Camilo Lopes

# GUIA PRÁTICO

## CONSTRUINDO APLICAÇÕES JEE COM FRAMEWORKS

"Exclusivo para Iniciantes – JSF, SpringSecurity, Hibernate, MySQL, Eclipse"


EDITORA
CIÊNCIA MODERNA

# Sobre o Autor

Baiano de 23 anos, Camilo Lopes trabalha com Tecnologia de Informação desde 2003, possui experiência em desenvolvimento, analise de requisitos, modelagem de dados, servidores de aplicações, banco de dados e linguagens de programação. Trabalhou de 2004-2006 como Freelance desenvolvendo sistemas em VB6 e PHP 4 para pequenas empresas. Programador certificado pela Sun Microsystems - SCJP 5 em 2007. Bacharelado em Análise de Sistemas, Pós-Graduado em Gestão Estratégia de Negócios. Trabalhou de 2008-Julho/2010 na IBM Brazil como Analista Programador Java em projetos Internacionais. Em 2009 foi nomeado Embaixador Técnico IBM onde ministrou palestras e mini-cursos em faculdades/universidades nas horas livres. Atualmente trabalha na HP Brazil como Software Engineer no Enterprise Computing Lab (ECL), onde é responsável pelo design, desenvolvimento, manutenção, testing, qualidade do Software utilizando a tecnologia Java. Nas horas livres gosta de escrever livros técnicos, artigos para as revistas Java Magazine, MundoJ (antiga MundoJava) e Espiríto Livre. Possui artigos Java no UOL IMasters e InfoQBrasil. Em seu blog www.camilolopes.com.br há diversos posts sobre Java, Carreira Profissional, Inglês, Banco de dados etc que são atualizados semanalmente.

**Contato:** Envie sua sugestão, crítica, elogio ou suas dúvidas para o e-mail do autor: camilo@camilolopes.com.br. Visite o *website*: www.camilolopes.com.br.

# Agradecimentos

Quero agradecer a um grande amigo, que contribuiu fortemente para que este projeto fosse publicado, o autor Edson Gonçalves; agradeço a ele pelas dicas, críticas, elogios e incentivo nas horas mais difíceis, quando os verdadeiros amigos aparecem e te mostram o caminho. Ao Paulo, diretor da Ciência Moderna, pela oportunidade, e a toda a equipe editorial, que trabalhou "duro" nas devidas correções e sugestões.

Quero agradecer a uma gaucha que és uma uma grande amiga desde de 2007 a qual contribuiu inumeras vezes na minha carreira desde quando nos conhecemos no GUJ (Grupo de Usuários Java – www.guj.com.br) e eu ainda dava "Hello World" no Java, falo da Veronica Nunes, a qual dedico esta obra.

Ariane Camori, uma paulista que pediu meu contato para tirar uma dúvida sobre Java e com o passar do tempo se tornou uma pessoa especial, à qual não poderia deixar de agradecer, pois a "Nany" tem participação direta e indireta neste resultado; obrigado pelo apoio, dicas e conselhos.

Não posso esquecer de dois amigos que conheci ainda em formação do meu bacharelado e que estudávamos juntos inglês via Skype e discutíamos todos os dias assuntos de T.I são eles: Alberto Leal e o Mario Razec. Abraços, meus amigos, vocês são responsáveis por este trabalho, pois quando um estava desmotivado, o outro estava ali para dar aquele fôlego e não deixar a peteca cair.

# Dedicatória

Dedico este trabalho, o qual batizei de “meu segundo filho”, aos meus avós, Sr. Camilo Medeiros e Sra. Benigna Sales, e à minha mãe, Sra.Celidalva Medeiros. Vocês são os responsáveis por tudo de bom que tem acontecido na minha vida e sou eternamente grato. Este trabalho é para vocês. Não poderia deixar de dedicar a mais uma pessoa que contribuiu positivamente nas minhas conquistas com incentivo, opiniões e não me deixou desistir quando pensei abandonar. Refiro-me à minha namorada/noiva Efigênia Maurício; agradeço a você eternamente pelo que fez por mim, que Deus te abençoe, esta obra é sua também. Uma pessoa que abriu as portas e que jamais vou esquecer no dia que me fez o convite para fazer parte da IBM e os helps mais como amigo que como colega de trabalho, eu dedico este trabalho, falo do Juliano Martins.

# Como nasceu este guia?

Bem, tive a ideia de criar este guia a partir de uma necessidade e das dificuldades que tive quando iniciei meus estudos com JEE, logo após ter visto JSP & Servlet. Buscava um livro que ensinasse de modo prático os *frameworks* mais usados no mercado, porém, queria um livro que não fosse muito extenso, mas bem focado naquilo que seria apresentado e com exercícios interessantes que eu pudesse praticar. Foi muito difícil encontrar obras com esse perfil, e eu não concordava muito em ter que comprar um livro para todos os *frameworks* que teria que aprender, sendo que para um iniciante, em primeiro momento, o mais importante é ver o básico e fazer a fusão de uma tecnologia com outra. Exemplo: JSF com Hibernate. A partir desta dificuldade que passei, resolvi criar uma obra para ajudar aqueles que estão chegando ao mundo JEE e se sentem perdidos em tantas siglas: *Struts, Jboss Seam, JSF, Hibernate, Ibatis* etc. Eu já tive minhas confusões e não conseguia entender o porquê de cada um e como eles iam trabalhar juntos em uma aplicação JEE, pois quando comprava um livro sobre JSF este não abordava nada de Hibernate, e o inverso também era válido, sem falar que os exercícios não eram nada práticos e prazerosos de fazer. Diante tudo isso, nasceu este guia, para suprir esse "buraco" na nossa vida de programador.

