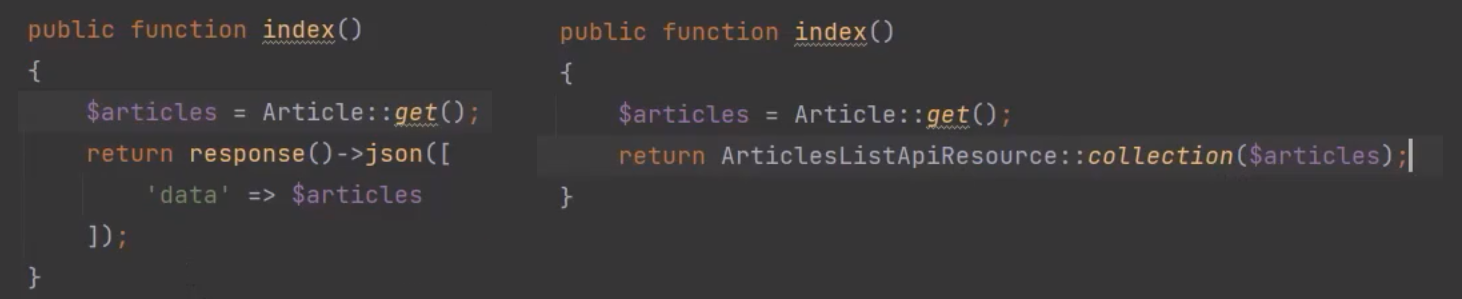
**API Resources یه ابزاریه که یه لایه تبدیل (یک لایه میانی بین مدلها و دیتاهایی که میخوایم به کلاینت بفرستیم) ایجاد میکنه؛ که میتونیم دیتاهای مدل‌هامون رو که قرار بفرستیم به کلاینت**



**رو؛ ی سری تغییرات بدیم (ترانزفرم)؛ هر ساختاری رو که بخوایم رو خیلی راحت میتونیم مشخص کنیم ( متد toArray - چه دیتاهایی باشه؛ چیا نباشه) و راحت هم میشه تغییرش داد**

**و کارهای دیگه ==> تمام این کارها رو؛ خیلی راحت؛ با api resource میتونیم انجام بدیم.**

**دیگه نیازی نیست که خودم Trait بسازم؛ متد success و failed بذارم؛ ساختار ریسپانس json رو تعیین کنم و... ==> تازه کلی فیچرهای دیگه رو هم بهمون میده.**

****

**مثال از فیچر تبدیل – transform (کار تبدیل رو چطور میشه انجام داد): تبدیل created\_at به ساختار ماه و روز؛ دیگه سال و ساعت حذف :دی | حرف اول first\_name بزرگ باشه**

**مثلا: کلاینت لیست یوزرها رو میخواد. ما این دیتاها رو از مدل User میگیریم؛ ی سری تغییرات روی دیتاها اعمال میکنیم و میفرستیم سمت کلاینت**

**اون تغییرات رو باید در لایه transform انجام بدیم ==> به ازای هر مدل؛ یک لایه transofrm میسازیم؛ و این لایه transofrm یک سری امکانات به ما میده برای اعمال تغییرات روی دیتاها**

**پس هر وقت خواستیم دیتاهای مدلها رو بصورت json؛ با یه فرمت انعطاف پذیر؛ قابل مدیریت – کنترل تبدیل کنیم از این ابزار استفاده میکنیم.**



**چه کارهایی با استفاده از Api Resources:**



1. **میتونیم تعیین کنیم که چه دیتاهایی/اطلاعاتی در پاسخ json ارسال بشه و چطور نمایش داده بشن: حذف دیتاهای غیر ضروری(کاهش حجم داده های ارسالی) و ارسال دیتاهای ضروری**



1. **مدیریت دیتاهای روابط مدلها: میتونیم دیتاهای مربوط به روابط بین مدلها رو مدیریت کنیم (کاری که در مورد اول گفته شد) و در پاسخ json قرار بدیم.**
2. **امکان سفارشی‌سازی آسان: به راحتی میتونیم دیتاهای مدلهای الکوئنت رو به json تبدیل کنیم + با هر ساختاری که مد نظرمونه ==> کنترل - مدیریت دیتاهای ریسپانس**

**تبدیل created\_at به هر ساختاری که مد نظرمونه | حرف اول first\_name بزرگ باشه**

1. **Pagination: میتونیم صفحه بندی – pagination داشته باشیم**
2. **سفارشی سازی براساس شرایط: میتونیم داده ها رو بر اساس شرایط خاصی نمایش بدیم: مثال: نشون دادن ی سری از دیتاها فقط واسه یک سری از کاربرا**
3. **جداسازی کدها: با جدا کردن لاجیک نمایش از بیزنس لاجیک، کدها خواناتر و نگهداری آن‌ها آسان‌تر می‌شود.**
4. **- قابلیت استفاده مجدد: میتونیم در قسمت‌های مختلف پروژه با تنظیمات مختلف از api resourceها استفاده کنیم.**

**چرا API Resources ایجاد شد؟**

**دیتا: وقتی میخوایم داده های مدل ها رو به فرمت json تبدیل کنیم؛ باید دستی این کار رو انجام بدیم => استفاده از آرایه‌های دستی یا متدهای accessor در مدل‌ها**

**اگه بخوایم یک کلیدی رو حذف یا اضافه کنیم به ساختار json؛ باید تمام کنترلرها رو تغییر بدیم**

**تحلیل – مشکلات ==> مشکلاتی که این روش ها دارن**

* **تکرار کد: برای هر مدل، باید کد تکراری برای تبدیل به JSON نوشته می‌شد.**
* **عدم انعطاف‌پذیری: تغییر در ساختار JSON نیازمند تغییر در کدهای مدل‌ها بود.**
* **کاهش خوانایی کد: مخلوط شدن منطق تبدیل داده با منطق مدل، باعث کاهش خوانایی و نگهداری کد می‌شد.**
* **افزایش احتمال خطا**
* **صرف زمان زیاد**

**API Resources با ارائه یک لایه انتزاعی، این مشکلات را حل می‌کند:**

* **کاهش تکرار کد: با تعریف یک Resource برای هر مدل، منطق تبدیل به JSON در یک مکان متمرکز می‌شود.**
* **افزایش انعطاف‌پذیری: می‌توانید به راحتی ساختار JSON را با تغییر کد در Resource و بدون تغییر در مدل‌ها؛ تغییر دهید.**
* **کاهش پیچیدگی کد: با جدا کردن منطق تبدیل داده از منطق مدل، کد شما ساده‌تر و قابل نگهداری‌تر می‌شود.**
* **بهبود خوانایی کد: جدا شدن منطق تبدیل داده از منطق مدل، باعث می‌شود کد تمیزتر و قابل فهم‌تر شود.**
* **توسعه‌پذیری بهتر: API Resources به شما امکان می‌دهند تا به راحتی قابلیت‌های جدیدی به API خود اضافه کنید.**