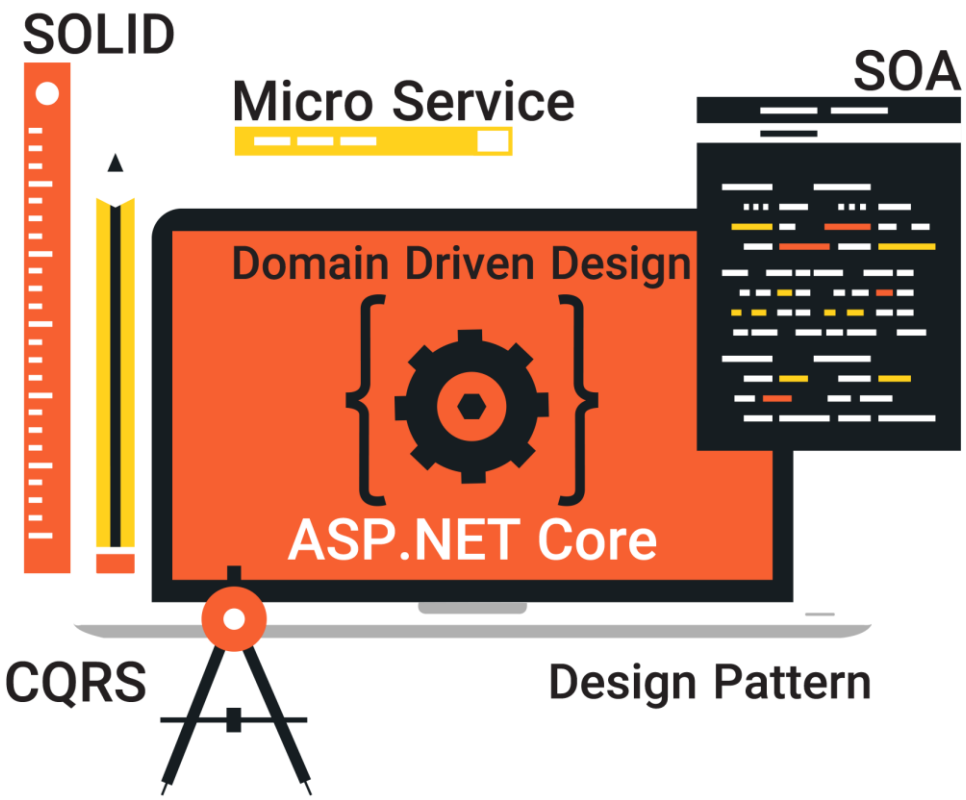


Domain Driven Desgin

Visual Studio

www.NikAmooz.com

مدرس: علیرضا ارومند

nikamooz;
آموزش برنامه نویسی و اجرای پروژه

جلسه دهم

معرفی علیرضا ارومند

۱. **مدرس و مشاور** ASP.NET Coreو معماری‌های نرم‌افزاری **(نیک آموز)**

۲. **مدیر فنی خبرگزاری نسیم**

۳. **کارشناس ارشد توسعه نرم افزار داتین (فناپ)**

۴. **کارشناس ارشد توسعه نرم افزار ارتباط فردا (بانک آینده)**

۵. **متخصص انجام پروژه‌های وب و .NET**

۶. **و...**

Value Objects

چه خواهیم آموخت؟

۱. Value Objectو تعریف آن
۲. چه زمانی استفاده می‌شود؟
۳. معرفی الگوهای توسعه Value Object
۴. ...

مقدمه

۱. به خاطر مقدار خواص مهم است
۲. شناسه نداشتن کار را راحت می‌کند
۳. Immutable هستند

مقدمه

4. باید Combinable باشند
5. راحی برای خلاصی از مشکلات Primitive

کاربرد Value Object

۱. وضعیتی از Entity را نشان می‌دهد

۲. توضیحی بر وضعیت است

شناسه و Value Object

۱. برای کار نیاز به شناسه نیست
۲. منطقا شناسه ندارند
۳. شناسه در Entity
۴. عدم نیاز به شناسه نشانه‌ای برای شناخت
۵. شناسایی ویژگی‌ها کلید موفقیت

