

---

Exercício de aprendizado de configuração e  
utilização do gerador de aplicação Captor

*Edison Kicho Shimabukuro Junior*

---

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Persistência de dados</b>	<b>1</b>
<b>3</b>	<b>Configuração da ferramenta</b>	<b>2</b>
3.1	Criação da linguagem de modelagem de aplicações e desenvolvimento do arquivo de configuração principal . . . . .	3
3.1.1	Definição da estrutura de formulários . . . . .	4
3.1.2	Seleção dos componentes . . . . .	5
3.1.3	Definição das regras de validação . . . . .	6
3.2	Desenvolvimento dos arquivos matrizes e templates . . . . .	7
3.2.1	Matriz SQL . . . . .	7
3.2.2	Matriz de classe persistente (Java - JDBC) . . . . .	8
3.3	Desenvolvimento do arquivo de mapeamento de transformação de templates	12
3.4	Exercício 1: Configuração da ferramenta . . . . .	12
<b>4</b>	<b>Utilização da ferramenta</b>	<b>13</b>
4.1	Exercício 2 - Utilização da ferramenta . . . . .	13
	<b>Referências</b>	<b>15</b>

# 1 Introdução

Este documento contém a descrição de um exercício utilizado no treinamento de capacitação técnica da utilização da ferramenta Captor. Esse exercício descreve as principais etapas que antecedem a configuração da ferramenta e os principais artefatos que devem ser criados para desenvolver e implantar uma nova configuração na ferramenta.

A leitura deste documento tem os seguintes pré-requisitos:

- Noções básicas da linguagem de programação Java.
- Noções básicas de banco de dados relacionais.
- Noções básicas da linguagem de transformação de *templates* XSL.
- Leitura e entendimento do manual de configuração e utilização da ferramenta Captor.

O exercício proposto neste documento tem o objetivo de configurar a ferramenta para a geração de classes persistentes Java e *scripts* de criação de tabelas SQL.

Na Seção 2 são apresentados os principais conceitos da persistência de dados e as principais tecnologias de persistência relacionadas com a linguagem de programação Java. Na Seção 3 são apresentadas as principais etapas do processo configuração da ferramenta Captor. Na Seção 4 são apresentadas as principais etapas da utilização da ferramenta Captor. Nas Sub-seções 3.4 e 4.1 são apresentados os enunciados da parte 1 e 2 do exercício proposto.

## 2 Persistência de dados

Muitas aplicações utilizam a persistência de dados para armazenar informações em um dispositivo de *hardware* com a finalidade de recuperar essas informações em um momento determinado.

A linguagem de programação Java oferece diversas maneiras de programar a persistência de objetos, dentre elas estão a interface de programação de aplicações (API) JDBC, o framework de persistência Hibernate (Hibernate, 2005) e o framework de persistência OJB (OJB, 2005).

Os exercícios apresentados neste documento utilizam a API JDBC. Essa API foi utilizada por estar disponível na biblioteca padrão da linguagem Java e possuir uma implementação mais simples em relação a abordagem de frameworks de persistência.

A tecnologia JDBC fornece conectividade com os principais sistemas gerenciadores de banco de dados utilizados no mercado. Essa API permite que o desenvolvedor utilize três recursos:

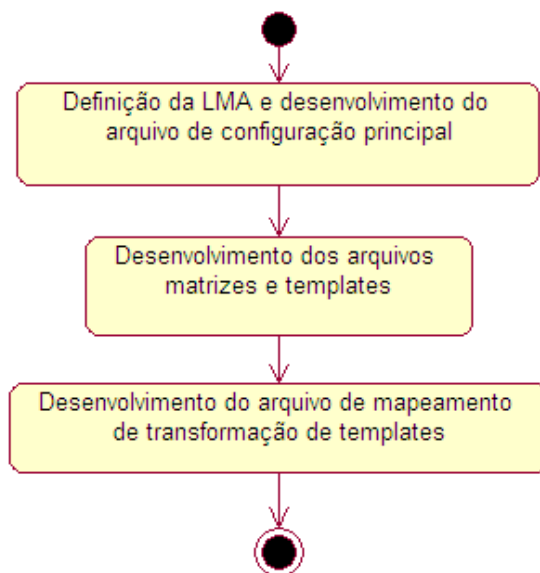
- Estabelecer uma conexão com um banco de dados.
- Enviar comandos SQL para o banco de dados.
- Processar os resultados.

