Gerência

Professor: José Eurípedes Ferreira de Jesus Filho
jeferreirajf@gmail.com

Universidade Federal de Jataí – UFJ

Antes de começar, o que é domínio mesmo?

E modelo?

E design?

Domain Driven Design

- O isolamento do **design do domínio** com relação à massa dos **outros detalhes** de um sistema de software esclarece bastante a ligação do design com o modelo.

- A adoção de **padrões** ajuda a gerar um modelo prático para implementação.

Domain Driven Design

- A **parte do software** que realmente resolve especificamente problemas do domínio geralmente constitui apenas **uma pequena porção** do sistema como um todo, embora sua importância seja desproporcional ao seu tamanho.

- Precisamos **desacoplar os objetos de domínio** de outras funções do sistema para que evitemos confundir os conceitos do domínio com outros conceitos relacionados somente à tecnologia do software.

Domain Driven Design

- É comum **escrever códigos de suporte** (código relacionado a interface, a banco de dados, a conexão com infra-estrutura, etc) **diretamente nos objetos do negócio**. É a maneira mais fácil e direta, no curto prazo. Isso faz com que **o código relacionado ao domínio fique escondido** e fica difícil raciocinar sobre ele.

DDD - Arquitetura em camadas

- O princípio essencial da arquitetura em camadas é que qualquer elemento de uma camada dependa somente dos elementos da mesma camada ou das camadas abaixo dela.

- É na camada do domínio que reside o modelo.

- O design e a implementação da lógica do negócio constitueem a camada do domínio.

O **isolamento da camada de domínio** é um pré-requisito do DDD.

Domain Driven Design

- Em sempre é recomendado aplicar DDD.
    1. Equipe imatura;
    2. Problema muito simples;
    3. Solução deve ser rápida e de baixo custo;
    4. Sistema não tem horizonte para expandir;
    5. Etc.

Entidade vs Objetos de Valor vs Serviço

- **ENTIDADES**: Algo com continuidade e identidade, que seja rastreado através de diferentes estados.

- **OBJETO DE VALOR**: Atributo que descreve o estado ou característica de algo.

- **SERVIÇO**: Aspectos do domínio que são expressos como ações ou operações.

Entidade vs Objetos de Valor vs Serviço

- No **DDD** os **MÓDULOS** fazem parte do modelo e devem refletir conceitos do domínio.

- Uma identidade conceitual precisa combinar através das várias implementações dos objetos e das diferentes formas dentro de um sistema.

Entidade vs Objetos de Valor vs Serviço

- A modelagem de objetos tende a fazer com que nos concentremos nos atributos de um objeto, mas o conceito fundamental de uma **ENTIDADE** é uma continuidade abstrata que percorre um ciclo de vida e até mesmo passa por várias formas (sua identidade).

- Exemplo: Depósito em contas bancárias

Entidade vs Objetos de Valor vs Serviço

- É comum que uma identidade seja relevante fora do sistema computacional, contudo, existem casos onde a identidade só existe dentro do modelo.

- Uma mesma coisa do mundo real pode ser representada como entidade em um modelo mas pode não ser uma entidade em outro modelo. Exemplo: venda de assentos em estádio.

Entidade vs Objetos de Valor vs Serviço

- Atributo chave vs ID único.

- É um erro comum tentar transformar cada objeto de domínio em uma **ENTIDADE**.

- Existem objetos de domínio que servem simplesmente para descrever características de algo.

Entidade vs Objetos de Valor vs Serviço

- Objeto de valor vs Entidade.

- Um objeto que representa um aspecto descritivo do domínio sem nenhuma identidade é um **OBJETO DE VALOR**.

- Trate o objeto de valor como **imutável**.

Entidade vs Objetos de Valor vs Serviço

- **SERVIÇOS DE MODELO**

  - Às vezes a situação simplesmente não se trata de uma coisa. Às vezes o design inclui operações que não pertencem conceitualmente a nenhum objeto.

- **SERVIÇOS**
  - Operações de domínio importantes que não encontram um lar natural em nenhuma entidade ou objeto de valor.

Entidade vs Objetos de Valor vs Serviço

- Operações complexas podem **obscurecer objetos simples** e geralmente juntam muitos objetos de domínio, seja coordenando-os, seja colocando-os em ação. Essas responsabilidades adicionais acabam criando uma dependência forte entre todos os objetos.

- Um **SERVIÇO** é uma operação oferecida como interface que fica isolada no modelo.

