J931

Padrões de Projeto J2EE

1

Introdução

Helder da Rocha (helder@acm.org)

argonavis.com.br

Objetivos de aprender padrões J2EE

- *Conhecer padrões para uso na plataforma J2EE*
  - *Padrões permitem maior reuso, menos risco, etc.*
- *Evitar a reinvenção da roda ao desenvolver arquiteturas e design para a plataforma J2EE*
- *Usar melhores práticas para projetar aplicações que usam as tecnologias JSP, servlet, EJB e JMS*
- *Identificar problemas em práticas e designs existentes e ter objetivos para refatoramento*

2

Arquitetos J2EE precisam saber mais

- Não é suficiente saber como usar os recursos das principais APIs. É preciso entender questões como
  - Quais as melhores práticas
  - Quais as práticas não recomendadas?
  - Quais os problemas recorrentes mais comuns e que soluções provadas existem para eles?
  - Como refatorar código de um cenário abaixo do ideal para um melhor, descrito por um padrão?

3

J2EE Blueprints

- Procedimentos recomendados para desenvolver aplicações J2EE. Divide aplicações em camadas
  - Camada cliente: interface do usuário ou de serviços. Tipicamente representa uma aplicação independente ou browser rodando applets ou páginas HTML
  - Camada Web: consiste de servlets e páginas JSP com o objetivo de capturar requisições e processar respostas para a camada do cliente
  - Camada EJB: contém toda a lógica da aplicação e representa o modelo de negócio implementado em EJBs.
  - Camada de integração: contém lógica de acesso ao EIS
  - Camada de dados (EIS): consiste de sistemas de bancos de dados, transações e outros recursos legados

4

J2EE Patterns

- Soluções de design baseadas no J2EE Blueprints
  - Representam soluções consideradas melhores práticas para implementar vários componentes essenciais em cada uma das camadas identificadas pelo J2EE Blueprints
  - Usam e se baseiam em vários padrões GoF
- São padrões de design e não de implementação (idioms)
  - Podem ser implementados usando tecnologias diferentes (por exemplo, usando RMI ou CORBA)
- Objetivos
  - Reduzir o tráfego de rede, aumentando a eficiência e facilitando a escalabilidade
  - Reduzir o acoplamento entre as camadas e os componentes

5

SJC J2EE Patterns

- Não existe um só catálogo de padrões
  - Este curso se baseia no catálogo do Sun Java Center (SJC), cujos padrões estão documentados no site da Sun e em livros como "Core J2EE Patterns"
  - Os SJC J2EE Patterns são divididos em camadas lógicas que refletem a organização dos J2EE Blueprints
- O catálogo atual (setembro/2003) define 21 padrões
  - Camada de apresentação: 8 padrões
  - Camada de negócios: 9 padrões
  - Camada de integração: 4 padrões
- Os nomes dos padrões são significativos

6

Padrões vs. Estratégias

- *Padrões refletem soluções para problemas genéricos descritos em abstrações de alto nível*
- *Estratégias mostram formas de implementar os padrões usando tecnologias e linguagens de programação específicas*
- *Um padrão geralmente possui diversas diferentes estratégias de implementação*
- *Neste curso serão apresentados os padrões e suas principais estratégias recomendadas pelo SJC*

7

Quando usar padrões

- *Se possível, antes de iniciar a implementação*
  - *Padrões facilitam o desenvolvimento, estimulam o reuso, evitam código complexo e de difícil manutenção, facilitam a adoção de frameworks*
- *Se o sistema já existe, identifique as implementações inadequadas e más práticas*
  - *Partindo de uma implementação inadequada, aplique um padrão de refatoramento que tenha como objetivo uma arquitetura que use padrões*

8

Web: implementações inadequadas

- *Indicam necessidade de aplicação dos padrões, Os problemas mais comuns são*
  1. *Código de controle em vários Views*
  2. *Exposição de estruturas da camada de apresentação a outras camadas e objetos de domínio*
  3. *Permissão para requisições de formulário duplicadas (quando cliente faz reload no browser)*
  4. *Permissão de acesso do cliente a recursos sensíveis*
  5. *Permissão de inconsistência de propriedades <jsp:setProperty>*
  6. *Controladores gordos*
- *Várias técnicas de refatoramento* mostram como solucionar os problemas acima usando padrões J2EE*

* Veja Catálogo com Presentation Tier Bad Practices e Refactorings em [Core]