Para maiores informações sobre a API JDBC utilize o link:

<http://java.sun.com/products/jdbc/index.jsp>.

### 3 Configuração da ferramenta

A realização da configuração da ferramenta Captor tem o objetivo de especializar a ferramenta para um domínio específico e gerar artefatos nesse domínio. Neste exercício, a ferramenta deve gerar scripts SQL e classes Java que realizam a persistência de dados em tabelas relacionais. Na Figura 1 é apresentado o processo de configuração da ferramenta Captor.



**Figura 1: Processo de configuração da ferramenta Captor**

Na primeira etapa da configuração da ferramenta deve ser definido a linguagem de modelagem de aplicações (LMA) e baseado nessa linguagem, deve ser desenvolvido o arquivo de configuração principal contendo as definições dos formulários e das regras de validação da especificação. Na segunda etapa são desenvolvidos os arquivos matrizes e os templates. Na terceira etapa, o arquivo de mapeamento de transformação de templates é desenvolvido para indicar para a ferramenta quais templates devem ser utilizados no processo de geração de artefatos.

### 3.1 Criação da linguagem de modelagem de aplicações e desenvolvimento do arquivo de configuração principal

Para gerar uma aplicação em um domínio particular, é necessário que seja modelado uma linguagem de especificação de aplicações<sup>1</sup>. Essa linguagem é utilizada para descrever aplicações nesse domínio e a sua estrutura deve expressar os parâmetros de variação que as aplicações nesse domínio podem apresentar.

No domínio de persistência, uma classe representa uma tabela do banco de dados e os atributos dessa classe representam uma tupla dessa tabela. As classes persistentes devem fornecer um método para inserir uma tupla em uma tabela, um método para recuperar uma tupla de uma tabela e um método para remover uma tupla de uma tabela.

Os parâmetros de variação necessários para especificar classes persistentes são apresentados na Tabela 1.

Tabela 1: Parâmetros de variação

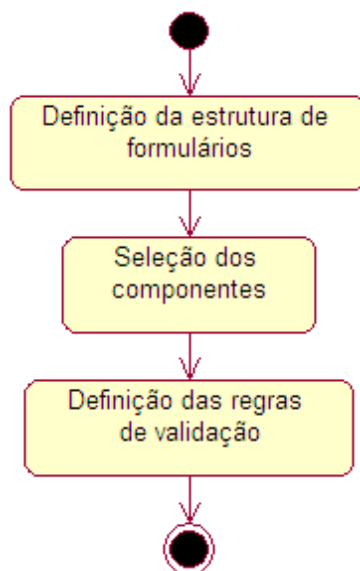
Parâmetro	Descrição
Nome do pacote	O nome do pacote em que as classes devem ser armazenadas.
Nome da classe	O nome da classe persistente que deve ser criada.
Nome da tabela	O nome da tabela do banco de dados que essa classe representa.
Atributos da classe	O nome dos atributos da classe persistente.
Atributos da tabela	O nome dos atributos da tabela do banco de dados.
Tipos dos atributos	O tipo dos atributos da classe e da tabela.

Para simplificar este exercício, uma classe não pode ter associações com outras classes e os tipos válidos dos atributos das classes e das tabelas são apenas os tipos inteiros e *strings*. Como exercício complementar, o leitor pode realizar a configuração completa da ferramenta para o domínio de persistência incluindo diversos tipos de relacionamentos entre classes e apoio para os principais tipos de dados disponíveis nos banco de dados comerciais.