وضوح کد با Value Object

۱. DDD بر وضوح کد تاکید دارد

۲. Primitive Type وضوح ندارد

۳. Primitive Type از خطا مصون نیست

وضوح کد با Value Object

۴. نیاز به بسته بندی Primitive Typeها

۵. خوانایی کد بالاتر می‌رود

۶. منطق‌های مرتبط یکجا جمع می‌شوند

ویژگی‌های Value Object

۱. بدون شناسه
۲. وابسته به مقدار
۳. ارائه ویژگی
۴. چسبندگی بالا
۵. قابل ترکیب
۶. Immutable
۷. Self-Validatable
۸. تست پذیری بالا

Identity Less

۱. شناسه ندارند

۲. معمولا بعد از Entity شناسایی می‌شوند

۳. می‌توانند به دلایلی Id داشته باشند

Attribute Base

۱. تساوی به شرط مساوی بودن ویژگی‌ها
۲. زبان سی شارپ این قابلیت را ندارد
۳. بازنویسی Equal

Attribute Base

۴. بازنویسی == و =!

۵. بازنویسی GetHashCode

Behavior Rich

۱. ارائه عملکردهای مورد نیاز یکجا

۲. پنهان سازی داده‌ها

Cohesive

۱. در برگیرنده مفاهیم کاملا مرتبط

۲. شامل یک تا هر تعداد خاصیت

IMMUTABLE

۱. دریافت مقدار موقع ساخت
۲. ارائه متد با نام مناسب برای تغییرات مختلف
۳. تغییر مقدار یعنی ساخت نمونه جدید
۴. عملکرد مانند DateTime

Combinable

۱. می‌توان دو نمونه را ترکیب کرد
۲. حاصل ترکیب شی جدید است
۳. نیاز به بازنویسی + و -

Self-Validatable

۱. هیچ گاه در وضعیت نادرست نباشد
۲. وظیفه چک کردن با خود شی
۳. معمولا موقع ساخت مقادیر چک

الگوهایی برای توسعه Value Object

در ادامه بعضی الگوها برای توسعه Value Object را می‌بینیم

Static Factory Method

۱. وظیفه پنهان سازی سازنده

۲. مانند Timestamp

Micro/Tiny Type

۱. افزایش وضوح به کمک Value Objectها

۲. افزایش وضوح بیشتر با Micro Type

مجموعه ای از Value Objectها

۱. مجموعه معمولا نیاز به Id دارد

۲. معمولا نشانه ای از Entity است

ذخیره و بازیابی

۱. ذخیره به کمک NoSql ها ساده
۲. در SQL مشکل Normalization
۳. استفاده از مدل Flat شده در SQL
۴. استفاده از ORM برای کار با Flat Object

