Introdução à Engenharia Software

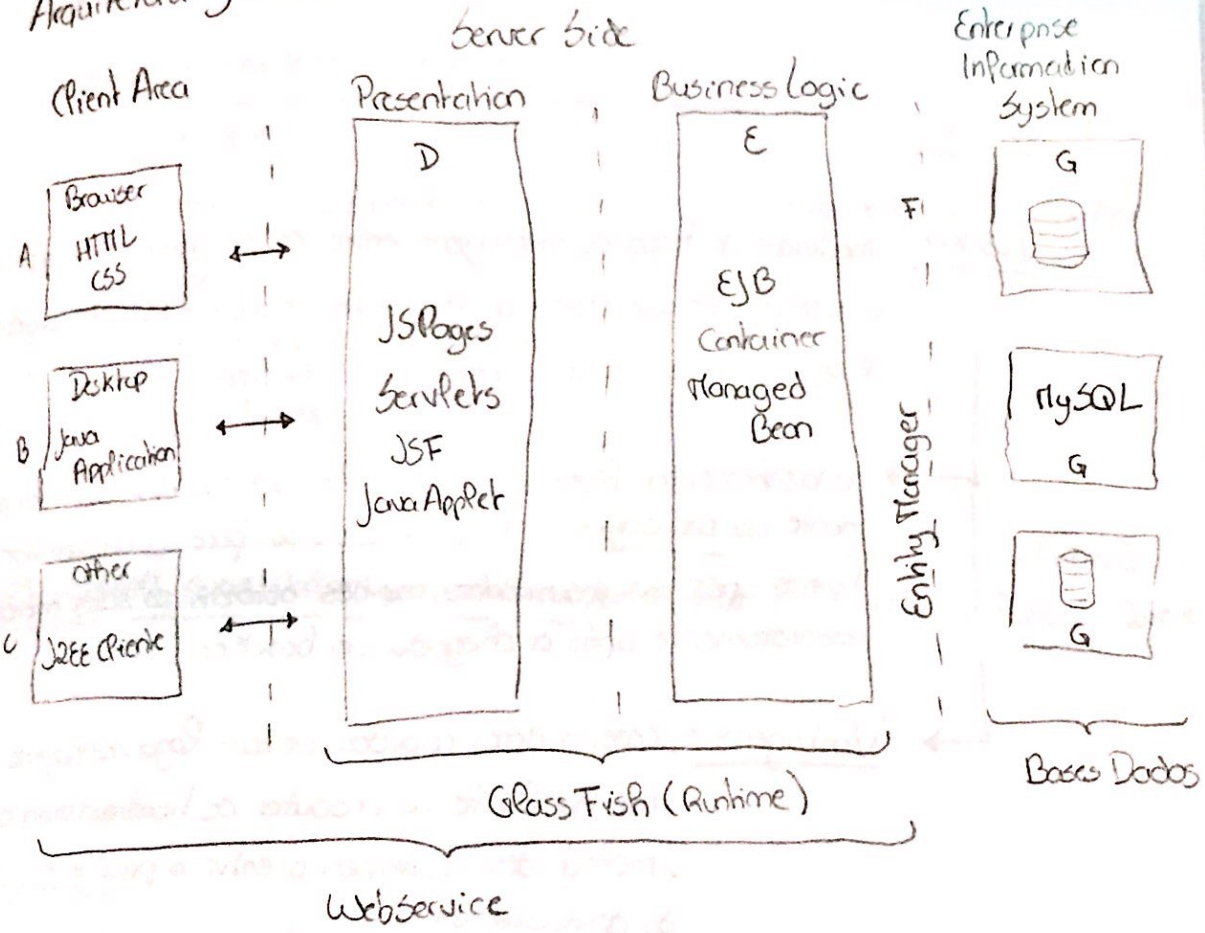
Resumos
2017/2018

Carolina Albuquerque | 80038

IES

Resumos Primeiro Teste

Arquitetura Java EE



@EJB	Classe	são componentes que permitem encapsular a camada lógica do negócio
@Stateless	classe	permite que durante o envio de info. sejam feitas as configurações necessárias para que o bean em questão não guarde os estados dos clientes
@Table	classe	permite especificar a tabela para a persistência da entidade