Espero atingir este objetivo.

# Sobre o Livro

O guia foi desenvolvido com o objetivo de orientá-lo no aprendizado prático de como desenvolver aplicações JEE usando os *frameworks* mais cotados no mercado. Neste caso buscamos unir um pouco de teoria com a prática através de aplicações Java e JEE. Não é nosso objetivo esgotar todas as características de um *framework* ou tecnologias que serão vistas aqui, mas buscaremos explorar o suficiente para permitir ao leitor criar aplicações reais.

Usamos uma linguagem um pouco coloquial em algumas partes do livro, pois acreditamos que usar a primeira pessoa ou alguns termos informais pode de fato ajudar a compreender assuntos mais complexos para um iniciante, e o nosso objetivo é facilitar a assimilação ao máximo possível.

# Público-alvo

Quem pode ler? Essa é pergunta que todo leitor faz; será que eu posso ler esse livro? Não queremos que você, leitor, compre um exemplar se não está preparado. O livro é recomendado para aquele programador que está acabando de dar seus primeiros passos em Servlet & JSP e pretende agora ir para o mundo dos *frameworks,* já que 99% das vagas de emprego pedem conhecimento de, no mínimo, 2 ou 3 *frameworks*. Se você está nessa situação e deseja aprender de modo prático alguns *frameworks* JEE mais usados no mercado, este livro é para você. Caso esteja ainda nos primeiros passos do Java/JEE, coloque o livro novamente na prateleira e não o compre.

# O que não vou aprender

- Você não vai aprender o básico do Java, Orientação a Objetos, JEE Básico;

- Não aprenderá a instalar e configurar um ambiente de desenvolvimento Java;

- Não vai terminar o livro um *expert* nos *frameworks,* mas conhecerá o suficiente para criar uma aplicação;

- Não se tornará um DBA, mas vai saber usar o banco de dados nas suas aplicações;

- Não será explicado JSE;

- Não esgotaremos todo o conteúdo das tecnologias apresentadas, uma vez que este não é o objetivo principal do livro.

# O que vou aprender

Você vai aprender como usar Hibernate em aplicações Java/JEE e criar aplicações JEE usando JSF. Além disso, vamos envolver um pouco de banco de dados, modelagem etc. Enfim, trazer um cenário real para dentro do livro. E não haverá atividades como "exercícios de sala".

Não vamos torná-lo um especialista nos *frameworks* (preciso reforçar isso para ter certeza de que não esqueceu), mas queremos muito que, ao término do livro, você possa criar suas aplicações usando as tecnologias abordadas aqui. Claro que para requisitos mais complexos terá que recorrer a uma bibliografia mais aprofundada no assunto. Para ter ideia, existem livros com mais de 400 páginas somente tratando de Hibernate, JSF etc. Então, não seria nada legal dizer que nesse pequeno guia teria tudo sobre cada um. Mas, aqui há o suficiente para sair do 0 x 0 e começar a desenvolver aplicações usando os frameworks apresentados.

# Como ler o livro?

Bem, apesar de ter uma sequência de capítulos, não há, de fato, uma obrigatoriedade em lê-los na sequência. Se você já viu Hibernate, mas não sabe integrá-lo com JSF, então não precisa ler os tópicos referentes ao básico do Hibernate. Recomendamos dar uma folheada no livro antes de iniciar.

Agora, se for iniciante, pode ler em sequência. Porém, quando for desenvolver aplicações JEE terá que ir para o capítulo JSF para entender como JSF trabalha. Se não o fizer, só vai ficar digitando os *codes* sem saber o significado, e isso não é bom para seu aprendizado. Citamos JSF porque todos os exemplos de JEE usam o *framework* JavaServerFaces.

# Metodologia adotada

Como escrevemos um livro diferente, a metodologia de leitura também deve ser diferente, uma vez que queremos torná-lo um programador prático. Aqui fizemos diferente, você vai acabar lendo todos os capítulos sem perceber que terminou o livro, uma vez que é bem comum requisitarmos que vá para o capítulo X e aprenda tecnologia Y antes de continuar e verá que será impossível ir adiante se não fizer o que indicamos; fizemos isso para ter a certeza de que você aprendeu todas as tecnologias apresentadas. Então, esqueça aquela ordem de leitura sequencial "presa" de capítulos. Mas não fique preocupado, porque isso não o deixará confuso, pelo contrário, achamos que será mais divertido e menos cansativo.

# Trabalho do dia - O que é isso?

Cansamos daqueles livros tradicionais que muitas vezes ensinam ao leitor como somar "2 + 3" e no exercício perguntam "quanto é 3 + 2". Você já entendeu aonde queremos chegar, não é? Então, criamos este tópico para ser diferente e realmente despertar o interesse de você programador querer resolver "o problema".

