Persistência de Objetos

Introdução

Última atualização: 22 de abril de 2008

Prof. João Eduardo Ferreira
Prof. Osvaldo Kotaro Takai

Definições

- Em linguagens de programação OO:
  - **Objeto** é uma instância de uma classe.
    - Objetos têm **estados** (os valores de seus atributos).
    - Objetos têm **comportamentos** (seus métodos).
  - **Modelo de objetos** é uma coleção de todas as definições de classes de uma aplicação.
  - As classes podem representar:
    - Elementos de interfaces do usuário.
    - Recursos do sistema.
    - Eventos da aplicação.
    - Abstrações dos conceitos de negócio.
      - Nomes significativos para pessoas de negócio não-técnicas.

Definições

- Objetos que abstraem conceitos de negócio:
  - Num sistema de processamento de pedidos:
    - Cliente, Pedido e Produto.
  - Numa aplicação financeira:
    - Cliente, Conta, Crédito, Débito.
- Esses objetos modelam o domínio do negócio onde a aplicação específica irá operar.
- Assim, eles são coletivamente chamados de **Modelo de Objetos de Domínio**.

Definições

- Os objetos Modelo de Objetos de Domínio:
  - Representam os principais estados e comportamentos da aplicação.
  - Normalmente:
    - são compartilhados por vários usuários simultaneamente.
    - são armazenados e recuperados entre as execuções da aplicação.
  - A capacidade desses objetos de sobreviverem além do tempo de execução da aplicação é chamada de **Persistência de Objetos**.

Definições

- A persistência de objetos não é exclusividade dos objetos do Modelo de Objetos de Domínio:
  - Por exemplo, um objeto da classe LogDeMensagens pode ter esta característica.
- Mas a maioria dos objetos persistentes de uma aplicação encontram-se no Modelo de Objetos de Domínio.

Repositórios para Persistência

- A persistência precisa armazenar o estado dos objetos em algum repositório para futuramente recuperá-los.
- Os repositórios podem ser:
  - Um BD relacional (mais comum).
  - Arquivos do sistema.
  - Um BD OO.

Técnicas de Persistência

Obj/Rel

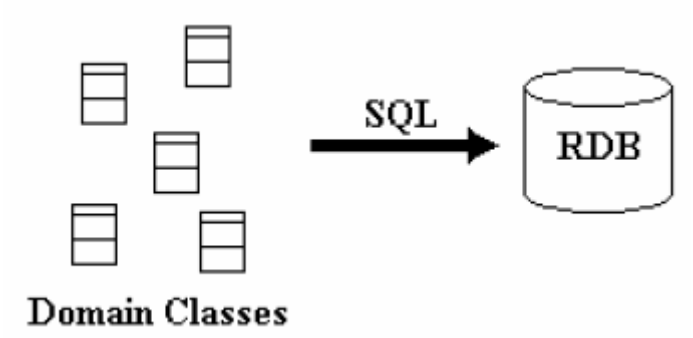
Técnicas de Persistência – Obj/Rel

- A maioria dos projetos de desenvolvimento de software utiliza:
  - a linguagem OO, tais como Java e C#.
  - o BD relacional para armazenar dados.
- O desenvolvimento de aplicações que usam linguagens OO e DB Relacional enfrentam o problema da **incompatibilidade conceitual** (*impedance mismatch*).
- Para superar este problema é importante conhecer:
  - O processo de mapeamento objeto-relacional.
  - Como implementar mapeamento objeto-relacional.

Técnicas de Persistência - Força Bruta

- Força Bruta (mais comum):
  - o código SQL é embutido no código-fonte das classes.
  - Vantagens:
    - permite escrever códigos muito rapidamente
    - viável para pequenas aplicações ou protótipos.
  - Desvantagens:
    - Alto acoplamento entre as classes de domínio e o esquema do banco de dados.
    - Mudanças simples no BD (por exemplo, renomear uma coluna) resulta na manutenção do código-fonte OO.

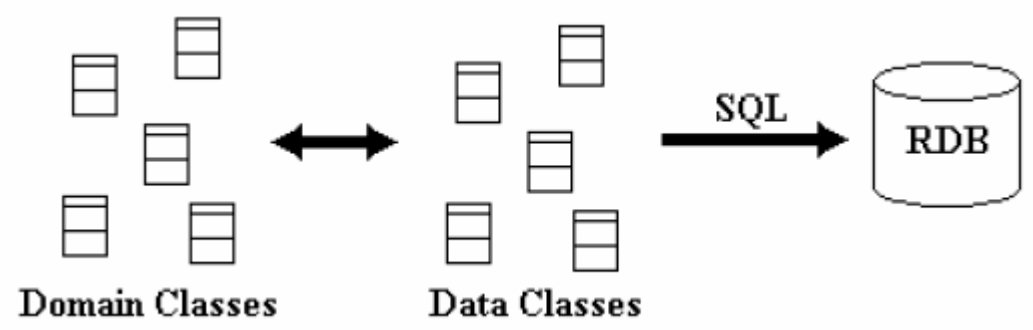
Técnicas de Persistência - Força Bruta



Técnicas de Persistência – Classes de Dados

- Classes de Dados:
  - SQL das classes de domínio são encapsuladas nas "classes de dados". Exemplos:
    - SP's no BD representam objetos (substituindo as classes de dados)
    - ActiveX Data Objects (ADO).
  - Vantagens:
    - Viável para protótipos e pequenos sistemas (40 classes de negócio).
  - Desvantagens:
    - Recompilação das classes de dados quando pequenas mudanças são feitas no BD.

