Manual Arquitetura – Engenharia de Software B – 2010/2

**Composição dos Projetos**

A aplicação foi dividida em cinco projeto do Netbeans, onde, cada projeto, é respectivo a uma camada da arquitetura do sistema, além de um projeto referente ao aplicativo em si, juntamente com as classes de interface gráfica e um projeto responsável por armazenar classes úteis a todos os outros projetos.

Cada projeto respectivo a arquitetura e o projeto contendo a interface gráfica é dividido em 6 pacotes, onde, 5 são respectivos a cada módulo do sistema e o outro é respectivo a parte de segurança e acessos dos usuários.

Segue abaixo a relação de todos os projetos e pacotes:

**Projeto “br.com.facdombosco.engb.dvo” (contém os objetos que são o mapeamento para as tabelas)**

- Package “br.com.facdombosco.engb.dvo.acessos” -> Parte de segurança da aplicação

- Package “br.com.facdombosco.engb.dvo.vendas”

- Package “br.com.facdombosco.engb.dvo.servicos”

- Package “br.com.facdombosco.engb.dvo.financeiro”

- Package “br.com.facdombosco.engb.dvo.suprimentos”

- Package “br.com.facdombosco.engb.dvo.crm”

**Projeto “br.com.facdombosco.engb.service” (contém a regra de negócio da aplicação)**

- Package “br.com.facdombosco.engb.service.acessos” -> Parte de segurança da aplicação

- Package “br.com.facdombosco.engb.service.baseclasses”

- Package “br.com.facdombosco.engb.service.vendas”

- Package “br.com.facdombosco.engb.service.servicos”

- Package “br.com.facdombosco.engb.service.financeiro”

- Package “br.com.facdombosco.engb.service.suprimentos”

- Package “br.com.facdombosco.engb.service.crm”

**Projeto “br.com.facdombosco.engb.dao” (contém a regra de negócio da aplicação)**

- Package “br.com.facdombosco.engb. dao.acessos” -> Parte de segurança da aplicação

- Package “br.com.facdombosco.engb. dao.baseclasses”

- Package “br.com.facdombosco.engb. dao.vendas”

- Package “br.com.facdombosco.engb. dao.servicos”

- Package “br.com.facdombosco.engb. dao.financeiro”

- Package “br.com.facdombosco.engb. dao.suprimentos”

- Package “br.com.facdombosco.engb. dao.crm”

**Projeto “br.com.facdombosco.engb.dao” (contém as regras de persistência da aplicação)**

- Package “br.com.facdombosco.engb. dao.acessos” -> Parte de segurança da aplicação

- Package “br.com.facdombosco.engb. dao.baseclasses”

- Package “br.com.facdombosco.engb. dao.vendas”

- Package “br.com.facdombosco.engb. dao.servicos”

- Package “br.com.facdombosco.engb. dao.financeiro”

- Package “br.com.facdombosco.engb. dao.suprimentos”

- Package “br.com.facdombosco.engb. dao.crm”

**Projeto “br.com.facdombosco.engb.app” (contém a interface da aplicação)**

- Package “br.com.facdombosco.engb. app.acessos” -> Parte de segurança da aplicação

- Package “br.com.facdombosco.engb. app.vendas”

- Package “br.com.facdombosco.engb. app.servicos”

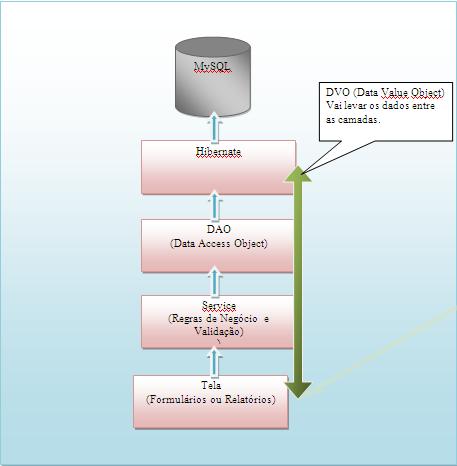
- Package “br.com.facdombosco.engb. app.financeiro”

- Package “br.com.facdombosco.engb. app.suprimentos”

- Package “br.com.facdombosco.engb. app.crm”

**Projeto “br.com.facdombosco.engb.framework” (contém classes utilitárias que podem ser compartilhadas entre todos os módulos)**