Quando adotamos este método não pensamos momento nenhum em pedir algo que não fosse possível implementar. Pelo contrário, *trabalho do dia* tem como objetivo de você programador poder testar seus conhecimentos adquiridos não somente no tópico em questão, mas consultar outros capítulos e o apêndice do livro. Assim, você consegue ler todo o livro de forma divertida e prática, pois vai aprendendo à medida que programa. Então, não se espante se você tiver que consultar o apêndice de Banco de Dados, modelagem de Dados etc. para poder implementar todos os requisitos para seu *trabalho do dia.*

Sem falar que o outro objetivo deste tópico foi transmitir o dia a dia de um desenvolvedor, pois saber resolver problemas é a maior habilidade que você deve ter antes de querer dominar os inúmeros *frameworks* disponíveis no mercado. Em função disso, não fornecemos as respostas para as soluções, pois você, como bom profissional, tem que ter a capacidade de confiar na sua solução e saber se ela está atendendo todos os requisitos do cliente.

# Laboratório *On-line*

O autor criou um laboratório *on-line*, onde o leitor pode fazer o *download* dos exemplos usados no livro, como também dos *frameworks* e algumas ferramentas. Basta acessar:
http://lab.camilolopes.com.br.

# Do autor para os leitores

Primeiramente, quero agradecer a você por ter confiado em mais um dos meus trabalhos e ter adquirido este exemplar. Fico muito grato e feliz. Quando criei este projeto, em momento nenhum pensei em fazê-lo um *best-seller*. Para quem já acompanha meus trabalhos sabe que sempre estou direcionado a ajudar aqueles que estão no início ou chegando a um mundo novo, neste caso aqui JEE com Frameworks, e me sinto muito feliz em poder contribuir de alguma forma para o desenvolvimento do próximo, uma vez que quando dei meus primeiros passos tive apoio de várias pessoas que tiveram a paciência e disponibilidade para ajudar aquele mero iniciante. E hoje poder retribuir da mesma forma que fizeram comigo é algo que faço por prazer. Então, quando puder, sempre envie um *feedback,* este é muito importante para os próximos projetos, seja uma crítica, um elogio, sempre será válido. Obrigado e espero que tenha uma boa leitura e que ao término do livro tenha aprendido de fato a desenvolver aplicações usando *frameworks* JEE (Hibernate e JSF).

# Sumário

# Capítulo 1
# Introdução

Não seremos cruéis em querer jogar os assuntos práticos logo no primeiro capítulo. Apesar de este ser um guia rápido e prático, teremos cuidado em como apresentar os assuntos para você. Temos o objetivo de facilitar o entendimento teórico e prático de como usar Hibernate + JSF. Principalmente se você vem do Servlet & JSP pode estar se sentindo perdido com tantas siglas de *frameworks* existentes no mercado e sem saber para onde ir. Eu passei por isso e acredito que muitos desenvolvedores iniciantes em JEE também venham a sofrer desse mal.

Neste capítulo você vai saber o porquê do Hibernate no ambiente de desenvolvimento JEE para *frameworks,* veremos um pouco de banco de dados etc. É somente para dar um *refresh* na sua mente. Mas não pule este capítulo, principalmente se você estiver vendo o assunto pela primeira vez. Muitas informações aqui expostas são requisitos para os próximos tópicos e, por ser um guia rápido de estudos, procuramos não ser repetitivos ou duplicar informações**.**

## O que é Hibernate?

O Hibernate é um *framework* que procura ter uma completa solução para o problema de gerenciamento de dados persistentes em Java. O relacionamento entre o Hibernate e o banco de dados é algo bem transparente e que não exige do desenvolvedor um *hard-work development*. E assim este pode se concentrar mais na lógica de negócios. Em uma mudança no BD, o desenvolvedor não precisa sacrificar o final de semana para implementar o recurso novo à nível de código,

apenas alterando algumas configurações no Hibernate já será possível ter toda a aplicação atualizada. Sem sombra de dúvidas, quem trabalha com JDBC diretamente e vem para o *framework,* com o passar do tempo até esquece de como era difícil trabalhar com BD usando JDBC no padrão DAO.

No geral, usamos o Hibernate para fazer a integração da nossa aplicação com o banco de dados da mesma forma que fazemos no JDBC. O resultado final é o mesmo, de ter dados atualizados no banco, mas a forma como esses dados do banco podem ser manipulados dentro de sua aplicação JEE é que faz toda a diferença.

## É difícil?

Essa é uma pergunta que já ouvi muitas vezes. Percebi que muitos iniciantes criam bloqueios somente em ver o nome do *framework,* já que são tantos que as empresas exigem. Mas, se olharmos por outro lado: "Se fosse difícil e complexo, por que de 10 empresas, 9 pedem? E ainda por cima pedem no mínimo 2 ou3?". Isso quer dizer que não é algo de outro mundo, que uma pessoa normal possa aprender 1, 2, 3.... *frameworks* ao mesmo tempo e saber em qual situação usar o X e o Y. Então, ser difícil ou não, depende do tamanho do obstáculo que você colocar na frente. Mas, é requerido que o desenvolvedor já tenha experiência em fazer conexões Java com Banco de Dados, saber programar em Java etc. Tudo depende de qual *framework* vai ser estudado, porém, o que foi citado é a base de tudo para desenvolver uma aplicação para o mundo real. Não precisa ser um *expert* em Java para aprender os *frameworks,* mas saber fazer uma aplicação completa já é um bom caminho percorrido. Agora, para quem nunca viu a tecnologia Java, não adianta querer pular as etapas de aprendizado; por favor, retorne à livraria e troque o nosso guia por outro livro recomendado para iniciantes Java (há alguns do autor Edson Gonçalves que valem a pena serem verificados). Pois aqui não ensinaremos nada para quem estiver ainda no "berço" do Java. O nosso livro é para quem já *saiu do berço* e está dando os primeiros passos com *frameworks* de modo prático.