9

EJB: implementação inadequadas*

- *Indicam necessidade de aplicação dos padrões*
  1. *Mapeamento direto: modelo de objetos/modelo de Entity Beans*
  2. *Mapeamento direto: modelo relacional/modelo de Entity Beans*
  3. *Mapeamento de cada use case a um Session Bean*
  4. *Exposição de todos os atributos de um EJB via métodos get/set*
  5. *Embutir lookup de serviços nos clientes*
  6. *Usar Entity Beans como objetos read-only*
  7. *Usar Entity Beans como objetos fine-grained*
  8. *Armazenar árvore de objetos dependentes de Entity Beans*
  9. *Expor exceções relacionadas a EJB a clientes não-EJB*
  10. *Usar finder methods para retornar um grande result set*
  11. *Usar EJBs para transações longas*
  12. *Agregar dados de componentes de negócio no cliente*
  13. *Reconstruir estado por chamada em Stateless Session Bean*

* Veja Business Tier Bad Practices e Refactorings em [Core]

10

Padrões da Camada de Apresentação (1)

- *(1) Intercepting Filter*
  - *Viabiliza pré- e pós-processamento de requisições.*
- *(2) Front Controller*
  - *Oferece um controlador centralizado para gerenciar o processamento de uma requisição*
- *(3) Context Object*
  - *Encapsula estado de forma independente de protocolo para compartilhamento pela aplicação*
- *(4) Application Controller*
  - *Centraliza e modulariza gerenciamento de Views e de ações*

11

Padrões da Camada de Apresentação (2)

- *(5) View Helper*
  - *Encapsula lógica não-relacionada à formatação*
- *(6) Composite View*
  - *Cria uma View composta de componentes menores*
- *(7) Service To Worker e (8) Dispatcher View*
  - *Combinam Front Controller com um Dispatcher e Helpers. O primeiro concentra mais tarefas antes de despachar a requisição. O segundo realiza mais processamento depois*

12

Padrões da Camada de Negócios (1)

- *(9) Business Delegate*
  - *Desacopla camadas de apresentação e de serviços*
- *(10) Service Locator*
  - *Encapsula lógica de consulta e criação de objetos de serviço*
- *(11) Session Façade*
  - *Oculta complexidade de objetos de negócio e centraliza controle*
- *(12) Application Service*
  - *Centraliza e agrega comportamento para oferecer uma camada de serviços uniforme*
- *(13) Business Object*
  - *Separa dados de negócios e lógica usando modelo de objetos*

13

Padrões da Camada de Negócios (2)

- *(14) Composite Entity*
  - *Implementa Business Objects persistentes combinando Entity beans locais e POJOs**
- *(15) Transfer Object*
  - *Antigamente chamado de Value Object ou DTO*
  - *Reduz tráfego e facilita transferência de dados entre camadas*
- *(16) Transfer Object Assembler*
  - *Antigamente chamado de Value Object Assembler*
  - *Constrói um Value Object composto de múltiplas fontes*
- *(17) Value List Handler*
  - *Lida com execução de queries, caching de resultados, etc.*

* Plain Old Java Object

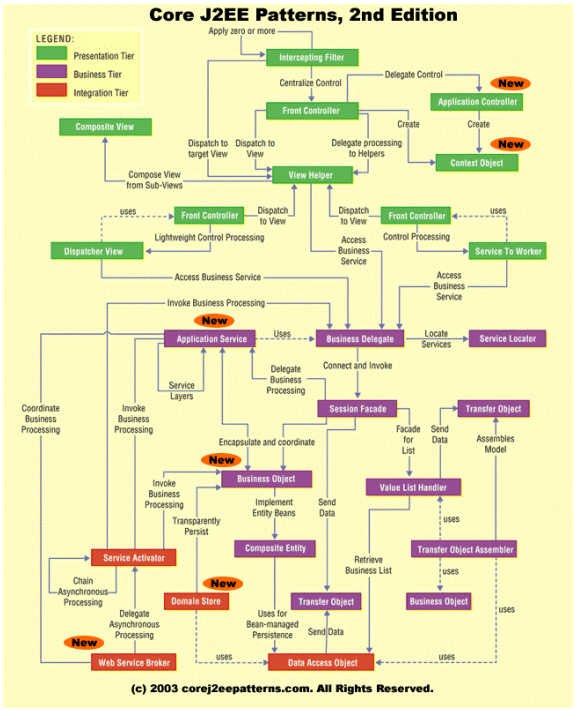
14

Padrões da Camada de Integração

- *(18) Data Access Object*
  - *Abstrai fontes de dados e oferece acesso transparente aos dados*
- *(19) Service Activator*
  - *Facilita o processamento assíncrono para componentes EJB*
- *(20) Domain Store*
  - *Oferece um mecanismo transparente de persistência para objetos de negócio*
- *(21) Web Service Broker*
  - *Expõe um ou mais serviços usando XML e protocolos Web*

15

Relacionamentos entre os padrões



Fonte: [Core]

16

Como usar os padrões?