Entities

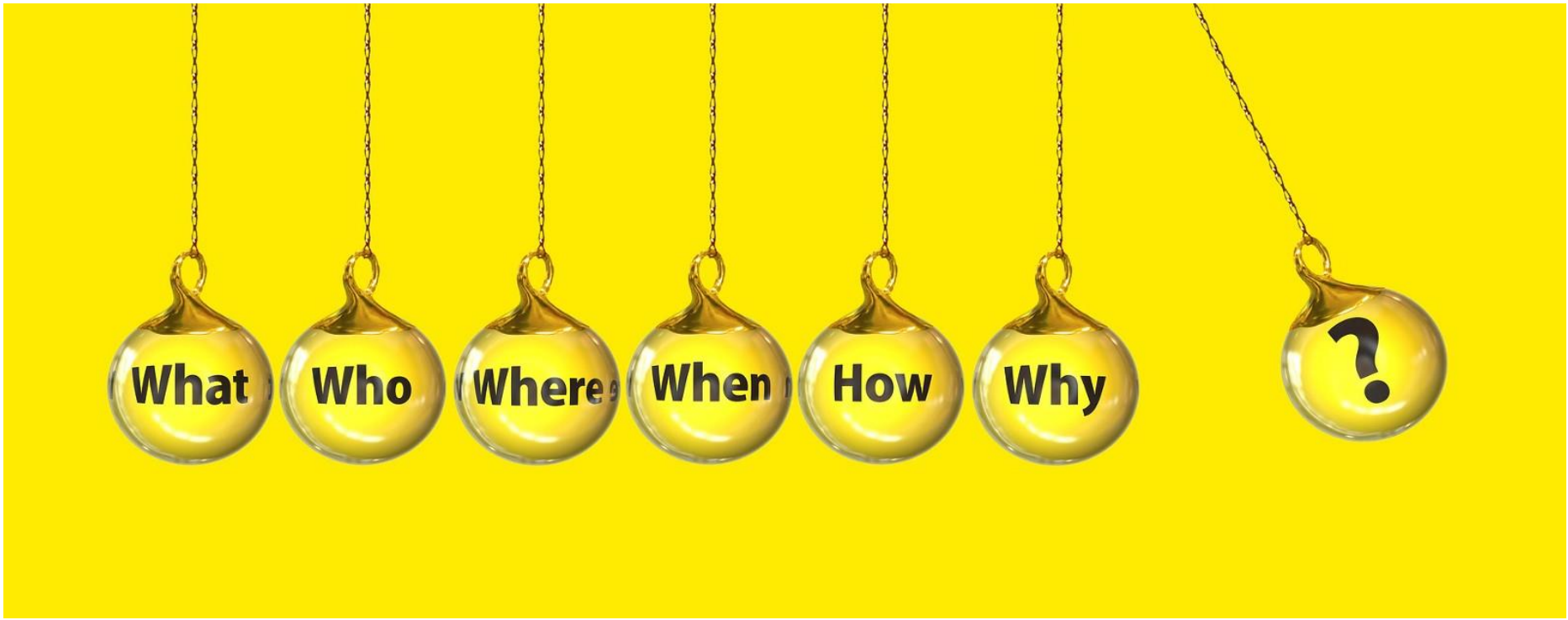
چه خواهیم آموخت؟

۱. تشخیص Entityها در صورت مسئله
۲. تفاوت Entity و Value Object
۳. الگوهای طراحی Entityها
۴....

مقدمه

۱. مفاهیمی داریم که شناسه دارند

۲. یکی از مهم‌ترین مسائل طراحی یافتن Entity



آشنایی کامل با Entity

۱. قابل بررسی از جهات مختلف

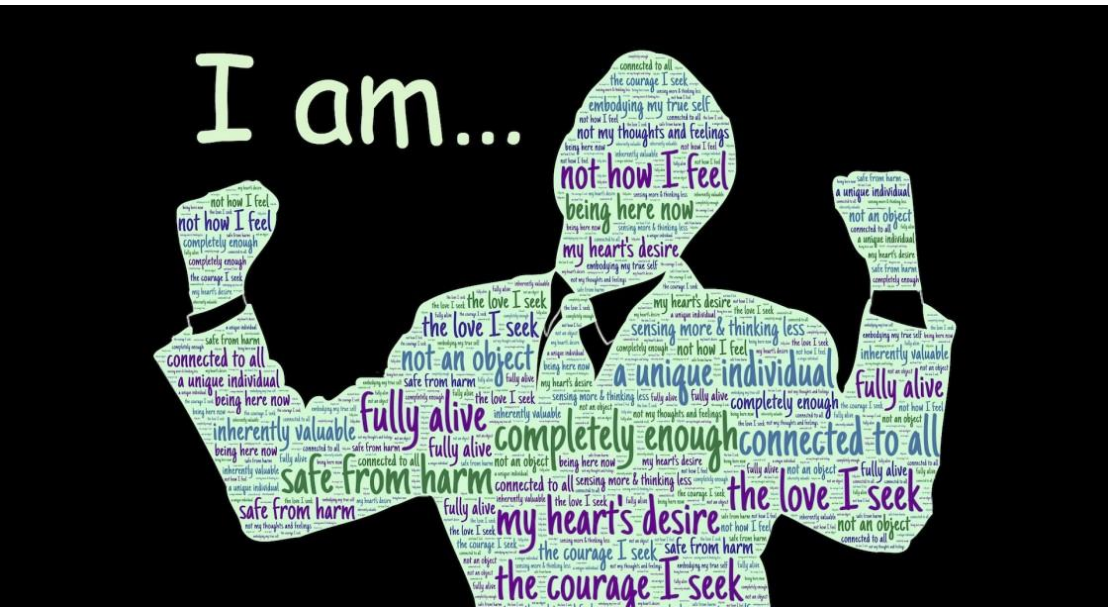
۲. در ادامه ویژگی‌های مختلف را بررسی میکنیم