## Ambiente de Desenvolvimento

Para configurar o ambiente de desenvolvimento com Hibernate é muito simples, e dizem por ai que é mais fácil do que *tirar doce de criança recém-nascida*. Neste guia teremos o ambiente abaixo para construção dos nossos projetos:

- ✓ Eclipse Galileo ou mais recente;
- ✓ MySQL 5.0.x;
- ✓ WorkBench 5.x;
- ✓ Frameworks: JSF 1.2 e Hibernate 3.x;
- ✓ Jboss Tools 3.x;
- ✓ TomCat 6.x.

---

***Nota:*** *Claro que teremos o Java já instalado na máquina. Estamos trabalhando com a versão 1.5 da tecnologia Java, você pode usar esta versão ou outra superior.*

---

Um detalhe importante é que sempre tenho o hábito de trabalhar em artigos, livros, postagens no meu blog etc. com versões não tão recentes de uma tecnologia, isso para atender a um público maior e poder expandir e compartilhar o conhecimento. Uma vez que funciona perfeitamente na versão anterior, teoricamente na atual deve funcionar.

## Preparando o ambiente: Eclipse

Para preparar o ambiente com o Eclipse é bem simples; basta ir ao site do Eclipse (www.eclipse.org) e fazer o *download* da versão Galileo.

Após terminar o *download*, descompacte o arquivo e execute o **eclipse.jar**. Essa é uma vantagem da IDE, não é necessário instalar, somente executar e já a teremos em ação.

Na primeira tela, o Eclipse deseja saber em qual *workspace* ficará seus projetos. Você pode usar o padrão da própria IDE ou criar uma.

Escolha onde seu projeto vai ficar armazenado e clique em OK. Após isso, o Eclipse tem uma tela de boas-vindas – *Welcome.* Caso queria obter maiores informações da IDE, essa tela pode te ajudar. Feche-a para irmos adiante.

Se escolher um *workspace* vazio, com certeza não teremos nenhum projeto sendo visualizado na IDE.

Pronto. Já temos a IDE apta para trabalhar.

---

***Nota:*** *O Java já deve estar devidamente instalado e configurado na máquina. O Eclipse é "inteligente" o suficiente e já deixa a IDE configurada e pronta para trabalhar, uma vez que o Java JDK já está na máquina. Se houver qualquer dúvida consulte o capítulo: "Links de Auxílio" e veja como instalar o Java na sua máquina.*

---

## Configurando o Ambiente: Hibernate

Configurar o Hibernate no Eclipse é muito simples. Basta ir ao site www.hibernate.org e clicar na opção *download*. No link abaixo, que estava ativo no momento que escrevi o guia, existem todas as versões do Hibernate.

O Hibernate informa quais arquivos **.jar** são requeridos. Mas, o que são eles?

Poucos sabem (iniciantes), mas o Hibernate possui um arquivo **.txt** localizado na pasta **lib** chamado de *README.txt*, que tem a função de informar ao programador quais arquivos são *necessários* para o *framework* trabalhar. E há uma pequena explicação do uso de cada *.jar,* que é fornecida pelo próprio fabricante.

Abra o arquivo *README.txt* e veja quais são *requeridos*. Abaixo, os listamos:

1. antlr-2.7.6.jar
2. asm.jar
3. asm.attrs.jar
4. c3p0-0.9.9.jar
5. cglib-2.1.3.jar
6. commons-collections-2.1.1.jar
7. commons-loggin-1.0.4.jar
8. dom4j-1.6.1.jar
9. hibernate3.jar
10. jta.jar

---

***Nota:*** *Não é objetivo explicar para que serve cada* ***.jar*** *acima, mas caso tenha interesse, verifique na documentação do* framework *uma vez que ele possui todas as informações que você precisa sobre a tecnologia*

---

Alguns programadores, principalmente os iniciantes usam o termo "instalar Hibernate". Bem, o *framework* não é instalado e sim configurado na IDE Eclipse. Não há nada para instalar, apenas para configurar, como veremos na prática mais à frente.

## Criando a biblioteca

Os passos aqui são válidos para qualquer outro tipo de *framework* (JSF, Struts etc.) ou algum arquivo **.jar** que deseja tornar uma biblioteca para facilitar adição em outros projetos. Sendo assim, para o Hibernate, vamos pedir a você que somente crie a biblioteca. Se houver qualquer dúvida futuramente como fazer, basta retornar a este tópico e ver como fizemos com o Hibernate e seguir os mesmos passos.