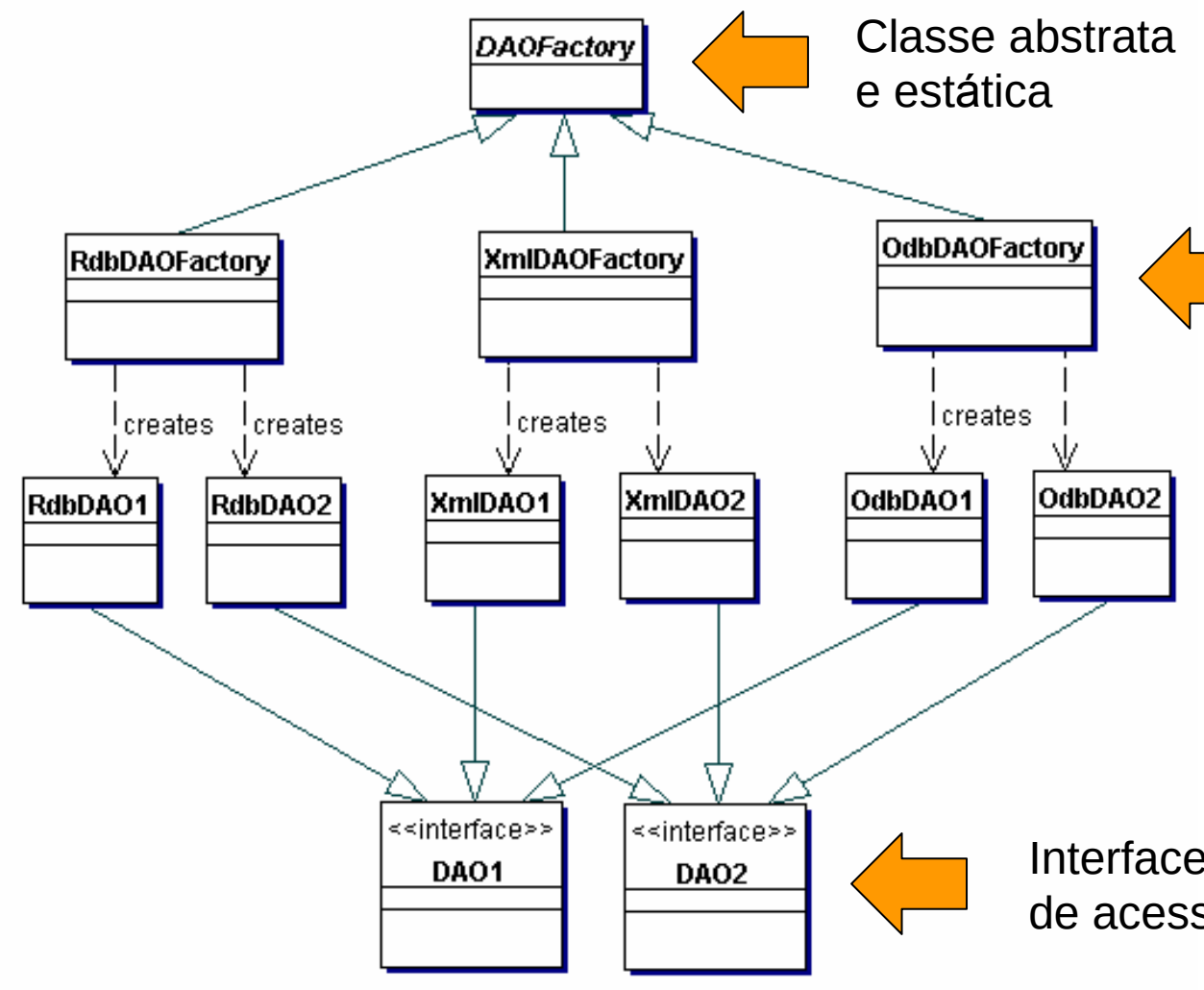
Técnicas de Persistência – Classes de Dados



Técnicas de Persistência – Classes de Dados
O Padrão DAO

- Este padrão permite criar as classes de dados independentemente da fonte de dados ser um BD relacional.
- Para isso, encapsula os mecanismos de acesso a dados e cria uma interface de cliente genérica para acessar os dados.
- Dessa forma, o DAO permite que os mecanismos de acesso a dados mudem independentemente do código que usa o dado.