**Diagrama da Arquitetura**



A arquitetura consiste basicamente em 3 camadas de implementação, não levando em consideração o framework Hibernate e o banco de dados MySQL. A camada “DVO” contém todas as entidades de negócios que são as classes que representam as tabelas do banco de dados. A camada “DAO” é a camada responsável por realizar o acesso ao banco de dados , realizando queries e fazendo a persistência das informações. A camada “Service” é responsável por armazenar todas as regras de negócio, como validação de informações e etc. Conclui-se que:

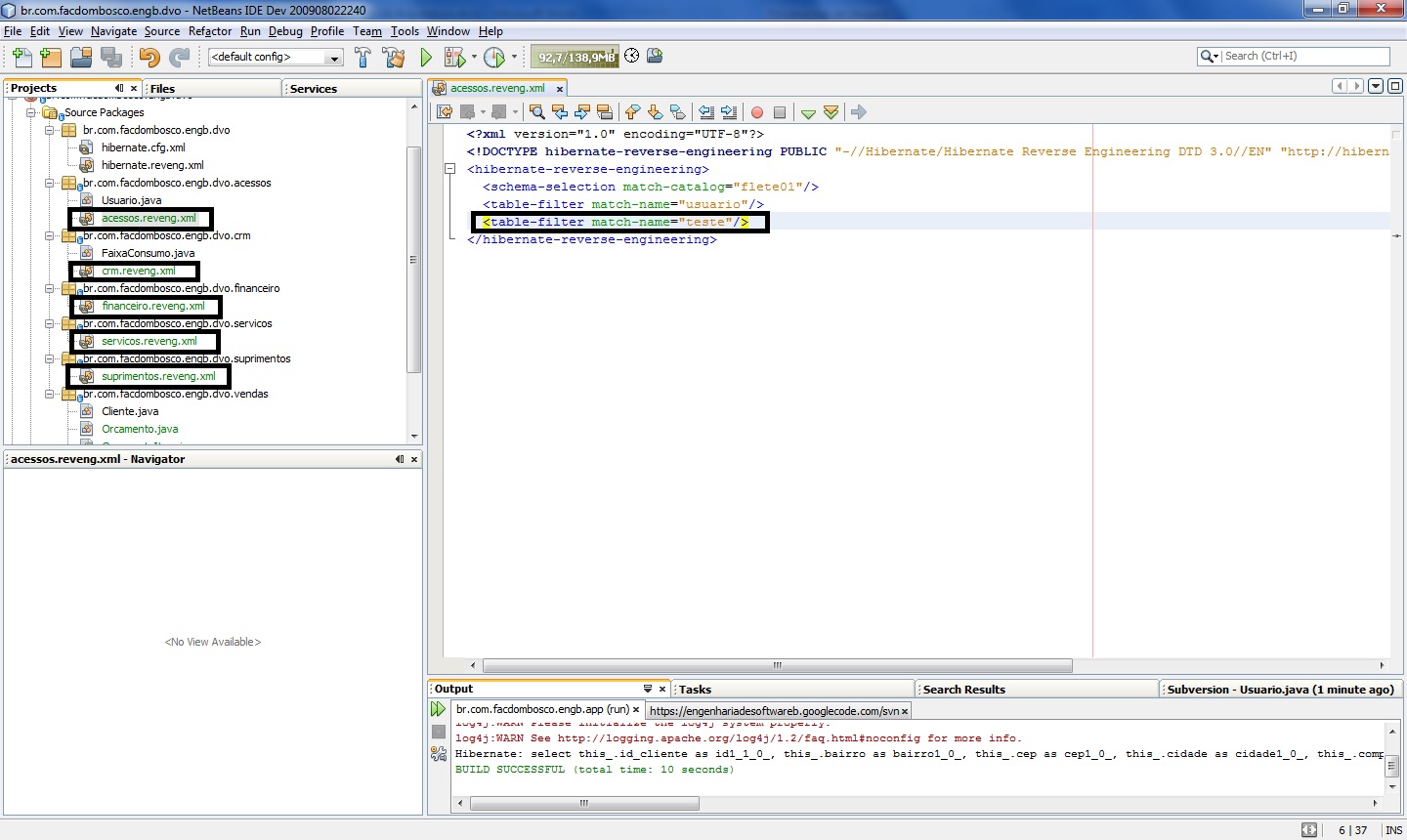
* A camada “DVO” é a camada de mapeamento das tabelas de banco de dados e serve para transportar as informações entre todas as camadas.
* A camada “Service” chama a camada “DAO”.
* As implementações das interfaces gráficas nunca devem chamar a camada “DAO” diretamente, sempre chamar a camada “Service” que, por conseqüência, chama a camada “DAO”.