Note que a especificação desses parâmetros de variação disponíveis na Tabela 1 não são baseadas na linguagem de programação Java ou na interface de programação JDBC. O objetivo da criação e utilização dessa linguagem é permitir que o engenheiro de aplicação especifique uma aplicação de forma abstrata e obtenha a diminuição do esforço e tempo de desenvolvimento de aplicações nesse domínio por meio da geração dos artefatos.

<sup>1</sup>Como entrada para o processo de modelagem da linguagem de aplicações, são necessários os artefatos da análise de domínio e projeto modular (Weiss e Lai, 1999). A persistência de dados em banco de dados relacionais é uma atividade recorrente durante a programação de aplicações, e por esse motivo, este manual pretende utilizar o conhecimento do leitor no domínio escolhido e omitir as atividades de análise de domínio e projeto modular.

Para iniciar a parte técnica do processo de configuração, é necessário a criação do arquivo de configuração principal. O arquivo de configuração principal descreve a estrutura dos formulários da especificação, os componentes gráficos que cada formulário contém e as regras de validação dessa especificação. Na Figura 2 são apresentadas as principais etapas do processo de desenvolvimento do arquivo de configuração principal.



**Figura 2: Processo de desenvolvimento da LMA**

### 3.1.1 Definição da estrutura de formulários

Os formulários são estruturados na ferramenta em forma de árvore. A especificação da aplicação começa com a edição de um formulário raiz que pode conter um ou mais formulários filhos.

Baseado no conhecimento de domínio e no conhecimento funcional da ferramenta, pode ser definida a seguinte estrutura de formulários para armazenar os parâmetros de variação contidos na Tabela 1:

Para o nó raiz da especificação, pode ser definido um formulário com uma caixa de texto contendo a descrição das classes de persistência que estão sendo geradas. As classes persistentes são agrupadas em pacotes. Os pacotes são especificados no nós filhos do formulário raiz. As classes persistentes são especificadas nos formulários filhos do formulário da especificação de pacotes. Os formulários que representam as classes persistentes devem conter um campo para especificar o nome da classe, um campo para especificar o nome da tabela que essa classe representa e um componente gráfico em forma de tabela para descrever o nome e tipo dos atributos das classes e da tabela.

Um formulário de descrição de classes pode ter um ou mais nós filhos com descrição de pacotes e os formulários de especificação de pacotes podem conter um ou mais nós filhos com os formulários da especificação de classes.

Na Figura 3 é apresentado essa estrutura de formulários e as Figuras 4, 5 e 6 apresentam esquematicamente os componentes gráficos de cada formulário da Figura 3.

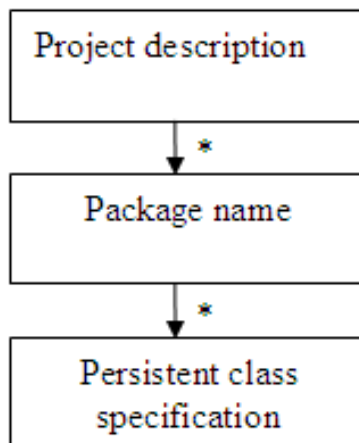


Figura 3: Estrutura de formulários

Figura 4: Formulário de especificação da descrição do projeto

### 3.1.2 Seleção dos componentes

Para montar os formulários definidos na Sub-seção 3.1.1 é necessário escolher os componentes de formulário que cada formulário deve conter. Baseado nas Figuras 4, 5 e 6 e no manual de componentes de formulário, o engenheiro de domínio pode escolher os seguintes componentes:

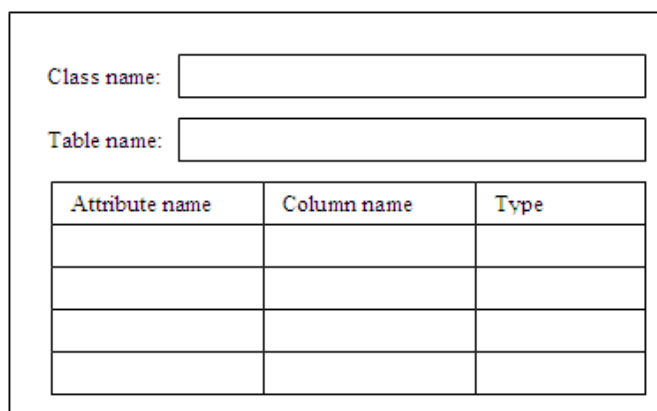
#### Formulário de descrição de projetos

- O componente de formulário `TextAreaPanel` para definir a descrição do projeto.



Package name:

**Figura 5: Formulário de especificação de pacotes**



Class name:

Table name:

Attribute name	Column name	Type
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>

**Figura 6: Formulário de especificação de classes persistentes**

#### Formulário de especificação de pacotes:

- O componente de formulário `TextPanel` para definir o nome dos pacotes.

#### Formulário de especificação de classes:

- O componente de formulário `TextPanel` para definir o nome da classe.
- O componente de formulário `TextPanel` para definir o nome da tabela.
- O componente de formulário `TablePanel` para definir o nome dos atributos da classe e das tabelas, e os tipos desses atributos.

#### 3.1.3 Definição das regras de validação

O engenheiro de domínio deve configurar a ferramenta para fazer a validação da especificação da aplicação. A definição das regras de validação tem o objetivo de não permitir que o engenheiro de aplicação gere artefatos à partir de uma especificação inválida.

A ferramenta `Captor` fornece duas formas de validação: a validação estrutural e a validação sintática.



A validação estrutural deve assegurar que não exista nenhum formulário de especificação de pacotes que não contenha pelo menos um formulário filho com a especificação de uma classe e a validação sintática deve ser aplicada nos formulários de especificação de pacotes e nos formulários de especificação de classes. As regras de validação sintática dos dados inseridos nos componentes de formulário são apresentados na Tabela 2.

Tabela 2: Regras de validação

Parâmetro	Descrição
Nome do pacote	O nome do pacote deve ser especificado de acordo com a expressão regular: $([a-zA-Z]+[a-zA-Z\_-0-9]+)([\ \backslash\ /][a-zA-Z]+[a-zA-Z\_-0-9]+)^*$
Nome da classe	O nome da classe deve ser especificada de acordo com a expressão regular: $[A-Z]+([a-zA-Z\_-0-9]+)^*[A-Za-z]+$
Nome da tabela	O nome da tabela deve ser especificada de acordo com a expressão regular: $[A-Za-z]+([a-zA-Z\_-0-9]+)^*[A-Za-z]+$
Atributos da classe	O nome dos atributos da classe devem ser especificados de acordo com a expressão regular: $[a-z]+([a-zA-Z\_-0-9]+)^*[A-Za-z]+$
Colunas da tabela	O nome das colunas da tabela devem ser especificados de acordo com a expressão regular: $[A-Z]+([a-zA-Z\_-0-9]+)^*[A-Za-z]+$
Tipos dos atributos	Os tipos de dados devem ser especificados de acordo com a expressão regular: $[int]^*[String]^*$

### 3.2 Desenvolvimento dos arquivos matrizes e templates

Para iniciar o desenvolvimento de *templates* é necessário que exista pelo menos um exemplo de código desenvolvido manualmente. À partir desse exemplo, chamado de exemplo matriz, os *templates* são desenvolvidos.

#### 3.2.1 Matriz SQL

Na Listagem 1 é apresentado o código SQL necessário para realizar a criação de uma tabela no banco de dados MySQL (MySQL, 2005).

Listing 1: Código SQL necessário para criar uma tabela do banco de dados

```
1 CREATE TABLE NOME.DA.TABELA
  (att1 int , att2 int , att3 varchar(50));
```

### 3.2.2 Matriz de classe persistente (Java - JDBC)

A Matriz dos código Java que realiza a persistência de uma classe Java é apresentada na Listagem 2.