Técnicas de Persistência – Classes de Dados
O Padrão DAO



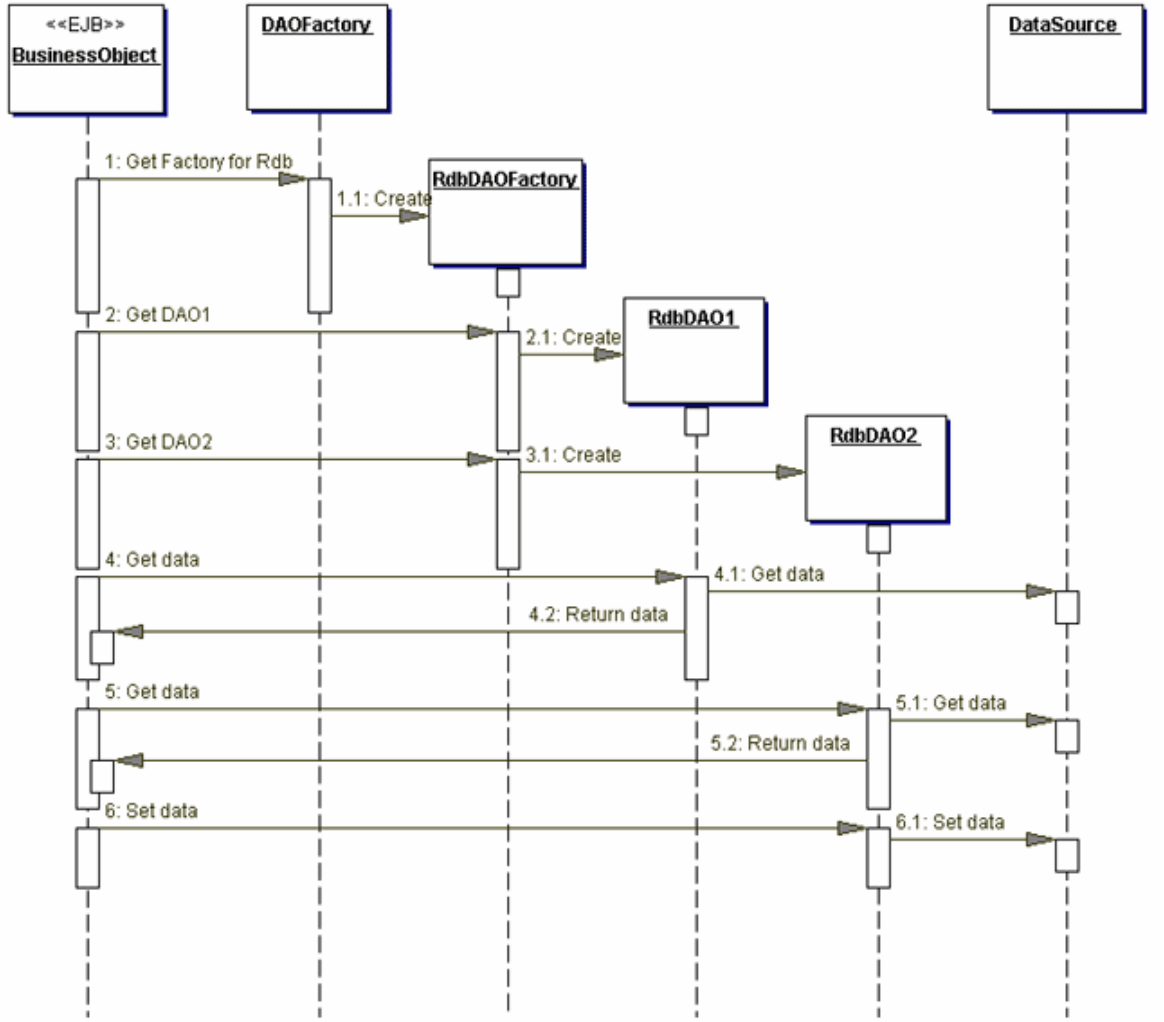
Classe abstrata
e estática

Fábrica de objetos
de acesso a dados
específico

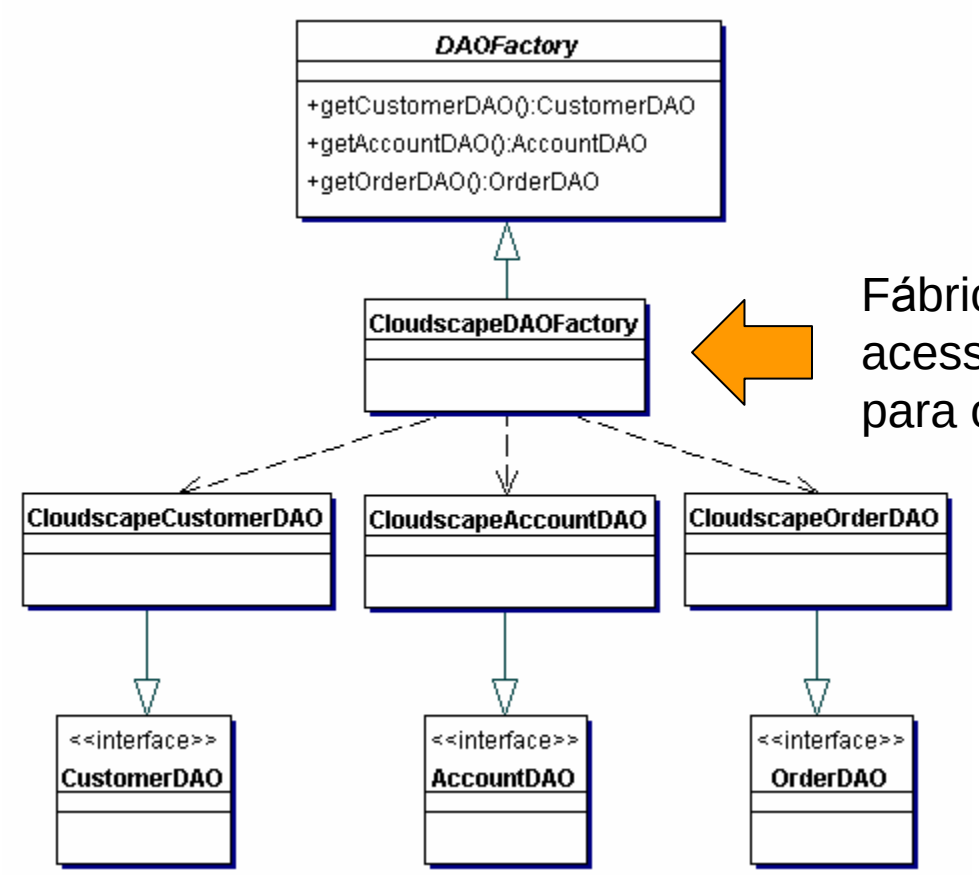
Classes de
acesso a dados

Interface padrão
de acesso

Técnicas de Persistência – Classes de Dados
O Padrão DAO



Técnicas de Persistência – Classes de Dados