**Passo a passo da implementação da arquitetura**

1. **Criação do DVO**

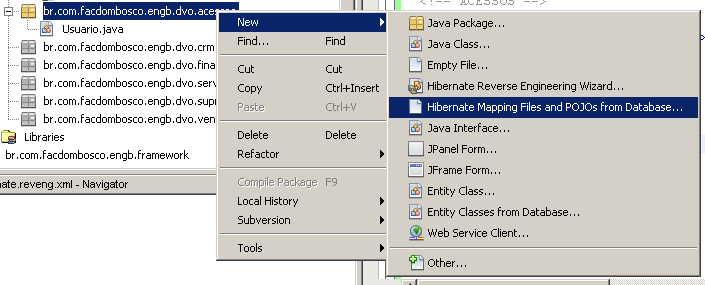
O primeiro passo a ser realizado para iniciar na arquitetura é incluir a classe “DVO”, que serve como mapeamento da tabela do banco de dados. Partindo do pressuposto que a tabela do banco de dados já está criada, a ferramenta “Netbeans”, gera a classe “DVO” automaticamente, contendo as informações necessárias (Annotations JPA) para informar ao framework “Hibernate” que a classe “DVO” é, um mapeamento para a tabela do banco de dados. Para que o Netbeans “enxergue” a tabela do banco de dados no momento de realizar a geração automática da classe “DVO”, é necessário incluir a referência para a tabela no arquivo que serve como assistente de geração de código. O arquivo assistente de geração de código chama-se “NOME\_MODULO.reveng.xml” (localizado dentro do projeto “br.com.facdombosco.engb.dvo”, dentro do respectivo pacote do módulo), e, ele pode ser gerado novamente pelo Netbeans a cada inclusão de uma tabela do banco de dados (a qual será gerado uma classe “DVO” respectiva) ou alterado manualmente incluindo a linha referente à a tabela. Para facilitar o desenvolvimento, as linhas serão incluídas manualmente nos arquivos. Existe um arquivo de configuração do assistente de geração de código por módulo para minimizar os conflitos e facilitar o desenvolvimento.

Partindo do pressuposto que a tabela “teste” já está criada no banco de dados, devemos incluir uma linha no arquivo “NOME\_MODULO.reveng.xml” referente ao módulo respectivo. Na figura abaixo estão destacados os arquivos em cada pacote do “DVO” e também a classe que está sendo incluída no arquivo assistente de geração de código do pacote “br.com.facdombosco.engb.dvo.acessos”:

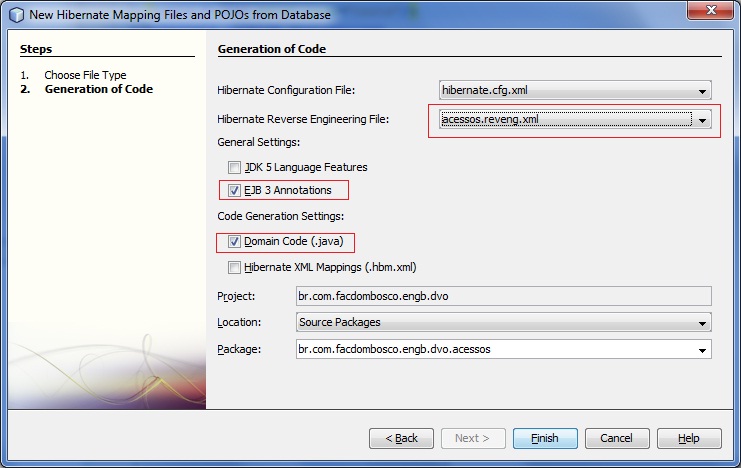


Após a inclusão da linha no arquivo assistente de geração de código do Hibernate, é possível gerar a classe de acordo com a tabela, para isto, deve-se seguir os seguintes passos:

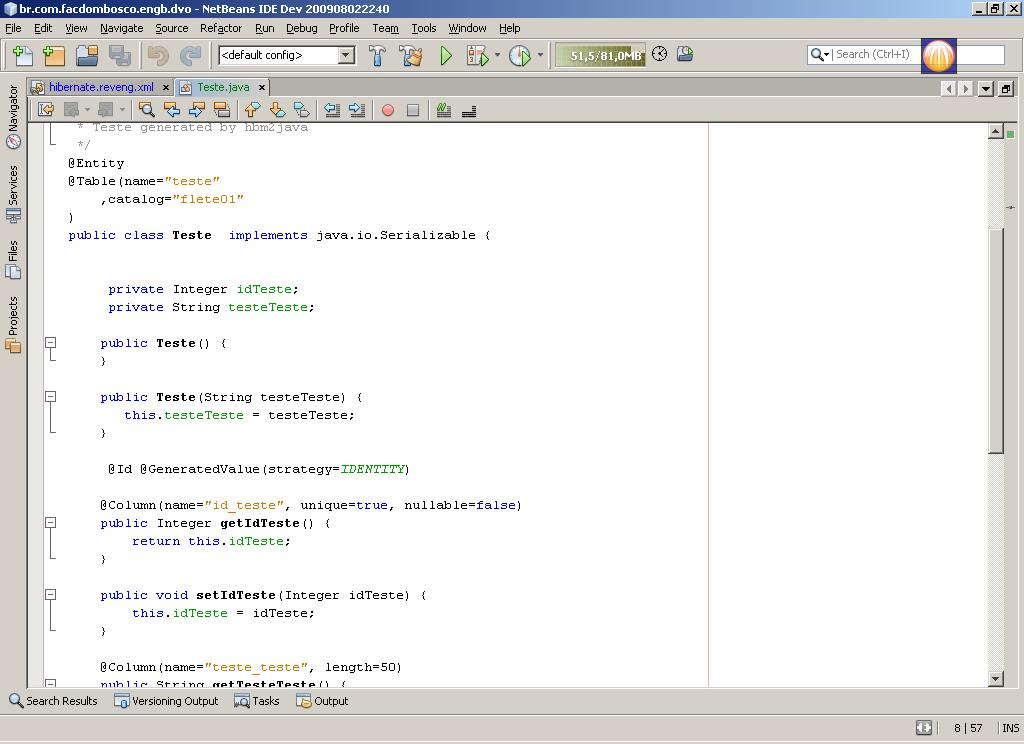
1 – Selecionar o pacote relacionado ao módulo com que o “DVO” será criado, clicar em “New” -> “Hibernate Mapping Files and POJOs from Database”.



2 – Marcar a opção “EJB 3 Annotations”, desmarcar a opção “Hibernate XML Mappings (.hbm.xml)” e selecionar o arquivo de assistente de geração de código respectivo ao módulo. Ao realizar esta configuração, a configuração do mapeamento da classe para tabela será realizada através de Annotations na própria classe e não, em arquivos XML, facilitando a manutenção posterior.