1. Clique em *Window =>Preferences*

2. À esquerda, digite *user* . Veja na imagem abaixo:

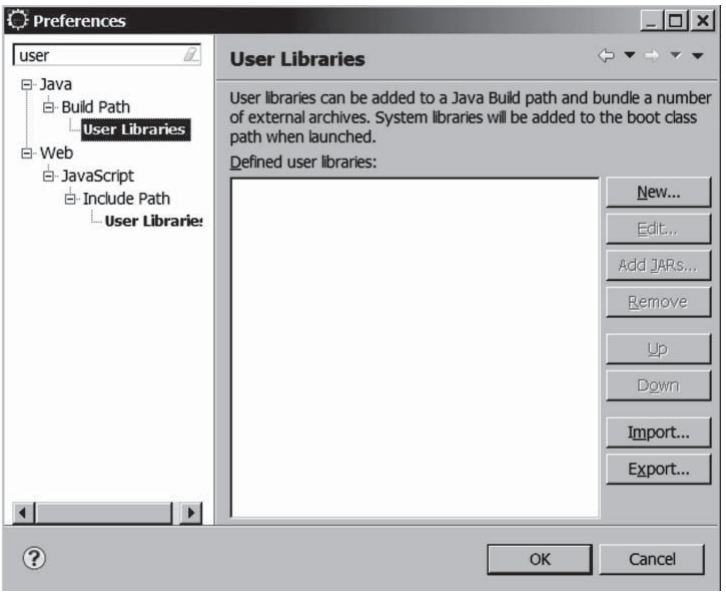



***Figura 1: Criando uma biblioteca no Eclipse***

3. Clique no botão *new.*

4. Agora precisamos dar um nome à nossa biblioteca, então digite: *Hibernate 3.2.1.* É considerado boa prática de programação o uso do nome do *framework* + versão. Assim, podemos ter vários projetos e cada um usando uma versão diferente do *framework.* Não se esqueça de que você ainda será convidado para trabalhar em sistemas legados, então será comum ter uma biblioteca configurada com uma versão mais antiga do *framework*.

5. Após ter definido o nome, vamos adicionar os **.jars** à biblioteca. Para isso, clique em *add jars.*

6. Agora você precisa localizar os .jars *required* do Hibernate e adicioná-los à biblioteca. Veja:

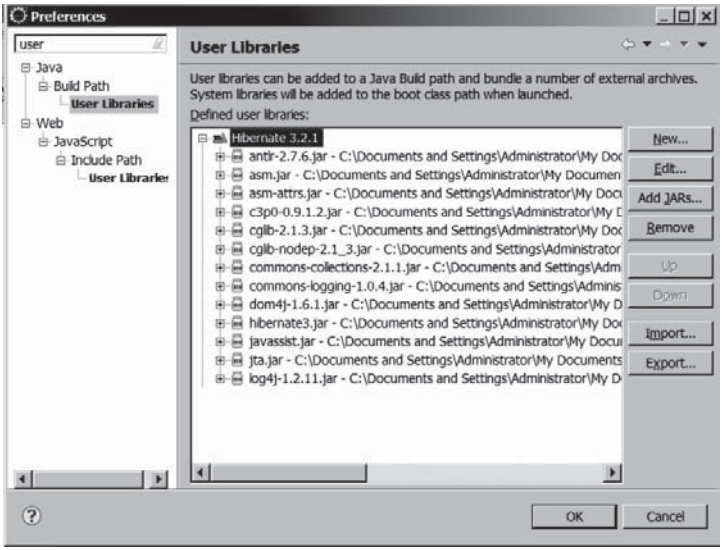



***Figura 2: Biblioteca Hibernate***

7. Feito isso, clique no botão OK.

Assim, criamos uma biblioteca para o Hibernate. Para adicioná-la ao projeto precisamos apenas clicar com o botão direito em cima do projeto e escolher a opção:

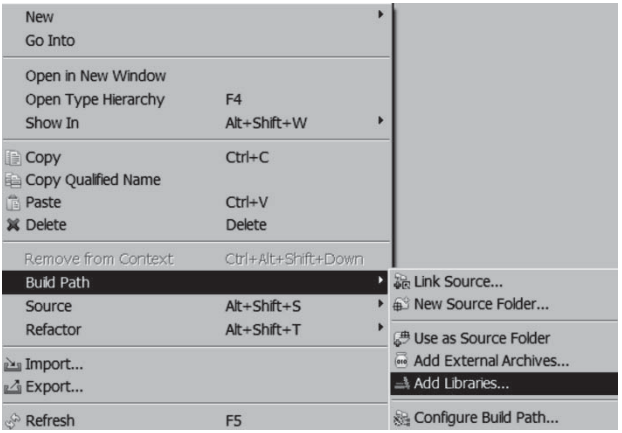



***Figura 3: Adicionando Bibliotecas***

1. Em seguida, escolha a opção *user finish* e clique em *next*. Na próxima tela, escolha a biblioteca Hibernate para adicionar ao projeto. Caso tivéssemos mais de uma *library*, esta seria exibida nesta página.

2. Clique em *finish* para concluir.

Assim, adicionamos o Hibernate ao projeto. Agora, podemos dizer que o projeto já pode trabalhar com o *framework*, o que veremos mais adiante.