@WebService — classe — indica que a entidade implementa um webservice usado para a comunicação de sistemas

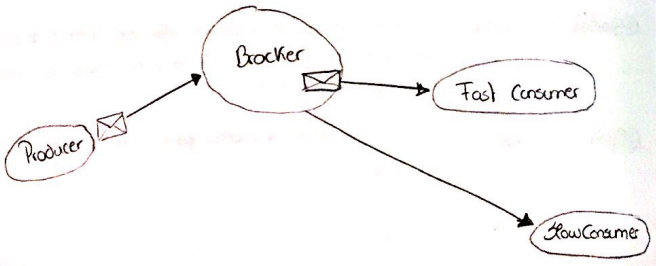
@OneToMany — Método — indica que a relação entre objetos.
o objeto indicado na anotação é composto por vários objetos de outra classe
associação unidirecional de um para muitos

Messaging

Broker - mediador de troca de mensagens entre os endpoints produtor e o consumidor que recebe as mensagens transmitidas pelo produtor e os redistribui para os consumidores existentes

→ armazenamento ficticios temporários que irá guardar temporariamente as mensagens em queue que irão para os consumidores
Pertos após os consumidores rápidos obterem as suas mensagens imediatamente após a chegada ao broker.

→ **Vantagens:** . consumidores rápidos obtêm logo mensagens
. não há limite ao produtor de transmissão de mensag
. melhor desacoplamento entre o produtor e os endp do destinatários
. o produtor não tem que saber os endereços, o brok é que sabe




```
public abstract class EndPoint {
    protected Channel channel;
    protected Connection connection;
    protected String endPointName;

    public EndPoint (String endpointName) throws ... {
        this.endPointName = endpointName;

        ConnectionFactory factory = new ConnectionFactory();
        //hostname of rabbitMQ server
        factory.setHost("hostname");
        //getting Connection
        connection = factory.newConnection();
        // creating a Channel
        channel = connection.createChannel();
        channel.queueDeclare(endpointName, false, false, false, null);
    }
    ...
}
```

Create Message Handler

Define and Publish Queue

No pom.xml

```
<dependency>
    <groupID> commons-lang </groupID>
    <artifactID> commons-lang </artifactID>
    <version> 2.6 </version>
</dependency>
```

```
public class Producer extends EndPoint {
    ...
    public void sendMessage (Serializable object) throws ... {
        channel.basicPublish("", endPointName, null, SerializationUtils.serialize(object));
    }
}
```

Publish message to queue

Teste 2014/2015 - Resolução

Java EE Architecture

①

webservice estão nas áreas D e E?	Verdadeiro
Glassfish inclui áreas A, D e E?	Falso
Managed Bean estão nas áreas D e E?	Falso
JSF inclui a área E?	Falso
Entity Manager está na área F?	Verdadeiro
Containers estão na área D e E?	Falso
JSP pages estão na área A e D?	Falso
MySQL só pode estar na área G?	Verdadeiro
EJB estão nas áreas D e E?	Falso

②

@EBJ	classe	são componentes que permitem encapsular a camada da lógica do negócio
@stateless	classe	permite que durante o deployment sejam feitas as configurações necessárias para que o bean em questão não guarde os estados do cliente
@Table	classe	permite especificar a tabela para ~~que~~ a persistência da entidade
@webservice	classe	indica que a entidade implementa um webservice
@OneToMany	Método	indica a relação entre objetos. o objeto indicado na anotação é composto por vários objetos de outra classe

③ Os Java EE containers executados no servidor Java EE são responsáveis pela gestão de vários EJB em execução na aplicação em questão

Annotations

EJB Annotations:

@stateless - indica que a classe é uma stateless bean (não guarda est dos anteriores)

@statefull - indica que a classe é uma statefull bean (guarda todos os estados anteriors)

@Message Driven Bean

@EJB

@Singleton

@Local

@Remote

@Post Active

Java Persistent:

@ID
@Generated Value
@Transient

} Annotation for fields

@OneToOne
@OneToMany
@ManyToOne
@ManyToMany
} Relationship Mapping

@Entity - Classe

Jws Annotations

@ WebService
@WebMethod
@ Get
@Produces
@Path

JPA Anotations:

@ Entity
@Table
@OneToMany
@ OneToOne
@PersistenceContext

Servlet:

> Classe Java usada para extender as funcionalidades de um servidor
> Normalmente usadas para as aplicações hospedadas por web servers.
> Utilizam protocolo HTTP, não sendo restritas a ele.