Entidade vs Objetos de Valor vs Serviço

- Um serviço é um verbo ao invés de um substantivo. É necessário nomear um serviço de acordo com a linguagem onipresente.

- Definindo um bom serviço
    1. A operação está relacionada a um conceito do domínio e que não é parte natural de uma ENTIDADE ou de um OBJETO DE VALOR.
    2. A interface é definida em termos de outros elementos do modelo do domínio.
    3. A operação não possui um estado.

Entidade vs Objetos de Valor vs Serviço

- Serviços de Domínio vs Serviços de Aplicativo vs Serviços de Infraestrutura

    - Serviços de domínio e serviços de aplicativo trabalham em conjunto com serviços de infraestrutura.

Módulos

- Os Módulos da camada do domínio deve surgir como uma parte significativa do modelo, contando a história do domínio em uma escala maior.

- Módulos não trata somente do código ser dividido de forma coesa, mas sim dos conceitos - Baixo acoplamento e alta coesão.

- Se um modelo está contando uma história, os MÓDULOS são os CAPÍTULOS.

Módulos

- Módulos são muito mais trabalhosos de serem refatorados do que classes.

- CUIDADO com o EMPACOTAMENTO dirigido pela INFRAESTRUTURA
    - Divisão mais evidente costuma ser as camadas.
    - Divisão lógica dos assuntos.

Módulos

- Use o empacotamento para separar a camada do domínio de outros códigos. Do contrário, deixe o máximo de liberdade possível para que os desenvolvedores do domínio possam empacotar os objetos de domínio de forma a suportar suas escolhas de modelo e de design.

Agregados, Fábricas e Repositórios

- FÁBRICAS
    - Utilizadas para criar e reconstruir objetos de domínio complexos ou até mesmo agregados inteiros.

- REPOSITÓRIOS
    - Utilizadas para encapsular toda a complexidade de deletar, adicionar ou recuperar objetos/agregados na infraestrutura (banco de dados, por exemplo).

- AGREGADOS
    - Encapsulamento interno de objetos de domínio que fazem sentido juntos e que precisam proteger suas regras internamente.

Agregados, Fábricas e Repositórios

- É difícil manter a consistência das alterações feitas em objetos em um modelo com associações complexas.

- Um AGREGADO é um grupo de objetos associados que tratamos como sendo uma unidade para fins de alterações de dados. Cada AGREGADO possui uma raiz e um limite. O limite define o que está dentro do AGREGADO. A raiz é uma ENTIDADE única e específica contida no AGREGADO e é por ela que objetos externos tem permissão de fazer referências.

Agregados, Fábricas e Repositórios

- Regras que abranja diferentes agregados não precisa estar sempre atualizadas. Elas podem ser atualizadas através de eventos ou processamento em lote. Regras dentro de um AGREGADO devem estar sempre atualizadas.

Agregados, Fábricas e Repositórios

- Agrupe as ENTIDADES e os OBJETOS DE VALOR em AGREGADOS e defina os limites em torno de cada um. Escolha uma ENTIDADE para ser a raiz e controle todo o acesso aos objetos dentro do limite através da raiz.

- Os AGREGADOS demarcam o escopo para as invariantes.

Agregados, Fábricas e Repositórios

- A criação de um objeto pode ser uma importante operação por si só, mas operações complexas de montagem não se encaixam com a responsabilidade dos objetos de domínio.

- Uma FÁBRICA encapsula o conhecimento necessário para criar um objeto complexo ou um AGREGADO inteiro.

Agregados, Fábricas e Repositórios

- Um REPOSITÓRIO representa todos os objetos de um determinado tipo como sendo um conjunto conceitual.

- Um REPOSITÓRIO tretira um grande fardo das costas do cliente, que agora pode se comunicar com uma interface simples, reveladora de intenções e pedir o que precisa em termos do modelo.

Agregados, Fábricas e Repositórios

- Vantagens de um REPOSITÓRIO

    1. Oferecem aos clientes um modelo simples para obter objetos persistentes e controlar seu ciclo de vida.
    2. Desacoplamento da tecnologia de persistência.
    3. Comunica decisões de design.
    4. Permitem fácil substituição da implementação facilitando, inclusive, testes.

Agregados, Fábricas e Repositórios

- Normalmente as equipes adicionam uma estrutura à camada de infraestrutura para suportar a implementação do REPOSITÓRIO.