---

***Nota:*** *Há um pequeno detalhe que deve ser levado em conta nos projetos JEE, principalmente ao adicionar arquivos .jars. O Eclipse não copia fisicamente os* ***.jars*** *para a pasta lib de forma automática. E isso traz várias exceções que pode deixá-lo estressado tentando buscar erros em locais que não existem. Consulte nosso capítulo links de auxílio e veja como abordamos e resolvemos esse problema, que é da IDE Eclipse não copiar de forma automática os .jars quando estes são adicionados ao projeto. Lembre-se de que isso somente acontece com projetos JEE.*

---

## Configurando o JSF

Para configurar o JSF, utilizaremos o mesmo processo que foi feito com o Hibernate. Ou seja, criaremos uma biblioteca (*library)* com os .jars para JSF.

Para fazer o *download* do JSF vá no seguinte endereço:

http://java.sun.com/javaee/javaserverfaces/download.html

Usaremos a versão 1.2 por ser compatível com a tecnologia JavaEE 5. Os arquivos .jars que precisamos são:

- ✓ jsf-api.jar
- ✓ jsf-impl.jar

Crie uma biblioteca *user* para este framework e chame de *JSF 1.2*. Quando precisar ter JSF em seus projetos JEE, basta adicionar esta biblioteca.

## Instalando/Configurando Jboss tools

O *Jboss tools* é um *plugin* que instalamos no Eclipse para dar suporte aos nossos projetos JEE. A ferramenta dá suporte a vários *frameworks* como: *Struts, JSF, JPA* etc. No *link* a seguir há um vídeo explicando como instalar e configurar a ferramenta no Eclipse:

www.camilolopes.com.br/videos/InstalandoJBossTools31/

Após a instalação, será solicitado um *restart* no Eclipse. Faça isso para que as alterações entrem em vigor.

## Banco de dados

Usaremos o MySQL como banco de dados para as nossas aplicações. Além de ser potencialmente famoso, possui muitas características que não ficam a desejar com relação aos BD pagos.

Como ferramenta de modelagem teremos o WorkBench, da própria Sun, que trabalha muito bem com o MySQL. O WorkBench é, na verdade, uma continuação do *BD Designer* que permaneceu no mercado como ferramenta de modelagem por muito tempo.

Se você não tem o MySQL e o WorkBench instalados, vá no capítulo referente ao assunto e faça a instalação. Muito cuidado com a instalação do MySQL para não esquecer a senha de *root*.

O WorkBench vai nos ajudar na modelagem. Que tal aplicar *engenharia reversa* em seu Banco? Ou gerar um banco a partir de uma modelagem? É isso que a ferramenta oferece. Veja abaixo uma modelagem feita no WorkBench:

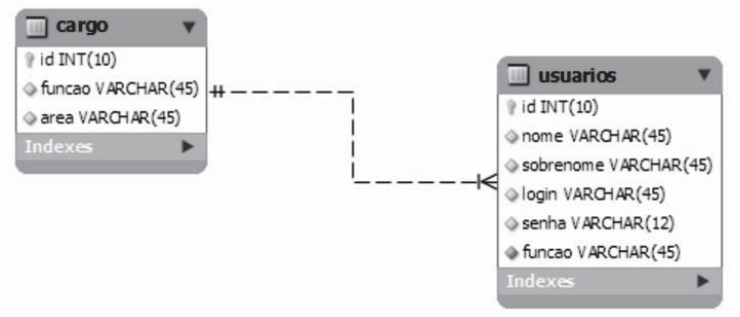



***Figura 4: Modelagem de dados***

É obvio que a ferramenta precisa melhorar, e muito. Até porque não atende a projetos mais complexos, falta muita coisa e volta e meia aparecem *bugs*. Mas nada a impede de trabalhar em projetos pequenos. Por ser *free*, tem seu custo *x* benefício.

Não espere se tornar um *expert* em DB ou em modelagem de dados; vamos apresentar aqui o assunto para você não sair do guia somente com *codes* em mente. Acreditamos que um bom desenvolvedor deva ser especialista em sua área de atuação, mas conhecedor das demais. Você não precisa ser um DBA para trabalhar com BD. Mas, tem que conhecer o ABC de um BD.

## Configurando BD MySQL no Eclipse

Para trabalharmos com o BD em nossos projetos JEE/Java precisamos ter o *driver* de conexão do BD. O *driver MySQL* que temos é o *connector/J,* que pode ser baixado no site do fabricante. Não passa de um arquivo *.jar.*

http://www.mysql.com/downloads/connector/j/

Adivinha o que você precisa fazer com esse *.jar*... Isso mesmo, criar uma biblioteca chamada MySQL 5.0 e, quando criar seus projetos com conexão ao BD, adicionar a eles essa biblioteca.

### Concluindo

Não fique assustado com o que foi visto neste primeiro capitulo, apenas fizemos um *overview* das coisas basicas e essenciais, no decorrer dos próximos capítulos vamos mostrando de modo prático como desenvolver de fato.

# Capítulo 2
# Hibernate

Esperamos que você aprenda o bastante nesse capítulo, pois tudo o que você precisa saber para dar os primeiros passos com Hibernate Frameworks estarão aqui. Alguns tópicos são curtos, para não cansá-lo. Lembre-se de que você adquiriu um guia rápido de estudo focado na praticidade, e não um livro que vai esgotar tudo sobre uma tecnologia. Mas, mostraremos o caminho da "felicidade" e cabe a você ir em frente ou não.