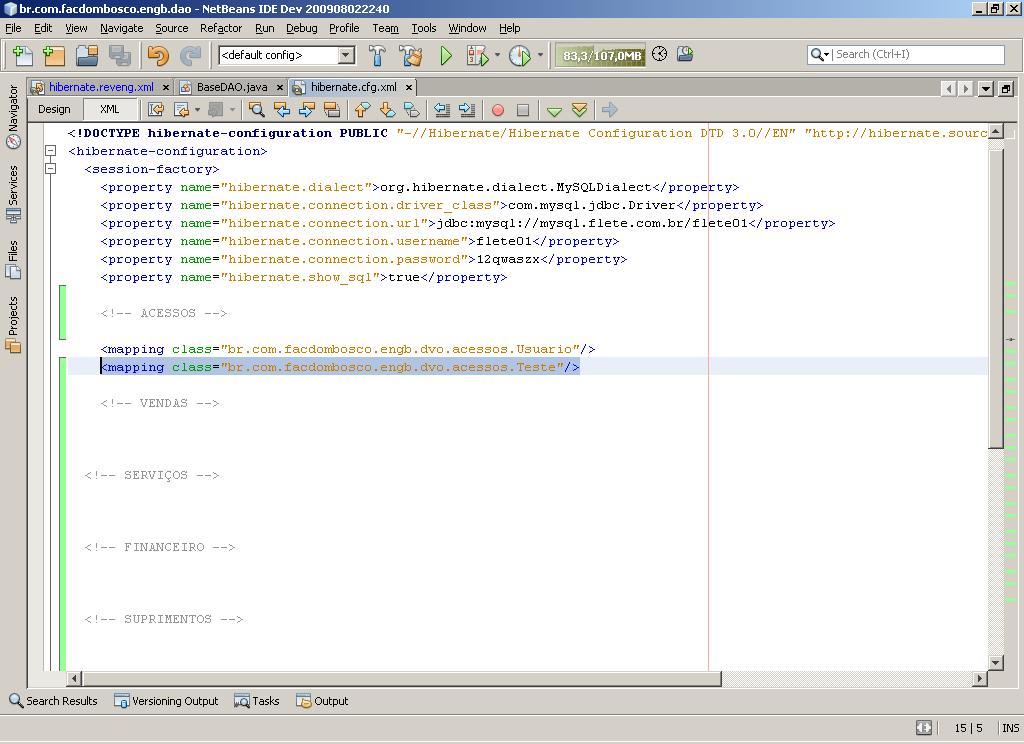


3 – Ao clicar em “Finish”, a classe será gerada automaticamente:



Se alguma alteração for realizada na tabela, o arquivo de mapeamento deve ser gerado novamente ou alterado manualmente respeitando o padrão de configuração das Annotations do Hibernate, para que fique atualizado em relação ao banco de dados.

4 – Atualizar o arquivo “hibernate.cfg.xml” (arquivo de configuração do Hibernate) que fica no projeto “br.com.facdombosco.engb.dao” (NÃO alterar o arquivo “hibernate.cfg.xml” que fica dentro do pacote “br.com.facdombosco.engb.dao”) na package “default” no módulo respectivo para que o mesmo possa identificar o mapeamento:



1. **Criação do DAO**

Para criar a classe DAO, é necessário incluir uma nova classe no projeto “br.com.facdombosco.eng.dao” dentro do pacote respectivo ao módulo. A classe criada deve herdar da classe tipada “BaseDAO”, que contém as operações em comum de CRUD já implementadas. Ou seja, as operações de busca e persistência já estão implementadas na classe “BaseDAO” com o objetivo de facilitar a implementação do sistema. Ao herdar da classe “BaseDAO” é necessário passar os tipos da “DVO” e da chave primária da classe (Integer, já que serão utilizadas chaves AutoIncrement no banco de dados). Abaixo um exemplo:



Qualquer método de busca ou persistência customizado deve ser implementado nesta classe.

1. **Criação do Service**

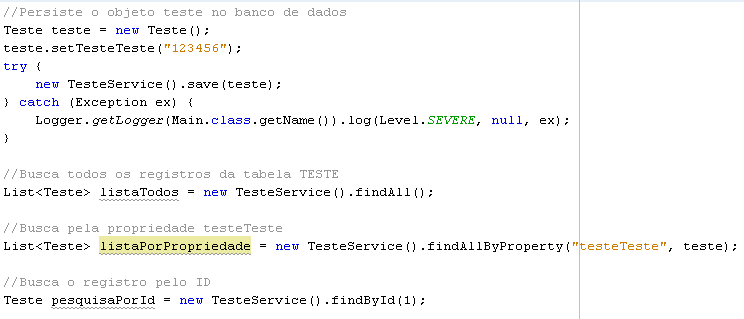
Para criar a classe DAO, é necessário incluir uma nova classe no projeto “br.com.facdombosco.eng.service” dentro do pacote respectivo ao módulo. A classe criada deve herdar da classe tipada “BaseService”, que já contém a implementação dos métodos de busca e persistência da classe “BaseDAO” com o objetivo de facilitar a implementação. Ao herdar da classe “BaseService” é necessário passar os tipos da “DVO” e da chave primária da classe (Integer, já que serão utilizadas chaves AutoIncrement no banco de dados), além de criar um construtor passando a classe respectiva ao “DVO” para que o Hibernate identifique o tipo necessário. Abaixo um exemplo:



Qualquer método de regra de negócio ou validação pode ser sobrescrito na classe “Service”. Abaixo um exemplo de validação de campos ao salvar um usuário:



1. **Utilização da arquitetura**



Lembrando que, ao chamar os métodos apresentados acima estão implementados na classe “BaseService”, que, por sua vez, chama a classe “BaseDAO”, com o objetivo de facilitar o desenvolvimento do sistema. Nota-se que, não foi necessário escrever nenhum comando SQL para realizar as operações CRUD. Nas classes “BaseDAO” e “BaseService” também estão implementados outros métodos para facilitar a realização do CRUD.