O Padrão DAO



Fábrica de objetos de acesso a dados específico para o BD Cloudscape

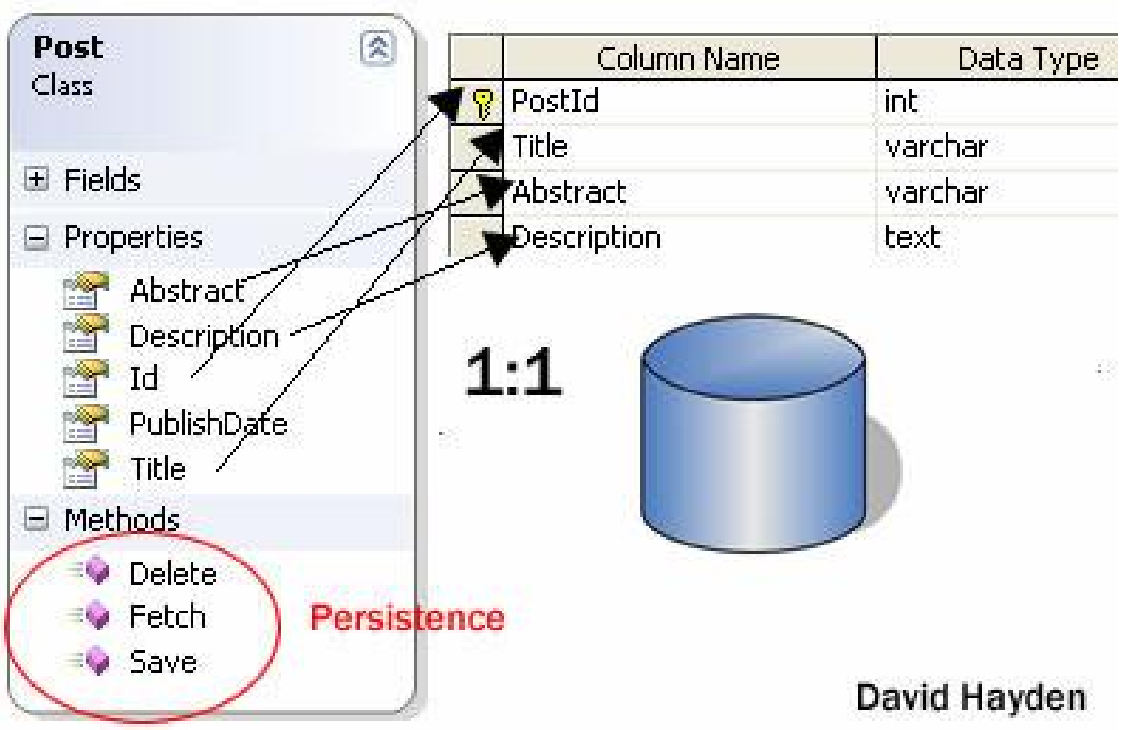
Técnicas de Persistência – Classes de Dados
O Padrão DAO

- Estratégia para desenvolvimento rápido usando o padrão DAO com BD relacionais:
  - Gerador de DAOs
    - http://www.akcess.in/download.html
    - http://www.codefutures.com/products/firestorm/