Identity و ماندگاری Entityها

۱. از نحوه صحبت BEها قابل تشخیص

۲. در صورت تفاوت ID حتما متفاوت

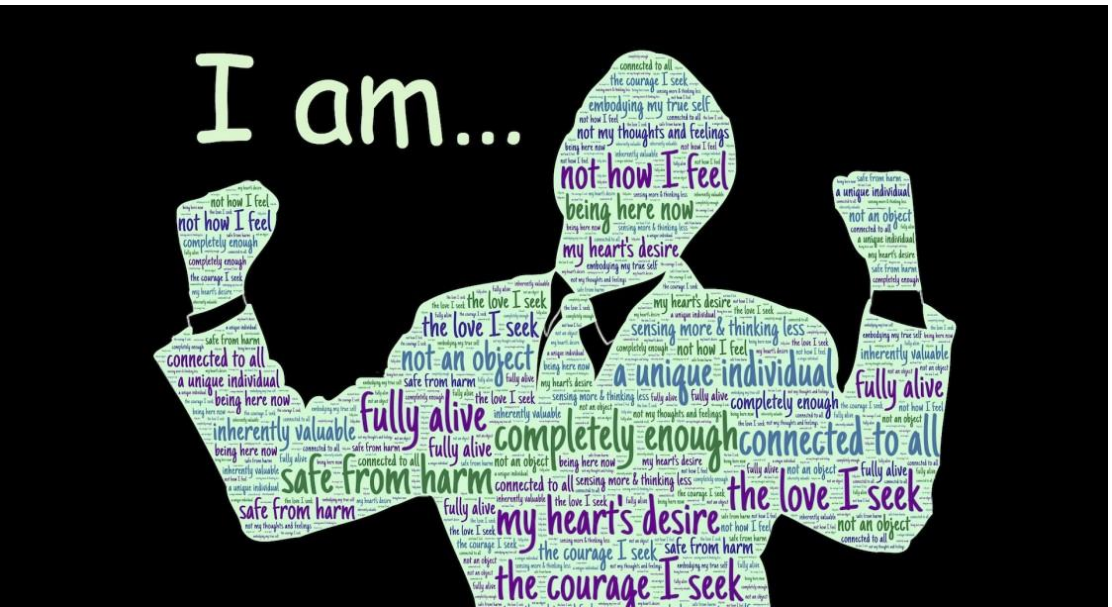
۳. باید در تعاملات شناسه را بیابیم



Identity و ماندگاری Entityها

۴. تشخیص وجود شناسه یا نیاز به ساخت آن

۵. اگر طول عمر مطرح باشد احتمالا Entity



وابستگی به Context

۱. تمایز Entity/Value Object چالش

۲. Entity ذاتا شناسه دار است

۳. در Context معنا می‌یابد

پیاده سازی Entity

۱. ویژگی‌های مشترک داریم
۲. نحوه تخصیص شناسه ویژگی اول
۳. Validate کردن ویژگی دوم
۴. بررسی چگونگی واگذاری مسئولیت

تخصیص شناسه

۱. ممکن است ذاتا شناسه داشته باشد

۲. بعضا نیاز به تولید شناسه داریم

۳. Natural Key/کلید تولیدی

Natural Key

۱. شناسه ذاتی موجود در دامنه
۲. کدملی، ISBN، نام کشور
۳. اطمینان از عدم تغییر شناسه

Natural Key

۴. معمولا پارامتر سازنده

۵. بررسی امکانات ORM پیش از انتخاب

شناسه تولیدی

۱. در صورت نیاز باید تولید شود
۲. استفاده از شمارنده عددی
۳. استفاده از GUID

شناسه تولیدی

۴. استفاده از مقادیر رشته ای

۵. تولید شناسه در پایگاه داده

چقدر منطق مجاز است؟

۱. باید اصل SRP رعایت شود
۲. مدیریت Identity مسئولیت اصلی
۳. رعایت SRP با واگذاری مسئولیت