## Conhecendo a estrutura do Hibernate

Antes de colocarmos a *mão na massa,* precisamos conhecer a estrutura funcional do Hibernate. Abordaremos aqui apenas o necessário e que consideramos importante para que se, amanhã, o seu colega na esquina perguntar a você para que serve um *SessionFactory,* você saiba dar uma resposta rápida e inteligente.

Um dos primeiros pontos que queremos abordar é que o Hibernate usa JDBC para acesso ao Banco de Dados. Com Hibernate ainda é necessário passar algumas informações "manuais" para conexão como se faz via JDBC, porém, agora temos mais flexibilidade. Como bom desenvolvedor, acreditamos que você vai conseguir perceber a diferença no decorrer deste guia.

O que você precisa saber:

- ✓Para cada Banco de Dados precisamos informar o *dialect* para execução do Hibernate com o Banco de Dados (não se preocupe, ainda vamos apresentar a você uma tabela com alguns *dialects*);

✓As configurações de conexão com o Banco de Dados podem ficar em um arquivo Hibernate.properties ou em um arquivo XML chamado *hibernate.cfg.xml* (mas você pode alterar o nome, desde que o especifique quando for criar *SessionFactory*), que deve ficar no package *src,* se estiver usando *Java Project,* ou em *JavaSource,* se for um *Web Project;*

✓ O mapeamento normalmente é criado com o nome do **bean.hbm.xml (Pessoa.hbm.xml)**. Podemos colocar o mapeamento/ persistência onde acharmos mais conveniente. Porém, teremos que informar isso no *hibernate.cfg.xml* quando formos mapear o arquivo *.hbm.xml*. A maioria dos desenvolvedores põe o mapeamento no mesmo *package* do *bean*.

✓Para criar um objeto *Session* é bem simples, veja o código a seguir:

```
1. SessionFactory sessionFactory = new Configuration.configure().
buildSessionFactory();

2. Session session = sessionFactory.openSession();
```

✓Quem faz a busca pela configuração do Hibernate com o seu BD é a classe *Configuration.*

O arquivo "default" **hibernate.cfg.xml**, por isso que deixamos o argumento do método *configure()* vazio. Mas, se for diferente, devemos informar *configurate("meuhibernate.cfg.xml")*.

✓Para persistência dos dados usamos um objeto **Session;**

✓**SessionFactory:** é a fábrica de sessões, onde temos o ciclo de vida. Com ela criamos um objeto Session para gerenciar a conexão com o BD;

✓Para cada BD diferente precisamos ter um SessionFactory;

✓Obter a uma Session temos que usar um objeto de SessionFactory ( no exemplo acima temos sessionFactory) e chamar o método *OpenSession();*

✓Então, memorize a estrutura básica de arquivos que você vai precisar antes de pensar em escrever uma linha de código na camada de negócio da sua aplicação. Veja:

1. Mapeamento da classe Bean;
2. Configuração do Hibernate com BD:

✓No arquivo *hibernate.cfg.xml* temos algumas linhas de extrema importância que você jamais pode esquecer. São elas:

- hibernate.connection.DataSource = passamos o JNDI name
- hibernate.jndi.url = Url do BD
- hibernate.connection.UserName = usuário do banco de dados
- hibernate.connection.password = que tal a senha do BD?

✓Nunca esqueça que devemos usar o **Dialect SQL** de acordo com o banco de dados a ser conectado. Abaixo, temos um código de configuração para o MySQL 5 que usa InnoBD.

```
1.<session-factory>
2.<property name="hibernate.connection.driver_class">
  org.gjt.mm.mysql.Driver</property>
3.    <property name="hibernate.connection.password">axe
      </property>
4.<property name="hibernate.connection.url">jdbc:mysql://
  localhost/meubancodedados</property>
5.    <property name="hibernate.connection.username">root
      </property>
6.    <property name="hibernate.dialect">org.hibernate.dialect.
      MySQLInnoDBDialect</property>
7.</session-factory>
```

***Nota:*** Você pode estar se perguntando: *"Como saber qual é o* dialect *correto? Eu não uso MySQL; como saber se há suporte para outro banco?"* Bem, não é preciso fazer mágica, você pode ir ao site *www.hibernate.org* e verificar os BDs suportados com seus respectivos *dialects.*

Para facilitar, veja uma tabela resumida dos *dialects* com os bancos mais tradicionais:

| DataBase | Dialect Class |
|---|---|
| DB2/390 | DB2390Dialect |
| DB2/400 | DB2400Dialect |
| DB2 | DB2Dialect |
| Derby | DerbyDialect |
| Firebird | FirebirdDialect |
| FrontBase | FrontBaseDialect |
| HSQLDB | HSQLDialect |
| MySQL 5 | MySQL5Dialect |
| MySQL(<5.x) | MySQLDialect |
| MySQL with InnoDB tables | MySQLInnoDBDialect |
| MySQL with MyISAM tables | MySQLMyISAMDialect |
| Oracle9i | Oracle9Dialect |
| Oracle9i/(DataDirect drivers) | DataDirectOracle9Dialect |
| PostgreSql | PostGreSQLDialect |
| SQL Server | SQLServerDialect |

Na imagem a seguir, temos uma estrutura básica usando um *Java Project* com *Hibernate.* Observe que nossos *mapping* encontram-se na raiz do projeto. Não se desespere porque isso será visto na prática, assim que começarmos a desenvolver.