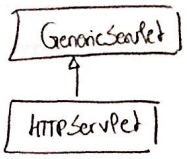
⚠ Nota: Um servlet precisa de um container web para ser executado

São usados para: - Processar ou armazenar dados que foram submetidos de um formulário HTML
- Fornecer conteúdo dinâmico, como os resultados de consulta de uma base de dados
- Gerir a informação de estado que não existe no protocolo sem estado HTTP, como inserir/retirar itens de uma cesta de compras de um cliente específico

> Classe Java no JavaEE que obedece à API Java Servlet (um protocolo pelo qual uma classe Java pode responder a requisições).
> Podem comunicar sobre qualquer protocolo cliente-servidor, mas são usadas na maior parte das vezes com o protocolo HTTP.

Ideia Principal:
> Um desenvolvedor de software pode usar um servlet para adicionar conteúdo dinâmico para um servidor web usando a plataforma Java.

Hierarquia: Extendem direta ou indiretamente da classe GenericServlet.



Conceitos e Processos de Testing

① Qualidade de Software:

Avalia o grau que um produto a nível de software cumpre os requisitos ~~definidos~~, atende às expectativas do cliente em relação aos atributos de ~~qualidade~~ e se cumpre as melhores práticas na indústria.

② Verificação vs Validação:

Verificação - Resposta à pergunta "Estamos a implementar o sistema de forma correta?"
- Verificar os work-products face às respetivas especificações
- verificar módulos de consistência

Validação - "Estamos a implementar o sistema certo?"
- verificar os work-products face às necessidades e expectativas dos utilizadores

Testes
- Gerar Risros e ganhar confiança nos resultados
- obter informação sobre o processo de construção
- verificação de erros / bugs

SOLID

S - Single Responsability Principle
O - Open Closed Principle
L - Liskov Substitution Principle
I - Interface Segregation Principle
D - Dependency Inversion Principle

Single Responsability Principle:
> Cada classe deve ser apenas responsável por realizar uma tarefa

Open-Closed Principle
> Indica que as entidades do código devem estar abertas para extensão mas fechadas para modificação.

Liskov Substitution Principle
> Diz que qualquer tipo de filho de um tipo pai deve ser capaz de substituir esse pai sem qualquer erro

Interface Segregation Principle
> Diz que se deve preferir muitas, mas pequenas, interfaces específicas do cliente numa interface maior
> Os clientes não devem ser obrigados a depender de interfaces que não utilizam

Dependency Inversion Principle:
> Escrever código que depende de abstrações e não de detalhes concretos

GRASP

- High Cohesion (Alta Coesão):

É um padrão avaliativo que tenta manter os objetos adequadamente focados, gerenciáveis e compreensíveis.

As responsabilidades de um determinado elemento estão fortemente relacionadas e altamente focadas.

> Quebra Programas em classes e subsistemas - aumenta a coesão
> Baixa Coesão - elemento tem muitas responsabilidades distintas e não relacionadas.

- Low Coupling (Baixo Acoplamento):

Medida de quão forte um elemento está conectado, tem conhecimento ou depende de outros elementos

> Menor Dependência entre classes
> Mudança numa classe com menor impacto noutras
> Maior Potencial Reutilização

Padrão que determina como atribuir responsabilidades de suporte.

- Polimorfismo:

A responsabilidade de definir a variação dos comportamentos com base no tipo é atribuída ao tipo para o qual essa variação ocorre.

O uso do tipo deverá usar operações polimórficas em vez de ramificações explícitas com base no tipo.

