/authentication/" \l "setting-the-authentication-scheme)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

نکته مهمی که وجود دارد این است که در برنامه ای که به وسیله جنگو نوشته میشود اولین چیزی که اجرا میشود authentication و permission ها میباشند  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

basic authentication در موارد تستی کاربرد دارد و از آن در پروژه ها ی عملی زیاد استفاده نکنید   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## [TokenAuthentication](https://www.django-rest-framework.org/api-guide/authentication/" \l "tokenauthentication)

[https://www.django-rest-framework.org/api-guide/authentication/#tokenauthentication](https://www.django-rest-framework.org/api-guide/authentication/" \l "tokenauthentication)

روش های مختلفی برای ست کردن token برای کاربران وجود دارد از signalها گرفته تا استفاده از orm اما بهترین راه این است که هر کاربر خودش بیاد و توکنش را بگیرد

[https://www.django-rest-framework.org/api-guide/authentication/#by-exposing-an-api-endpoint](https://www.django-rest-framework.org/api-guide/authentication/" \l "by-exposing-an-api-endpoint)

در این روش کاربر یوزرنیم و پسورد خودش را وارد میکند و توکن مخصوص به خود را دریافت میکند ( به دلیل اینکه از   
get\_or\_create استفاده شده است توکنی که کاربر دریافت میکند ثابت میباشد و تغییری نمیکند )

اگر authentication موفقیت آمیز بود tokenauthentication دو مورد را set میکند : request.uset که instance از User میشود و توکنی که کاربر داشته داخل request.auth قرار میگیرد

================================================================================

Permission  
در این قسمت با permission در django rest framework آشنا مشوید.با استفاده از permissionها میتوانید برای دسترسی کاربرها محدودیت ایجاد کنید. بعد از اینکه کاربر authenticate شد permissionها با استفاده از request.user و request.auth مشخص میکنند که آیا کاربر اجازه دسترسی خواهد داشت یا نه. permissionها همیشه قبل از هر کدی فعال میشوند. permissionها انواع مختلفی دارند مثل IsAuthenticated که برای شناسایی کاربران لاگین کرده بکار میرود یا IsAdmin که فقط اجازه دسترسی به کاربرهایی که is\_staff آنها برابر با True باشد, میدهد

<https://www.django-rest-framework.org/api-guide/permissions/>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

اگر در فایل settings.py عبارات REST\_FRAMEWORK را تعریف کردید و داخل آنDEFAULT\_PERMISSIONS\_CLASSES را مقدار دهی کردید به صورت سراسری permission رو تمام viewها اعمال میشود اما میتوانید برای هر view که دارید permission متفاوت تعریف کنید

[https://www.django-rest-framework.org/api-guide/permissions/#setting-the-permission-policy](https://www.django-rest-framework.org/api-guide/permissions/" \l "setting-the-permission-policy)

هم برای authentication و هم برای permission اولویت برای اونی هست که به صورت اختصاصی روی view قرار گرفته باشد

================================================================================

ViewSet  
در این قسمت با viewset در django rest framework آشنا میشوید. viewsetها نوعی class based view هستند که عملیاتهایی که مشابه هم هستند را داخل یک کلاس جمع آوری میکند. به طور کلی در هر viewset میتوانید شش نوع متد داشته باشید که نسبت به روش و نوع متدی که کاربر درخواست خود را ارسال میکند فعال میشوند. برای کارکردن با viewsetها میتوانید از متد as\_view استفاده کنید اما معمولا برای رجیستر کردن viewset ها از router استفاده میشود.

در متدهای list و retrieve نباید اطلاعاتی که به serializer ارسال میکنید را به ورودی data بدهید :

def list(self, request):  
 queryset = Person.objects.all()  
 serializer = PersonSerializer(queryset, many=True)  
 return Response(serializer.data)

اما در بحث create و update و … بر خلاف بالا اطلاعاتی که از طرف کاربر آمده است را به ورودی data در seializet بدهید :

def create(self, request):  
 serializer = PersonSerializer(data=request.data)

=======================================================

Routers  
در این قسمت با routers در django rest framework آشنا میشوید. فریمورک drf با استفاده از routers میتواند به صورت اتوماتیک برای viewها شما url ایجاد کند. routerها به دو شکل تعیین میشوند: SimpleRouter و DefaultRouter از routerها بیشتر برای آدرس دهی به viewsetها استفاده میشود.