Técnicas de Persistência – Classes de Dados
O Padrão Active Record

- Objeto de negócio ⇔ linha de tabela ou visão
- Encapsula o acesso ao BD
- Lógica de domínio ∈ Objetos de negócio
- Ideal quando:
  - existirem poucas regras de negócio
  - for possível manter relacionamento 1:1 entre propriedades do objeto de negócio com colunas de uma tabela
  - não houver necessidade de mapeamento adicional

Técnicas de Persistência – Classes de Dados
O Padrão Active Record



Técnicas de Persistência – Classes de Dados
O Padrão Active Record

- Com o esse padrão, focamos a persistência como uma responsabilidade ao invés de serviço.
- Operações CRUD são mapeados para métodos de instância estáticos do objeto de domínio.

Técnicas de Persistência – Classes de Dados
O Padrão Active Record

- Gerador de Active Record
  - http://subsonicproject.com/
- API - jPersist
  - http://www.jwebapp.org/

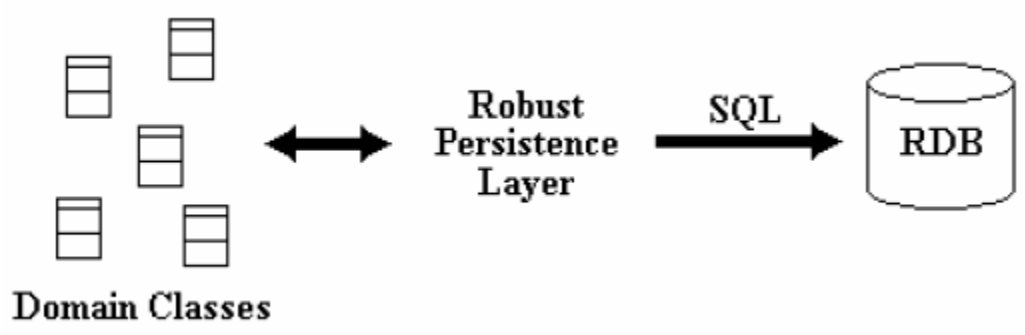
Técnicas de Persistência

Framework

Técnicas de Persistência – Framework

- Camada de Persistência Robusta (framework):
  - O framework mapeia objetos para bancos de dados relacionais, de maneira que as pequenas mudanças no esquema relacional não afetem o código orientado a objetos.
  - Vantagens:
    - Programadores não precisam conhecer nada a respeito do esquema do BD relacional.
    - Permite desenvolvimento de aplicações em larga escala.
  - Desvantagem:
    - Impacto no desempenho das aplicações.
    - Propicia degeneração conceitual com impacto negativo no modelo de banco de dados.

Técnicas de Persistência – Framework



Java Data Objects

Persistência com JDO

Prof. João Eduardo Ferreira
Prof. Osvaldo Kotaro Takai

Última atualização: 22 de abril de 2005

O que é JDO?

- O JDO (Java Data Objects) é um padrão para persistência transparente para objetos Java definida pela JSR-000012/JCP.
  - Fornece aos desenvolvedores uma visão de persistência e acesso ao repositório de dados centrada em objetos Java.
- Atualmente na versão 2.0.
- Permitir “plugar” drivers de diferentes fornecedores para acessar qualquer DB/repositório de dados.
- Trabalha em conjunto com *Application Servers*.

As Metas do JDO

- Persistência Transparente de Objetos
  - Reduzir para zero o número de restrições para construir classes
  - Nenhuma nova linguagem de acesso a dados
    - Java é a DDL & DML
- Disponibilidade para criar várias implementações
  - J2ME - Embedded, device-oriented
  - J2SE - Client/server
  - J2EE - Enterprise Java Beans
- Independência do repositório de dados
  - Relacional, objeto, objeto-relacional, hierárquico, sistema de arquivos, etc.

Audiência do JDO

- Desenvolvedores de aplicações Java
  - Persistência Transparente de Objetos
  - Centrado em Java, não necessitando conhecer como acessar um BD
- Desenvolvedores de aplicação EJB
  - Gerenciamento de transação e de conexões via Application Server
  - Acesso ao BD transparente para soluções não CMP (Session Beans & BMP)
    - Não há necessidade de usar diretamente o JDBC
    - OQL para encontrar instâncias