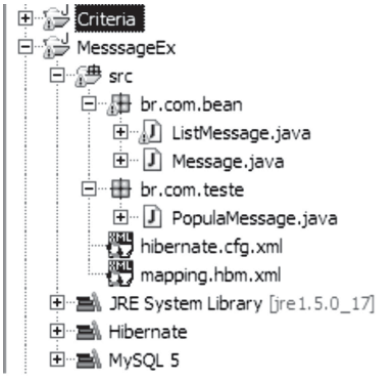



***Figura 5: Exemplo aplicação Java com Hibernate***

Para executar o Hibernate 3, é preciso colocar os arquivos *.jars* na pasta *lib* para um projeto *JEE* e para *Java Project,* basta *add library* ao projeto.

---

***Nota**: Sem os arquivos .jars não temos o Hibernate em ação, por isso eles são requisitos. Sabemos que isso é obvio, porém, apenas reforçamos, caso você tenha esquecido de adicioná-los ao projeto.*

---

Isso é o que você precisa saber basicamente para montar uma estrutura Hibernate. Temos, até aqui, 50% do caminho percorrido. E antes de colocar "a mão na massa", é importante entender o que foi explicado aqui.

## Preparando ambiente Hibernate Eclipse

Veremos neste tópico como preparar o Eclipse para rodar o Hibernate. Primeiramente, você precisa ter feito o *download* da IDE na página www.eclipse.org. Aqui usaremos a versão Galileo, conforme já dito no capitulo anterior.

É importante ter feito o *download* do Hibernate conforme foi mencionado no capítulo 1.

---

***Nota:*** *Usaremos a versão Hibernate versão 3.2.1.GA. O link para ver todas as versões disponíveis no momento em que o livro foi escrito é: http://sourceforge.net/projects/hibernate/files/hibernate3/*

---

Muitos desenvolvedores não sabem quais os *.jars* necessários e muitos apenas escolhem porque um amigo indicou. Porém, existe um motivo para isso (óbvio), basta irmos ao diretório *lib* após descompactarmos o arquivo Hibernate.x.x.x. Na pasta *lib,* há um arquivo chamado *_README.txt*, no qual podemos saber quais *.jars* precisaremos para ter a tecnologia em ação nos projetos. Se houver alguma dúvida sobre quais são os arquivos, retorne ao capítulo 1.

## Adicionando Hibernate ao Projeto no Eclipse

**Opção 1:** Para saber o que adicionar a um projeto, é preciso saber o tipo de projeto que está criando. Se for um *Java Project,* precisamos apenas adicionar a ele os *.jars*

**Opção 2:** Se não for um *Java Project* (*jsf project, web dynamic project* etc.), além de adicionar os *.jars,* é necessário copiar os arquivos fisicamente para a pasta *lib* do projeto. Isso porque ao adicionar os *.jars* ao seu projeto Webo Eclipse não os copia fisicamente.

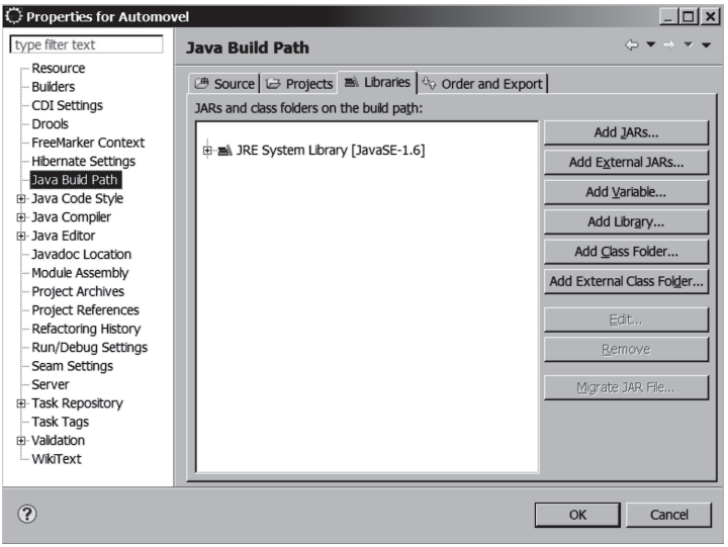
Hibernate Java/JavaEE Project

1. Crie um Java Project;

2. Clique com o botão direito sobre o projeto e escolha *properties;*

3. Na próxima tela, adicione os .jars ao projeto,

4. Clique à esquerda na opção *Java Build Path e* depois, na aba *libraries*. Veja a imagem a seguir:




5. Clique em *add external jars...* e vá até onde estão os arquivos *.jars* do Hibernate;

6. Selecione os *.jars* do Hibernate e adicione-os ao projeto;

7. Uma vez adicionado, clique em OK.

Você terá que fazer isso para cada projeto que criar no Eclipse. A não ser que você coloque os *.jars* no *classpath*, o que não recomendamos, pois como fará se estiver usando 5 ou 6 *frameworks*? Há uma solução bem legal no Eclipse para evitar ficar importando os *.jars* a cada projeto. Basta criar uma *library* e importá-la para o projeto.