https://www.django-rest-framework.org/api-guide/routers/

به این صورت است که شما یک router برای viewset خود میسازید و این router به صورت اتوماتیک ۶ تا url برای شما براساس متدها (get, post , ….) برای شما میسازد (البته حداکثر ۶ url مثلا در همین پروژه ۳ url ساخته میشود چرا که تنها از ۳ متد استفاده شده است)

[https://www.django-rest-framework.org/api-guide/routers/#usage](https://www.django-rest-framework.org/api-guide/routers/" \l "usage)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

براساس لینک زیر url ها ساخته میشوند :

[https://www.django-rest-framework.org/api-guide/routers/#usage](https://www.django-rest-framework.org/api-guide/routers/" \l "usage)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

بهتر است که routerها در فایل جداگانه ای نسبت به urlها قرار بگیرد که اسم این فایل در این پروژه api\_urls میباشد

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

basename در routerها همان namespace در urls ها میباشد به طور کلی همان طور که در داکیومنت آمده است اگر شما get\_gueryset خود را ایجاد کردید لازم است که حتما از basename استفاده کنید

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

بعد از اینکه routerهاتون رو ساختید لازم است که به url اصلی برنامه این router ها را بشناسونید :

from .api\_urls import router

urlpatterns = [  
 ….  
 path('api/', include(router.urls)),  
 …..  
]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

به طور کلی DefaultRouter شبیه SimpleRouter میباشد با این تفاوت که root view در DefaultRouter ساخته میشود ( مثلا در همین پروژه اگر داخل url عبارت person را وارد کنید به ارور 404 برمیخورید چرا که از simplerouter استفاده میکنید اما اگر از defaultrouter استفاده میکردید این urlی که وارد کرده بودید به مشکل برنمی خورد )

=======================================================

ModelViewSet  
در این قسمت با کلاس ModelViewSet در django rest framework آشنا میشوید. کلاس ModelViewSet تمامی شش متدی که در ViewSet برنامه نویس مجبور به ساختن آنها بود را به صورت اتوماتیک ایجاد خواهد کرد. در عین حال میتوانید تمام این شش متد را برای نیاز خود override کنید. به طور کلی برای استفاده از ModelViewSet فقط نیاز است یک queryset و یک serializer به آن دهید.

[https://www.django-rest-framework.org/api-guide/viewsets/#modelviewset](https://www.django-rest-framework.org/api-guide/viewsets/" \l "modelviewset)

تفاوت update و partial update در این است که برای update باید تمامی فیلد هایی که در serializer مشخص کرده بودید را ارسال کنید و pattial update همان طور که از اسمش پیداست تنها قسمت از instance باید update شود و لازم نیست تمام فیلد هایی که در serializer تعریف کرده بودید را ارسال کنید

اگر بخواهید تنها از متد get که برای list و retrieve استفاده میشود استفاده شود میتوانید از   
ReadOnlyModelViewSet استفاده کنید

========================================================================

validation  
در این قسمت با validation در django rest framework آشنا خواهید شد. در زمان کار کردن با serializerها میتوانید نسبت به نیاز, اعتبارسنجی های خاص خودتان را به serializer اضافه کنید. دو نوع validation را میتوانید به serializerها اضافه کنید: اول Field-level validation برای زمانی است که میخواهید فقط یک فیلد از یک آبجکت را اعتبارسنجی کنید. دوم Object-level validation برای زمانی که میخواهید چند فیلد از یک آبجکت را اعتبارسنجی کنید

[https://www.django-rest-framework.org/api-guide/serializers/#validation](https://www.django-rest-framework.org/api-guide/serializers/" \l "validation)

در Field-level validation به دلیل اینکه میخواهید یک فیلد را validate کنید به این صورت انجام میشود

def validate\_name(self, value):  
 if value == 'admin':  
 raise serializers.ValidationError('name cant be admin')  
 return value

[https://www.django-rest-framework.org/api-guide/serializers/#field-level-validation](https://www.django-rest-framework.org/api-guide/serializers/" \l "field-level-validation)

value حاوی مقدار fieldی است که شما میخواهید روی آن validate انجام دهید (در این مثال name)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

و در object-level validation به دلیل اینکه چندین فیلد قرار است validate شوند و این فیلد ها با هم در ارتباط هستند از روش زیر باید استفاده شود

