//extends class yaratıldığnda main class da yaratılır , kendi contructu varsa override edilir

\_\_construct instanse alındığında \_\_destruct son çalışan fx dir

Static function , instanse alınmadan çalışan fx dir , prop da olabilir . static hızlı çalışır ama extra ram ayrılır.

Static $deger = 123 ; fx içinde $this ile değil , self::deger diye çağırılır

Static değişken her çağrılığında değeri değişebilir , üzerine yazılabilir. Statik olmayan bir prop, hep aynı kalır

Trait :: bir class sadece bir class dan türeyebilir , başka bir parent ı olabilmesi için trait olarak tanımlanmış classlar kullanılır. Use anahtar kelimesi ile , use TraitClass, DigerTrait ... varsa dahil edilir. Trait in fx ve proplarına class içinde parent:: ile ulaşılır.

Interface :: içindeki tüm fxler public olmalıdır. Başka bir sınıfın işlemleri için gerekli olan fx leri sağlamakla yükümlüdür. Yaratılan sınıf , implament edilen interface in ne olduğuna değil , yapabildiklerine bağlıdır.

Interface , kural bildirir , içinde getContent(); getName(); gibi fxler varsa , implement edildiği class da da aynı adlı fxler olmak zorunda yoksa fatal error !

Abstract :: soyut class , instance ı alınamayan classtır sadece kendinden extend edilecek classlar için base oluşturur. Abstract class da kural belirleyebilir.

**abstract public function** al($b);

şeklinde , extend edilen class da al($b) isimli bir fx in bulunmasını zorunlu kılar.

İnterface de değişken tanımlayamazken , abstract da değişken tanımlanabilir.

NAMESPACE ::: classların çakışmasını engellemek için kullanılır. Bir dosyada farklı namespaceler çağrıldığında ve aynı isimde class lar içerdiğinde

**use** ulasUzayi\baglan **as** yeniisim;

şeklinde farklılık yaratılabilir.

Chaining Method :: laraveldeki class::where()->get() gibi artarta çalışan fxler chaining

dikkat edilmesi gereken şey fx sonuna return $this eklemek enson çalışan fx de ister return $this dersin ister demessin sırayla çalışırlar

ROUTER :::