Listing 2: Código Java de uma classe persistente

```
package exemplo.persistente;
2
import java.sql.*;
4
public class Pessoa {
6
    private int id;
8    private String nome;
    private int idade;
10   private boolean sexo;

12   private Connection con;

14   public Pessoa(Connection con) {
        nome = new String();
16        idade = 0;
        id = 0;
18        sexo = true;
        this.con = con;
20    }

22    //getters and setters
    public int getId() {
24        return id;
    }

26    public void setId(int id) {
        this.id = id;
28    }

30    public String getNome() {
        return nome;
32    }

    public void setNome(String nome) {
```

```
34     this.nome = nome;
35 }
36
37 public int getIdade() {
38     return idade;
39 }
40 public void setIdade(int idade) {
41     this.idade = idade;
42 }
43
44 public boolean getSexo() {
45     return sexo;
46 }
47 public void setSexo(boolean sexo) {
48     this.sexo = sexo;
49 }
50
51 //persistent methods
52 public boolean save() {
53     Statement stmt = null;
54
55     //create a query
56     String query = "INSERT INTO PESSOA (ID, NOME, IDADE,SEXO) ";
57     query = query + "VALUES (ID_, 'NOME_', IDADE_, 'SEXO_') ";
58
59     //create string data from class attributes
60     String idString = new Integer(id).toString();
61     String idadeString = new Integer(idade).toString();
62     String sexoString = new Boolean(sexo).toString();
63     sexoString = sexoString.substring(0,1);
64
65     //put the attribute values into the query
66     query = query.replaceFirst("ID_", idString);
67     query = query.replaceFirst("NOME_", nome);
68     query = query.replaceFirst("IDADE_", idadeString);
69     query = query.replaceFirst("SEXO_", sexoString);
70
```

```

72     try {
73         //create and execute the statement
74         stmt = con.createStatement();
75         stmt.executeUpdate(query);
76
77         try {
78             //close connection
79             stmt.close();
80             return true;
81         } catch (SQLException ex) {
82             System.out.println(ex);
83             return false;
84         }
85     } catch (Exception ex) {
86         System.out.println(ex);
87         return false;
88     }
89
90     public boolean getById(int id) {
91         Statement stmt = null;
92         ResultSet rs = null;
93
94         String idString = new Integer(id).toString();
95
96         String query = "SELECT * FROM PESSOA WHERE ID = ID_";
97         query = query.replaceFirst("ID_", idString);
98
99         try {
100             stmt = con.createStatement();
101             rs = stmt.executeQuery(query);
102
103             while (rs.next()) {
104                 this.id = rs.getInt("ID");
105                 nome = rs.getString("NOME");
106                 idade = rs.getInt("IDADE");
107                 String sx = rs.getString("SEXO");

```

```
108         if ( sx.equals( 't' ) )
                sexo = true;
110         else
                sexo = false;
112     }

114     try {
        stmt.close();
116         return true;
    } catch (SQLException ex) {
118         System.out.println(ex);
        return false;
120     }
    } catch (Exception ex) {
122         System.out.println(ex);
        return false;
124     }
    }

126 }

128 public boolean delete() {
    Statement stmt = null;
130
    String idString = new Integer(id).toString();
132
    String query = 'DELETE FROM PESSOA WHERE ID = ID_';
134    query = query.replaceFirst( 'ID_', idString );

136    try {
        stmt = con.createStatement();
138        stmt.executeUpdate(query);

140        try {
            stmt.close();
142            return true;
        } catch (SQLException ex) {
144            System.out.println(ex);
```

```
        return false;
    }
    } catch (Exception ex) {
        System.out.println(ex);
        return false;
    }
}
}
```

À partir das matrizes apresentadas nas Listagens 1 e 2, o engenheiro de domínio pode iniciar o desenvolvimento dos *templates*.