چقدر منطق مجاز است؟

۴. افزایش وظایف کاهش خوانایی

۵. Entityها شلوغ می‌شوند

Self-Validatable

۱. باید همیشه در وضع صحیح باشد
۲. در طول عمر ممکن است تغییر کند
۳. راه‌های مختلفی برای اعلام خطا

Invariants

۱. نوع خاصی از Validation
۲. در صورت عدم صحت شی وجود ندارد
۳. عدم تغییر در طول عمر

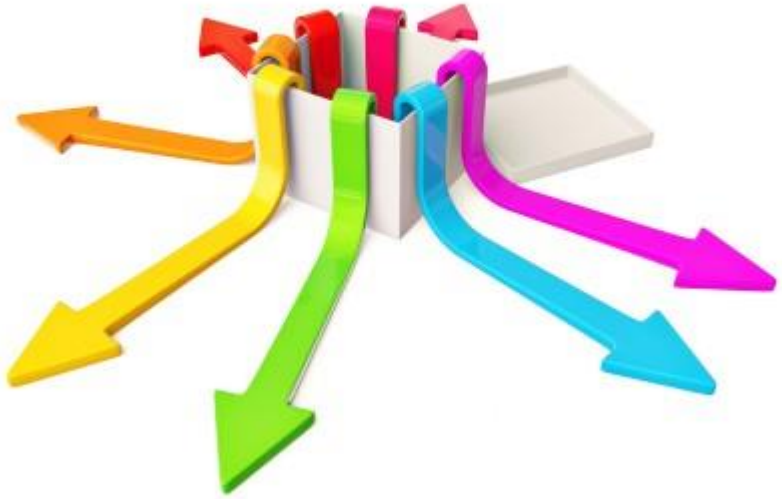
عملگرایی به جای داده محوری

۱. باید نمایشی از عملکردها باشند
۲. عدم انتشار اطلاعات
۳. دسترسی مستقیم به داده احتمال خطا

توسعه به اندازه نیاز

۱. فقط توسعه نیازمندی‌ها نه کل دنیای واقعی
۲. بعضا کدهایی داریم که هرگز اجرا نمی شود

توزیع پذیری

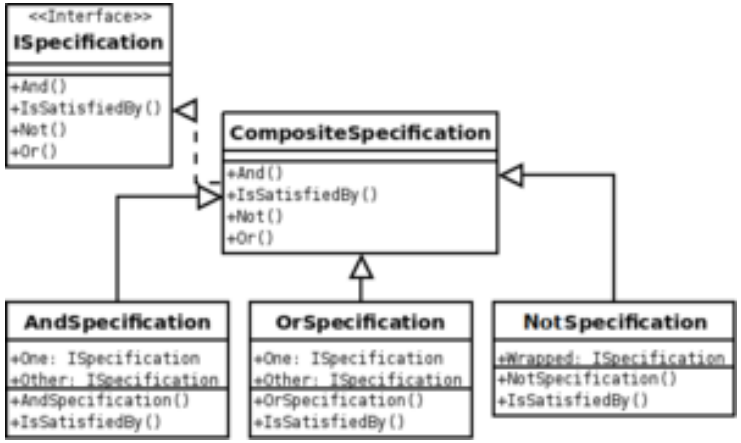


۱. توزیع شدگی برنامه‌ها روشی متداول
۲. نباید یک entity توزیع شود
۳. خاصیت ها را توزیع می‌کنیم

