object هایی دارید که

* ایجاد کردن اونها هزینه زیادی داره (از نظر منابع سیستمی)
* مایلید تا سطح دسترسی به اون Object ها رو (از نظر امنیتی منظورمه) به نوعی کنترل کنید
* قصد دارید تا به Object ای روی ماشین Remote دسترسی پیدا کنید (همون ایده Proxy/Stub در COM/DCOM).
* مایلید تا هنگام دسترسی اون Object ها، کاری اضافه بر سازمان انجام بدید

اگر چنین Object هایی در سیستم دارید، Proxy Pattern میتونه به شما کمک کنه. واژه Proxy به معنای نماینده یا وکیل هستش، نماینده ای برای دسترسی به Object اصلی. ایجاد این نماینده (بر خلاف Object اصلی) هزینه بر نیست. این نماینده هنگامیکه شما واقعا به اون Object اصلی نیاز داشته باشید اونو ایجاد میکنه و در دسترس شما قرار میده (در حقیقت با به تاخیر انداختن ایجاد Object اصلی، از Resource های سیستمی به خوبی بهره میبره)، ضمن اینکه قدرت این الگو اونجا نمایان میشه که شما بصورت مستقیم به Object اصلی دسترسی نداشته باشید و لزوما مجبور باشید از طریق Proxy فرایند مورد نظر رو پیش ببرید.  
  
به نمودار زیر دقت کنید:

همونطوریکه می بینید، عناصر مهم این الگو عبارتند از:

* ISubject: اینترفیسی که RealSubject و Proxy هر دو اونو پیاده سازی میکنن.
* RealSubject: کلاس اصلی که "عملکردش" از طریق Proxy در دسترس دیگران قرار میگیره.
* Proxy: کلاسی که RealSubject رو ایجاد و کنترل میکنه. این کلاس میتونه سطح دسترسی به RealSubject رو در اختیار خودش نیز بگیره.
* Request: درخواست مورد نظر که از طریق Proxy پردازش میشه یا به RealSubject اونو delegate میکنه.
* Adapter provides a different interface to its subject. Proxy provides the same interface. Decorator provides an enhanced interface.