**الگوی Strategy**

**یکی از پرکاربردترین الگوهاییست که به وفور در پروژه ها استفاده می شود. کار این الگو این است که الگوریتم های داینامیک به یک کلاس ارسال کنیم .**

**مثلا برای مرتب سازی میتوانیم از الگوهای مختلف مرتب سازی استفاده کنیم.شما در یک کلاس از یک الگو استفاده میکنید اما اگر نیاز به جایگزین کردن این الگو باشد باید کد کلاس را ویرایش کنید. برای حل این مشکل توسط این الگوی طراحی شما می توانید الگوی مرتب سازی مورد نظر خود را به کلاس مربوطه پاس دهید.**

**مثال بیشتر فرض کنید تعدادی Entity به عناوین زیر دارید :**

**User,Product,Category,Brand**

**و میخواهیم یک الگوریتمی برای مرتب سازی این موارد بنویسیم، با این دید که مرتب سازی برای هر کدام برای ما متفاوت است مثلا User براساس Name ، Product براساس Price ،Category براساس Id و Brand براساس Name**

**این الگو به صورت Runtime الگوریتم ها را تغییر میدهد.**

**نکات**

1. **توسعه الگلوریتم مستقل از داده ها**
2. **توسعه به صورت static یا ExtensionMethod**
3. **ارسال استراتژی به Context از طریق تزریق وابستگی**
4. **استفاده از Method Injection**
5. **استفاده از استراتژی پیش فرض در Context ( باعث اتصال محکم میشود بهتر است استفاده نشود)**
6. **استفاده از visitor به جای Strategy**

**مزایا**

1. **رعایت اصل SRP**
2. **رعایت اصل Open/Close**
3. **تغییر الگوریتم ها در زمان اجرا**
4. **جداسازی پیاده سازی الگوریتم از کدی که از آن استفاده میکند**

**معایب**

1. **کلاینت باید تفاوت بین استراتژی های پیاده سازی شده را بداند**
2. **افزایش پیچیدگی کد**

**چه زمانی از این الگو استفاده کنیم؟**

1. **زمانی که الگوریتم های زیادی برای انجام یک کار داریم و میخواهیم در زمان اجرا الگوریتم های تغییر دهیم.**
2. **زمانی که الگوریتم های پیچیده ای برای حل مسائل داریم که ارتباطی به کلاینت ندارد.**
3. **زمانی که کلاس ما تعداد زیادی عملگر شرطی برای استفاده از الگوریتم های متفاوت باشد.**