- Procure no catálogo [SJC, Core] um padrão que realize o objetivo desejado
  - Tabela de padrões
  - Roadmap
- Roadmap associa intenção com padrões
  - "Se você procura por isto... use os padrões tais"
- Consulte o padrão desejado e analise
  - analise o problema que ele resolve
  - analise a solução que ele propõe
  - escolha uma estratégia e implemente

17

Micro-arquiteturas

- Relacionam vários padrões em conjunto em um nível de abstração mais alto
  - Representa uma solução completa para um problema genérico
  - Exemplos típicos são encontrados em frameworks

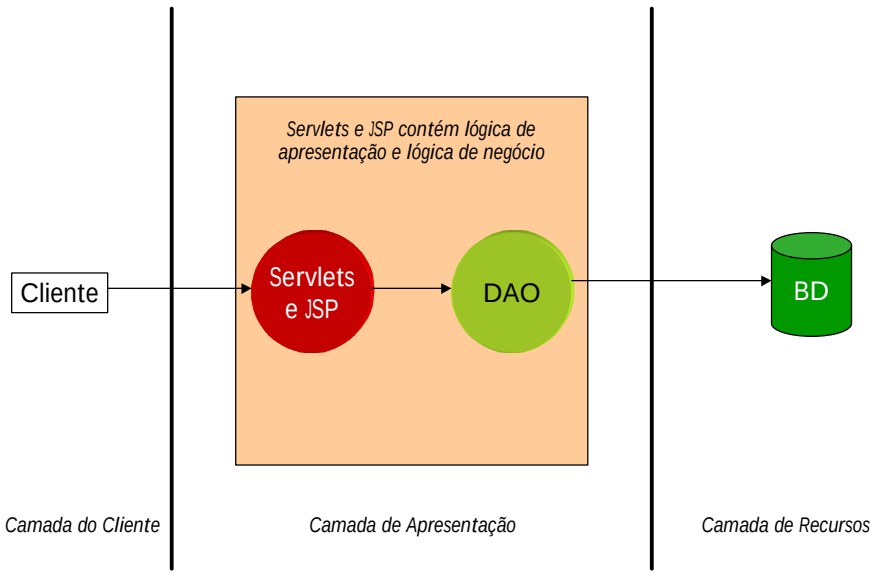
18

Refatoramento de arquitetura

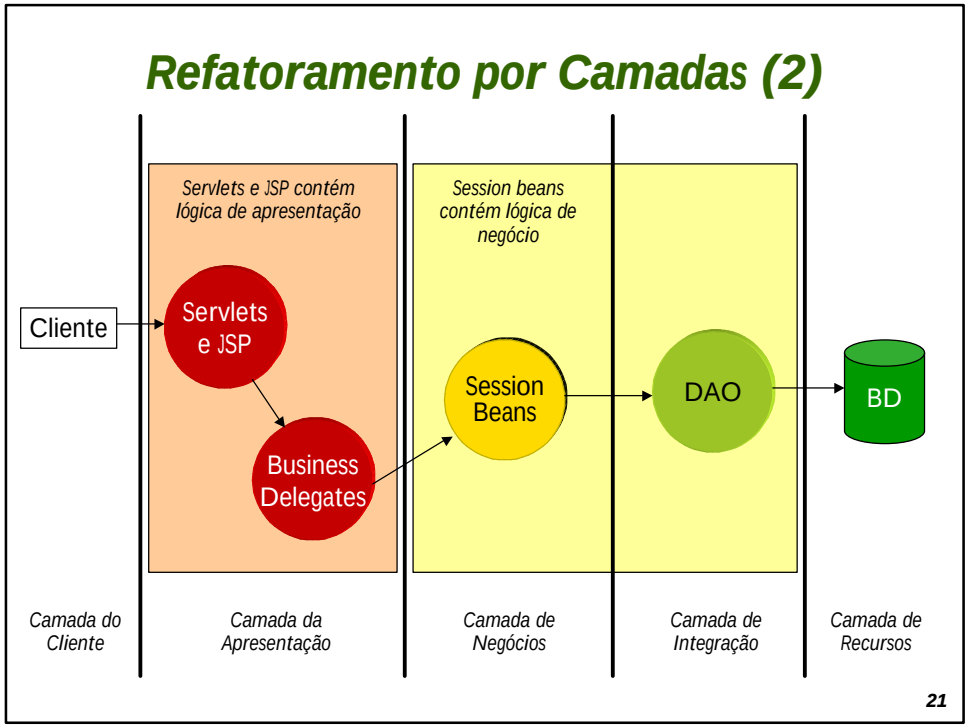
- *Padrões se encaixam bem quando é necessário alterar uma arquitetura para melhorar algum aspecto de um sistema*
  - *performance*
  - *escalabilidade*
  - *reuso e manutenção*
- *A maior parte dos padrões foram construídos visando esse tipo de evolução*