- Information Expert (Especialista na informação)

Determinar onde delegar responsabilidades (métodos, campos computados)

Olhar para a responsabilidade → Determinar a informação → Determinar onde a informação está armazenada

- Creator (Criador):

Classe B deve ser responsável por criar instâncias da classe A:

> Instâncias B contêm ou agregam instâncias A;
> " " gravam instâncias A;
> " " utilizam de perto instâncias A;
> " " têm as informações de inicialização das instâncias A e pas sam isso na criação

- Indirection (Indireção):

Suporta baixo acoplamento e potencial redução entre dois elementos, atribuindo a objeto intermediário a responsabilidade de ser mediador entre eles.

- Pure Fabrication (Invenção Pura):

Fabricação - classe artificial que não representa um conceito no domínio do problema especialmente feito para conseguir baixo acoplamento, alta coesão e potencial de reutilização.

- Protected Variations (Variações Protegidas):

O padrão variações protegidas protege elementos das variações noutros elementos envolvendo o foco de instabilidade com uma interface gráfica e usando polimorfismo para criar vários implementações desta interface

Enterprise Java Beans (EJB)

> Componente da plataforma JEE que roda num container de um servidor de aplicação.

> Principal Objetivo: fornecer um desenvolvimento rápido e simplificado de aplicações Java.

Tipos de EJB:

① Entity Bean - representa um objeto que vai persistir numa base dados ou uma unidade de armazenamento.

② Season Bean - executa uma tarefa para o cliente. Pode manter o estado durante uma sessão com o cliente (Statefull) e ou não (Stateless)

③ Message Driven Beans: Processa mensagens de modo assíncrono entre EJB's e cuja API de mensagens é JMS (Java Message Service)

> As interfaces de Beans podem ser locais ou remotas.

Java Persistance API

> API padrão da linguagem Java que descreve uma interface comum para frameworks de persistência de dados.

> Define um meio de mapeamento objeto-relacional para objetos Java simples e comuns denominados beans de entidades

> Gere o desenvolvimento de entidades do Modelo Relacional usando a plataforma Java SE e Java EE.

Annotations

EJB

@Stateless - indica que a classe é stateless bean (não guarda estados anteriores)

@Statefull - indica que a classe é um statefull bean (guarda todos os estados anteriores)

@Message Driven Bean - indica que a classe é um Message Driven Bean

@EJB - usado para especificar ou injetar a dependência como instância EJB noutro EJB

@Local - usado para especificar as interfaces locais de um session bean

@Remote - usado para especificar as interfaces remotas de um session bean

@PostActive - usado para especificar o método de retorno do ciclo EJB.

@Singleton -

JPA

@Entity - designar uma classe como uma entidade para poder ser usada com serviços JPA

@Table - especifica a tabela primária associada à entidade

@ID - Tipo Identity para designar um ou mais persistence fields ou propriedades

@GeneratedValue

@Transient -

@OneToMany -

@ManyToOne -

@ManyToMany -

@OneToOne -

@PersistenceContext -

Jws

@WebService - especifica que a Jws file implementa um web service

@WebMethod - especifica que o método é exposto como uma operação pública do webService Necessita de explicitar na anotação

@Get -

@Produces -

@Path -

@Context - especifica que o campo anotado permite acesso ao runtime context do WebSer...

JAX-RS é a implementação do REST

@Get - annotate your Get request methods

@Produces - especifica o tipo de output method a ser produzido (xml, json)

@Path - especifica o URL path no qual o método será invocado

@Post - annotate post request methods

@Delete - annotate delete request methods

@Put - annotate put request methods

Rabbit MQ Vs Kafka Vs Active MQ

> Kafka encontra-se preparado para lidar com streams (constante envio de dados e consumo)

> Kafka deve ser usado quando a quantidade de dados é enorme pois a sua eficiê... é maior.

> Rabbit Mq é mais fácil de implementar e deve ser usado para um número de mensa... gens pequenas e médias

> Active Mq - usar quando temos em pequenas e médias número de mensagens distribu... das