Comparação entre JDO, Seriação e JDBC

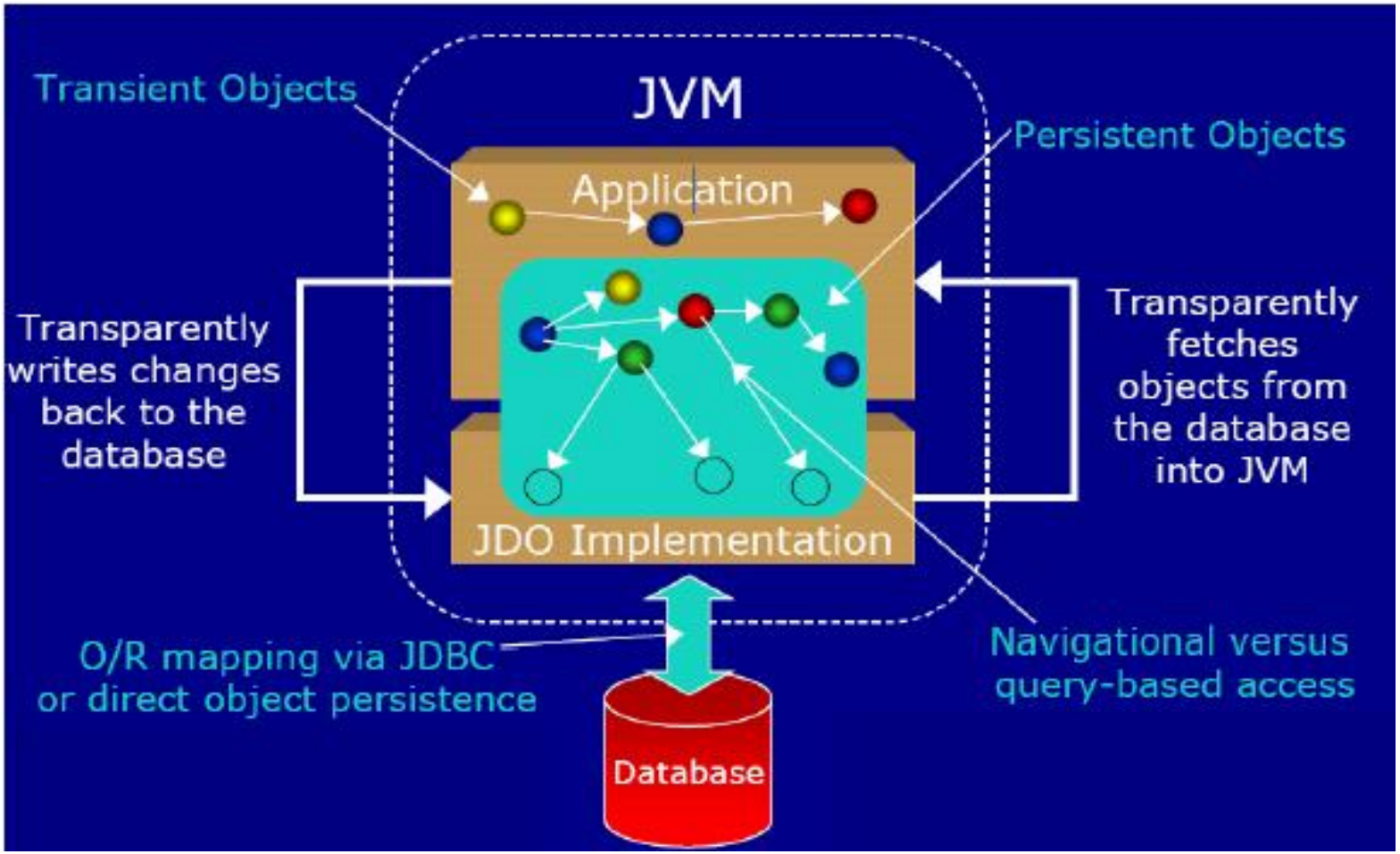
Feature	Serialization	JDBC	JDO
Data Model	Java	Relational table model	Java
Support of Java Classes	✔	✘	✔
Access granularity	Object Graph	Table cell	Object
Support of inheritance and polymorphism	✔	✘	✔
Support of references and collections	✔	✘	✔
Unique identity	✘	Primary key	Application or Datastore Identity
Automatic management of cache	✘	✘	✔
Transactions	✘	✔	✔
Concurrency	✘	✔	✔
Query Language	✘	SQL, each vendor has a different dialect (not portable)	JDOQL, standard language, Java-like syntax
Object model supported in queries	✘	✘	✔
JCA compatibility for application server integration	✘	JDBC 3.0, 4.0	✔

Fonte: Comparação entre JDO, Seriação e JDBC (JDOCentral.com)

Como o JDO Funciona?

- O JDO permite que as aplicações armazenem e recuperem “transparentemente” instâncias de qualquer classe Java de um repositório de dados.
- Para a aplicação, os “objetos persistentes” se parecem como objetos Java normais, residentes na memória.
  - Os campos dessas instâncias na realidade estão armazenadas em algum repositório de dados, persistentemente – mas sem qualquer ação explícita da aplicação.
- O JDO não precisa saber onde os métodos são executados.
  - Ele não possui meios de chamar remotamente, como no RMI ou EJB, métodos de objetos em repositórios de dados.