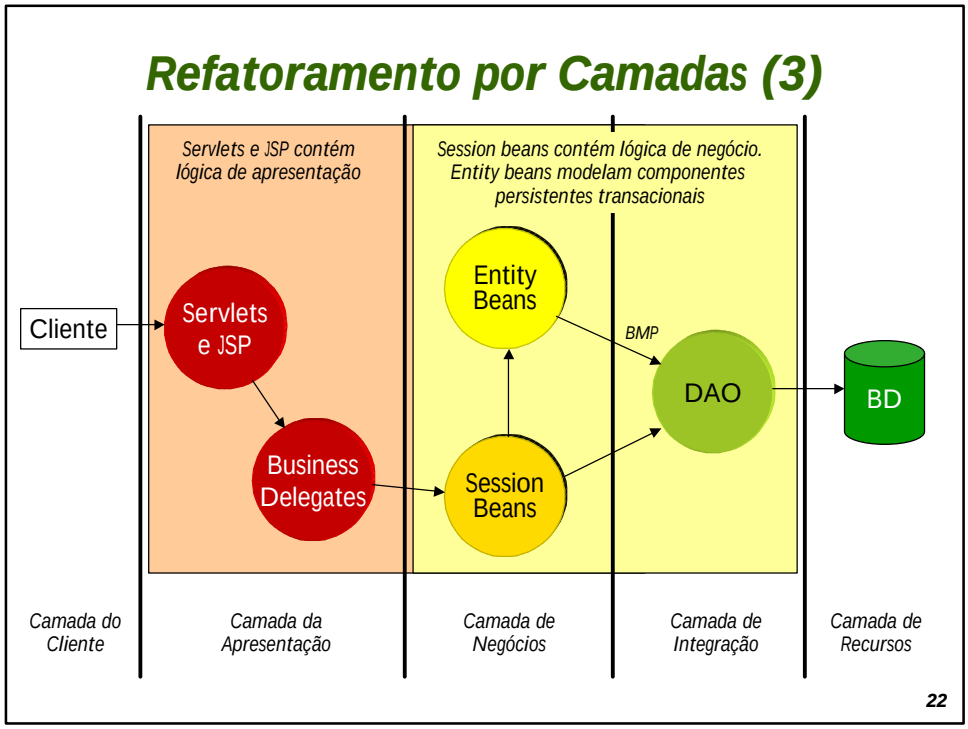
19

Refatoramento por Camadas (1)



20


Refatoramento por Camadas (2)
Servlets e JSP contém lógica de apresentação
Session beans contém lógica de negócio
Cliente
Servlets e JSP
Business Delegates
Session Beans
DAO
BD
Camada do Cliente
Camada da Apresentação
Camada de Negócios
Camada de Integração
Camada de Recursos
21


Refatoramento por Camadas (3)
Servlets e JSP contém lógica de apresentação
Session beans contém lógica de negócio. Entity beans modelam componentes persistentes transacionais
Cliente
Servlets e JSP
Business Delegates
Entity Beans
BMP
Session Beans
DAO
BD
Camada do Cliente
Camada da Apresentação
Camada de Negócios
Camada de Integração
Camada de Recursos
22

Fontes

[SJC] SJC Sun Java Center J2EE Patterns Catalog.
http://developer.java.sun.com/developer/restricted/patterns/J2EEPatternsAtAGlance.html.

[Blueprints] J2EE Blueprints patterns Catalog.
http://java.sun.com/blueprints/patterns/catalog.htm.

[Core] Deepak Alur, John Crupi, Dan Malks. Core J2EE Patterns: Best Practices and Design Strategies. Prentice-Hall, 2001.
http://java.sun.com/blueprints/corej2eepatterns/index.html.

23

Curso J931: J2EE Design Patterns

Versão 1.1

www.argonavis.com.br

(helder@acm.org)