الگوهای توسعه

هنگام توسعه می‌توان از الگوهایی استفاده کرد که در ادامه مورد بررسی قرار می‌گیرد

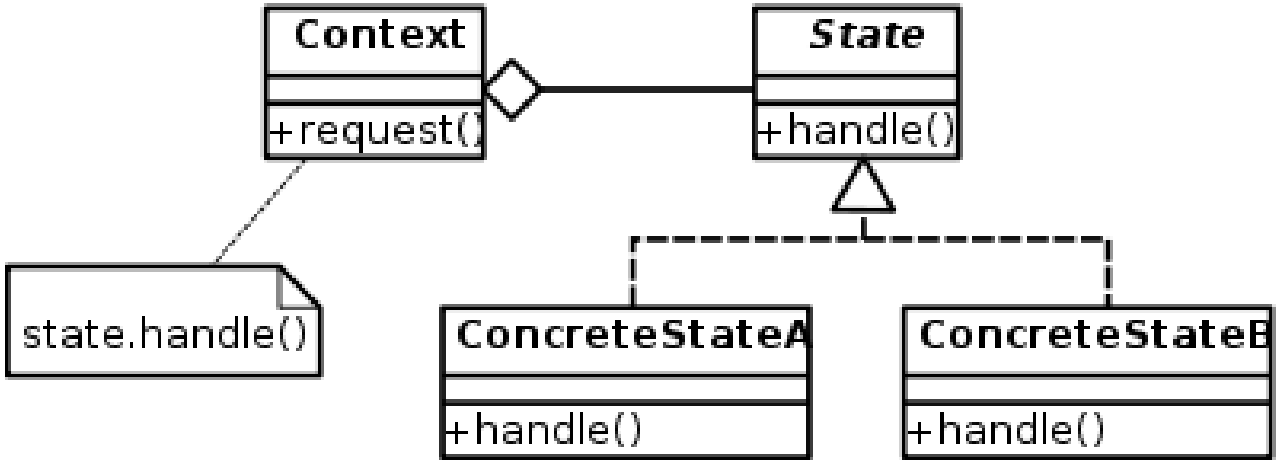
صحت عملکرد با Specification



۱. کلاسی کوچک با یک مسئولیت
۲. رعایت SRP
۳. تست پذیری بالا
۴. روشی برای واگذاری مسئولیت

State Pattern و Entityها

۱. عملکرد در طول عمر تغییر می‌کند
۲. عملکرد وابسته به وضعیت
۳. Stateهای زیادی بدون پیاده سازی