Ambiente de Execução JDO

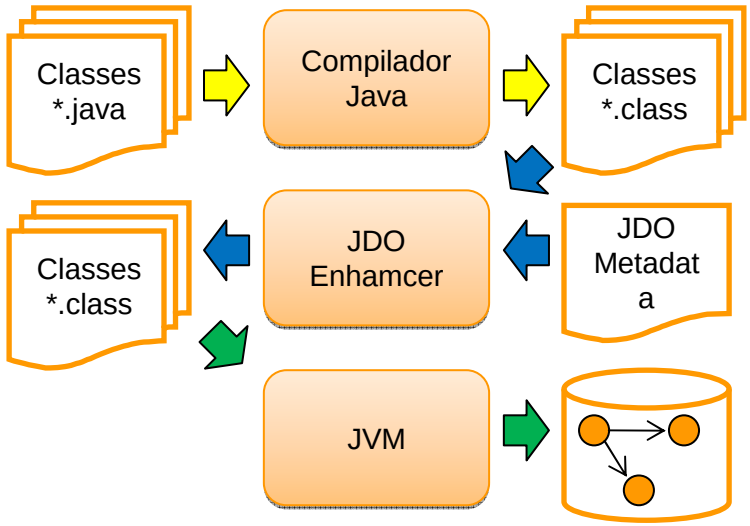


Responsabilidades do Desenvolvedor

- Determinar quais objetos do modelo de domínio precisam ser persistidos.
  - Marcar essas classes como ‘capazes de persistência’ (*persistence capable)*
  - Usar a API JDO para persistir instâncias dessas classes.

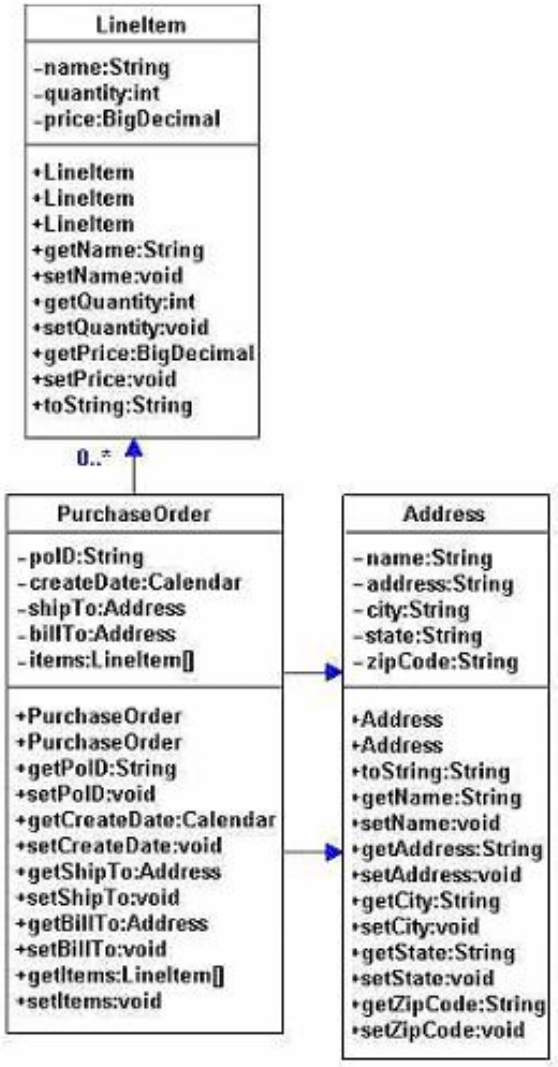
O Ciclo de Desenvolvimento JDO

- Qualquer classe que implemente **PersistenceCapable** pode ser persistida.



- Três maneiras de criar tais classes
  - Geração do código-fonte
  - Pré-processamento do código-fonte
  - Bytecode enhancement
    - O Enhancement é um passo de pós-compilação
    - Completamente transparente ao desenvolvedor