A linguagem de transformações XSL está fora do escopo desse manual. Instruções detalhadas da linguagem XSL podem ser encontradas no site: [http:// www.w3c.org/xslt](http://www.w3c.org/xslt) e em diversos manuais e livros disponíveis livremente ou para compra na internet.

No diretório `install_dir/doc/training/XSLT-examples` podem ser encontrados exemplos de transformações XML em ordem progressiva de complexidade. Esses exemplos podem ser executados na linha de comando ou com a ferramenta Ant.

### 3.3 Desenvolvimento do arquivo de mapeamento de transformação de templates

O arquivo de mapeamento de transformação de templates deve informar a ferramenta sobre a necessidade da geração de dois tipos de arquivos: o arquivo com os scripts SQL para a criação das tabelas no banco de dados e as classes Java persistentes.

Para cada projeto deve ser gerado um script SQL e dentro de um projeto, para cada especificação de classe persistente, deve ser criado um arquivo com o nome igual ao nome da classe definida no formulário de especificação de classes.

### 3.4 Exercício 1: Configuração da ferramenta

Na primeira parte do exercício, o leitor deve configurar a ferramenta Captor com base nos dados fornecidos nesta Seção. As atividades necessárias para completar o exercício são:

1. Criar o arquivo de configuração principal com a definição dos formulários e dos mecanismos de validação.
2. Criar os templates para gerar o código SQL e os templates para o Código Java.
3. Criar o arquivo de mapeamento de transformação de templates.

4. Implantar os artefatos criados na ferramenta.
5. Testar a nova configuração.

## 4 Utilização da ferramenta

Para utilizar a ferramenta, o engenheiro de aplicação deve possuir a especificação de requisitos de uma aplicação em um domínio que a ferramenta forneça apoio, editar os dados dessa especificação na ferramenta, gerar artefatos e utilizar esses artefatos no desenvolvimento de aplicações.

### 4.1 Exercício 2 - Utilização da ferramenta

O estudo conduzido nesse exercício deve implementar as classes persistentes para as tabelas relacionais apresentadas na Figura 7.

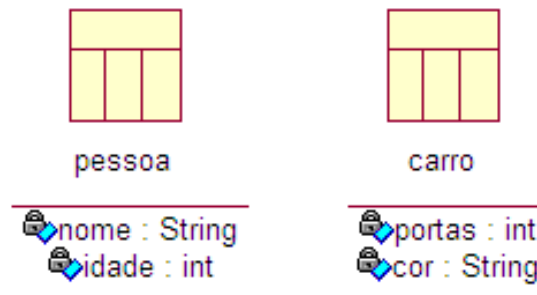


Figura 7: Tabelas relacionais da aplicação exemplo

As tabelas relacionais apresentadas na Figura 7 são mapeadas para as classes persistentes apresentadas na Figura 8.

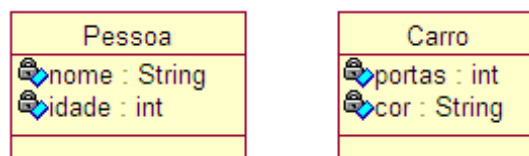


Figura 8: Classes persistentes

Para o exercício de utilização da ferramenta Captor são propostas as seguintes atividades:

- Edição de uma especificação da ferramenta Captor que represente as tabelas e classes apresentadas nas Figuras 7 e 8.
- Geração dos artefatos.
- Teste dos artefatos gerados.

- Criação de zonas de segurança nos templates para adição de novos métodos nas classes geradas.
- Modificação dos artefatos gerados dentro das zonas de segurança.



## Referências

Hibernate Hibernate framework. 2005.

Disponível em <http://www.hibernate.org> (Acessado em 19/10/2005)

MySQL The mysql database. 2005.

Disponível em <http://www.mysql.org> (Acessado em 19/10/2005)

OJB Object-relaction bridge. 2005.

Disponível em <http://db.apache.org/ojb/> (Acessado em 19/10/2005)

Weiss, D. M.; Lai, C. T. R. *Software product-line engineering*. Addison-Wesley, 1999.