دیزاین پترن ها

دیزاین پترن چیست؟

دیزاین پترن یا الگوی طراحی به راه حل های رایج مشکلات پر تکراردر توسعه نرم افزار ها گفته میشه.

دیزاین پترن ها متفاوت از آرشیتکت پترن ها یا الگو های معماری مانند mvc هستند.

دیزاین پترن ها معمولا دارای ساختار استاندارد زیرند:

در ابتدا بیان مسئله و مشکل رایج ,

سپس بیان راه حلش

و در آخر بیان فواید و زیان های استفاده از این الگوی حل مسئله.

لیست چندتا از مهمترین دیزاین پترن ها در توسعه وب به ترتیب زیر میباشد:

1 Singleton pattern

2 Observer pattern

3 Decorator pattern

4 Factory pattern

5 Strategy pattern

6 Repository pattern

1. دیزاین پترن singleton (سینگلتون):

سینگلتون برای حل مسئله وجود تنها یک نمونه از بعضی کلاس هاست که اگر بیش از یک نمونه داشته باشند ممکن است باعث اضافه بار یا از بین بردن ثبات در نرم افزار شوند.

راه حل این دیزاین پترن این است که از ایجاد نمونه ای جز نمونه اولیه کلاس جلوگیری کنیم.

مشکلات احتمالی استفاده از این دیزاین پترن این است که :

اولا میتونه فهم نرم افزار و تست و دیباگ رو سخت تر کنه, چونکه با استفاده از این پترن ما داریم یک استیت گلوبال در نرم افزار ایجاد میکنیم که در هرجای نرم افزار میشه به این کلاس دسترسی پیدا کرد و تغییراتی درش ایجاد کرد.

پس بهتره کلاس هایی که از این پترن استفاده میکنن زیاد قابل تغییر نباشن.

دوما ناقض اصل کاپلینگ (حداقل ارتباط داشتن بین اجزای مختلف نرم افزار) هست.

سوما در نرم افزار های پیچیده تر که با چند هسته کار میکنن میتونه مشکل ساز بشه اگر بدرستی اجرا نشده باشه.

مثال استفاده از کلاس های سینگلتون در فایل زیر است.

design patterns/ Singleton.php

1. دیزاین پترن Observer (آبزرور):

Observer برای حل مسئله نبود ارتباط بین آبجکت های مرتبطه.

با استفاده از این دیزاین پترن هرزمان که یک آبجکت تغییری درش ایجاد بشه , آبجکت های لازم هم بروز میشن.

از معایب این دیزاین پترن اینه که بخاطر نقض اصل loss coupling در شی گرایی باعث پیچیده تر شدن سیستم میشه.

و از طرف دیگه هم میتونه استفاده از مموری رو افزایش بده .

اما مهمترین ایراد این دیزاین پترن , پیچیده کردن فرایند دیباگه.

تفاوت این دیزاین پترن با dependency injection اینه که در دپندنسی اینجکشن , پیشنیاز های کلاس در شروع کار به کلاس اضافه میشه اما در آبزرور کلاس هایی که به کلاس اصلی طزریق میشن, توسط این کلاس اطلاع رسانی میشن از تغییرات جدید.

مثال استفاده از این دیزاین پترن:

design patterns/ observer.php

1. دیزاین پترن Decorator )دکوراتور):

این دیزاین پترن برای حل مسئله افزودن فانکشن و کارکرد جدید به کلاس بدون تغییر در ساختارش.

مشکلات دیگه ای که برطرف میکنه اینه جلوی ایجاد کلاس های فرزند اضافه رو میگیره .  
 از طرف دیگه باعث برقراری اصل open-closed یعنی پذیرا بودن کلاس به کارکرد های جدید اما بسته بودن به تغییر در کلاس میشه.  
  
از معایب استفاده از این دیزاین پترن به این موارد میشه اشاره کرد که در مواردی که میلی ثانیه ها مهمن استفاده از این پترن میتونه اضافه بار به کارکرد نرم افزار ایجاد کنه.  
میتونه استفاده از dependency injection رو سخت کنه. و در آخر اگر بد اضافه بشه باعث پیچیدگی نرم افزار بشه.

مثال در فالیل design patterns/ Decorator.php

1. دیزاین پترن Factory (فکتوری):

این دیزاین پترن فرایند ساخت آبجکت رو آسون میکنه.

با استفاده از این دیزاین پترن وابستگی بین آبجکت های مختلف مشخصتر میشه.

کپسوله کردن فرایند ساخت از سمت کاربر.

از معایب استفاده از این دیزاین پترن اینه که با تکیه زیادی بر کلاس factory اصل decupling نقض میشه.

مثال در فالیل design patterns/ factory.php

1. دیزاین پترن استراتژی:

انختاب داینامیک الگوریتم , برای نرم افزار هایی که نیاز دارن در ران تایم از الگوریتم های متفاوت استفاده کنند.

استفاده از این دیزاین پترن انعطاف پذیری نرم افزار رو برای ماژولار کردن و گسترش نرم افزار بدون تغییرکد های قدیمی افزایش میده.

از معایب استفاده از این دیزاین پترن. افزایش پیچیدگی و سنگینی نرم افزار رو میشه نام برد.  
  
مثال در فایل strategy/strategy.php