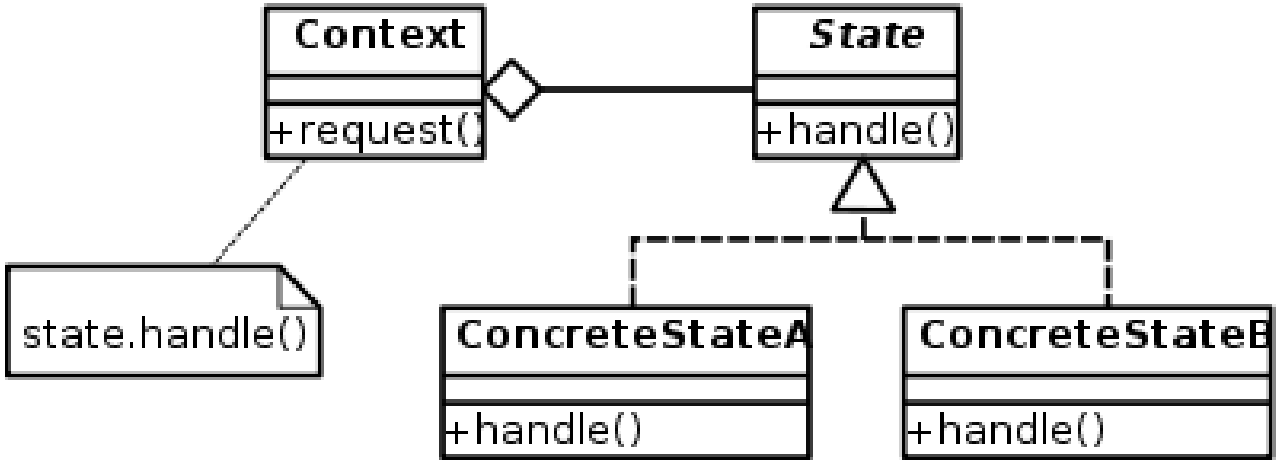


State Pattern و Entityها

۴. وضوح کار کم می‌شود

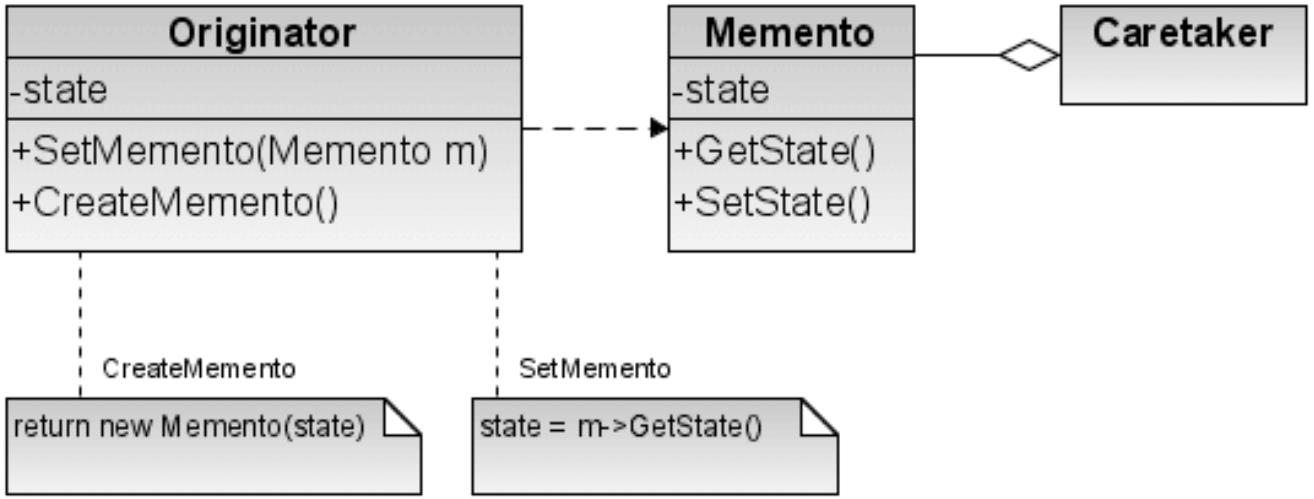
۵. طراحی چند Entity مختلف

۶. متدهای اضافی کم می‌شود



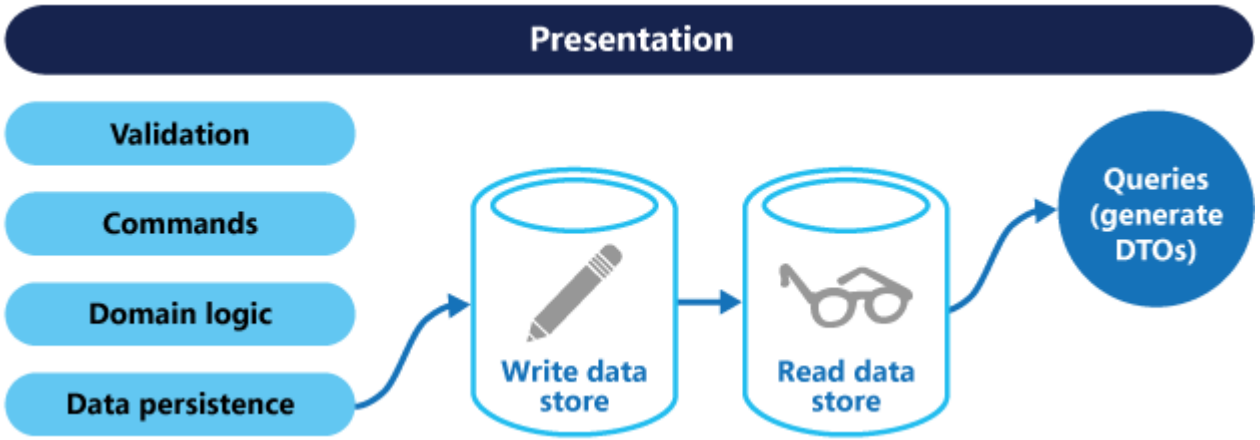
دسترسی به وضعیت درونی

۱. در قسمت های مختلف نیاز به داده‌ها
۲. استفاده از الگوی Memento
۳. فقط داده‌ها خارج می‌شود نه ساختار داده‌ها



Side Effects

۱. باید از تاثیرات جانبی جلوگیری شود
۲. بعضا تاثیرات جانبی پنهان است
۳. یکی از دلایل عدم رعایت CQRS



شبکه‌های اجتماعی نیک آموز

اطلاع رسانی سریع کارگاه‌های نسبتا رایگان،

کوپن‌های تخفیف، مقلات، فیلم و دوره‌های نیک آموز

Instagram

Telegram