Um exemplo rápido

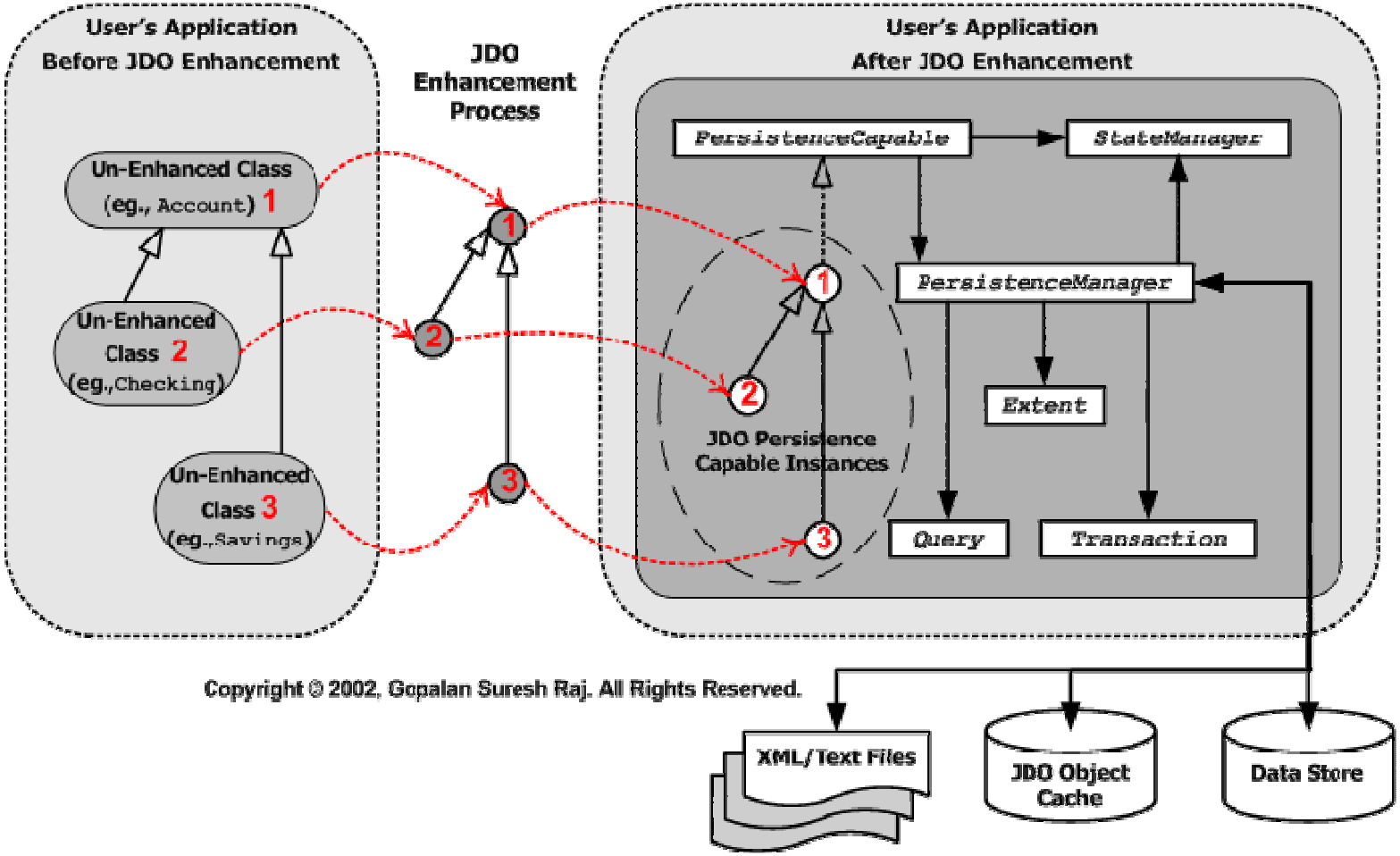



```
Properties p= new Properties();
// set some properties
PersistenceManagerFactory pmf= JDOHelper.getPersistenceManagerFactory(properties);
PersistenceManager pm =pmf.getPersistenceManager();
Transaction tx = pm.currentTransaction();
tx.begin();

Address addrs = new Address("1 Main Street“,  "Beverly Hills", "CA", "90210");

LineItem items[] = new LineItem[] {
   new LineItem("Copier Paper", new BigDecimal(10.00), 2)
};

PurchaseOrder order = new PurchaseOrder();

order.setCreateDate(Calendar.getInstance());
order.setBillTo(addrs);
order.setShipTo(addrs);
order.setItems(items);
order.setPoID("ABC-CO-19282");

pm.makePersistent(order);
tx.commit();
pm.close();
```

O que o JDO Enhancer faz?

- Lê o byte code e gera num novo byte code
  - Adiciona “ganchos” que permitem ao JDO transparentemente:
    - Recuperar de objetos
    - Rastrear mudanças nos estados de objetos
    - Escrever mudanças no repositório de dados quando confirmado
  - O Desenvolvedor não necessita fazer a busca e armazenamento explicito de objetos

Visão Geral


User's Application
Before JDO Enhancement
Un-Enhanced Class
(eg., Account) 1
Un-Enhanced
Class 2
(eg.,Checking)
Un-Enhanced
Class 3
(eg.,Savings)
JDO
Enhancement
Process
1
2
3
User's Application
After JDO Enhancement
PersistenceCapable
StateManager
PersistenceManager
Extent
Query
Transaction
JDO Persistence
Capable Instances
Copyright © 2002, Gopalan Suresh Raj. All Rights Reserved.
XML/Text Files
JDO Object
Cache
Data Store

Exemplos de implementação JDO

ObjectDB

- O ObjectDB é um GBDOO que implementa o JDO.
- Recomendado para aprender o JDO, pois não existe a necessidade de pensar no mapeamento objeto-relacional.
- Site: http://www.objectdb.com
- Existe uma versão para Download free.
- Link para o tutorial.

JPOX

- É uma iniciativa OpenSource, agora responsável pela implementação de referência do JDO 2.0.
- JPOX permite realizar o mapeamento objeto-relacional compatível com a maioria dos SGBD Relacional via JDOC.
- Site: http://www.jpox.org/
- Download da versão compatível com JDO 2.0.
- Plugin JPOX para Eclipse.
- Link para o tutorial.

Conclusão

- Modelagem de dados
- Dados
- Processos
- Delimitaçao dados x processos

FIM