[https://www.django-rest-framework.org/api-guide/serializers/#object-level-validation](https://www.django-rest-framework.org/api-guide/serializers/" \l "object-level-validation)

data لیستی هست حاوی مقدار فیلد هایی که فرستاده شده است

================================================================================

throttling  
در این قسمت با throttling در django rest framework آشنا خواهید شد. throttling هم مثل permission اقدام به ایجاد محدودیت برای درخواست های کاربران میکند. اما با این تفاوت که throttling تعداد درخواست های کاربران را کنترل میکند. به طور مثال throttling میتواند به کاربران معمولی اجازه ارسال تعداد 100 درخواست در روز را بدهد. این محدودیت را میتواند در روز, ساعت, دقیقه و ثانیه اعمال کند.

https://www.django-rest-framework.org/api-guide/throttling/

۳ نوع throttling داریم : اولین مورد AnonRateThrottle میباشد که درباره کاربرانی است که authenticate نکرده اند   
دومین مورد UserRateThrottle میباشد که مربوط به userهایی میباشد که authenticate انجام داده اند

و سومین مورد ScopedRateThrottle میباشد که زمانی مناسب است که شما قصد دارید بخشی از api را محدود کنید

[https://www.django-rest-framework.org/api-guide/throttling/#scopedratethrottle](https://www.django-rest-framework.org/api-guide/throttling/" \l "scopedratethrottle)

اسم هایی که در DEFAULT\_THROTTLE\_RATES در داکیومنت بالا تعریف شده اند ( uploads و contacts ) اختیاری میباشد و شما میتوانید هر اسم دیگری را جایگزین کنید

================================================================================

SessionAuthentication

در این قسمت با SessionAuthentication در django rest framework آشنا خواهید شد.به شما این ویژگی را میدهد که بتوانید از طریق مرورگر هم عملیات های authenticate را انجام دهید

[https://www.django-rest-framework.org/api-guide/authentication/#sessionauthentication](https://www.django-rest-framework.org/api-guide/authentication/" \l "sessionauthentication)

توجه شود که حتما باید به url اصلی پروژه اضافه شود

path('api-auth/', include('rest\_framework.urls')),

================================================================================

Renderer   
در این قسمت با Renderer  در django rest framework آشنا خواهید شد.همان طور که از اسمش مشخص است پاسخ نهایی را که قرار است به کاربر نشان دهید را render میکند ( البته داخل browser) . فریمورک drf به صورت پیشفرض چندین Renderer را در اختیار شما قرار داده تا بتوانید نوع پاسخی که به کاربر برمیگردانید را مدیریت کنید. همچنین امکان ایجاد Renderer مخصوص خودتان هم وجود دارد. فریمورک drf به صورت پیشفرض از BrowsableApiRenderer استفاده میکند که اجازه دسترسی به اطلاعات api را داخل مرورگر میدهد.JSONRenderer برای نشان دادن اطلاعات به شکل json و AdminRenderer برای نشان دادن اطلاعات به شکل admin panel جنگو استفاده میشود

https://www.django-rest-framework.org/api-guide/renderers/

================================================================================

pagination

در این قسمت با نحوه صفحه بندی اطلاعات در django rest framework آشنا خواهید شد.

https://www.django-rest-framework.org/api-guide/pagination/

================================================================================

# serializer relations

در این قسمت با serializer relation در djagno rest framework آشنا میشوید. برای مدیریت کردن نحوه نمایش رابطه بین modelها در serializer از serializer relation استفاده میشود. به صورت دیفالت یعنی اگر از relation استفاده نکنید pkای که مربوط به model میباشد را برمیگرداند

https://www.django-rest-framework.org/api-guide/relations/

اولین مورد StringRelatedField میباشد که به صورت str آن چیزی که در \_\_str\_\_ قرار دادید را نمایش میدهد

[https://www.django-rest-framework.org/api-guide/relations/#stringrelatedfield](https://www.django-rest-framework.org/api-guide/relations/" \l "stringrelatedfield)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

دومین مورد HyperlinkedRelatedField میباشد که اگر در همین پروژه برای model car یک serializer ایجاد کرده بودید لینکی به وجود می آورد که با کیلیک بر روی وارد صفحه detail آن car میشدید

[https://www.django-rest-framework.org/api-guide/relations/#hyperlinkedrelatedfield](https://www.django-rest-framework.org/api-guide/relations/" \l "hyperlinkedrelatedfield)

================================================================================