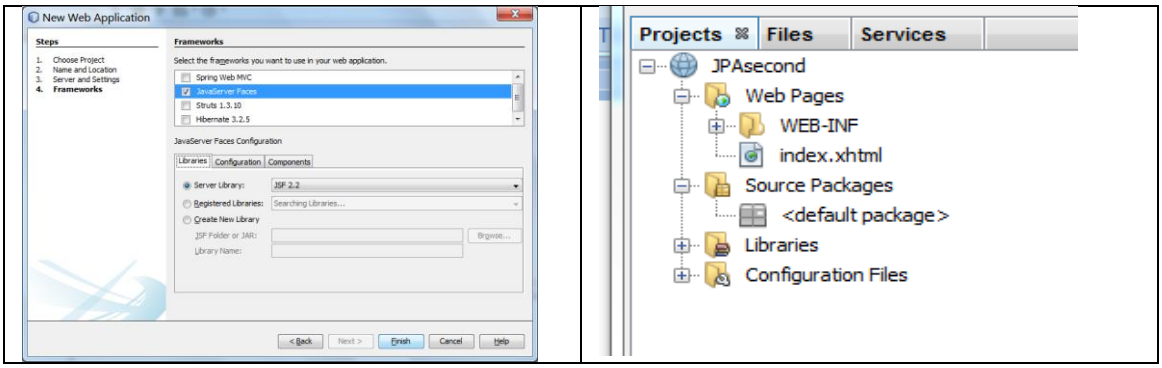
(Paula Prata)

Sessão Prática II – JPA “entities”e unidades de persistência

1 – Criar uma entity a partir de uma “web application” que usa a Framework “JavaServer Faces” (JSF)

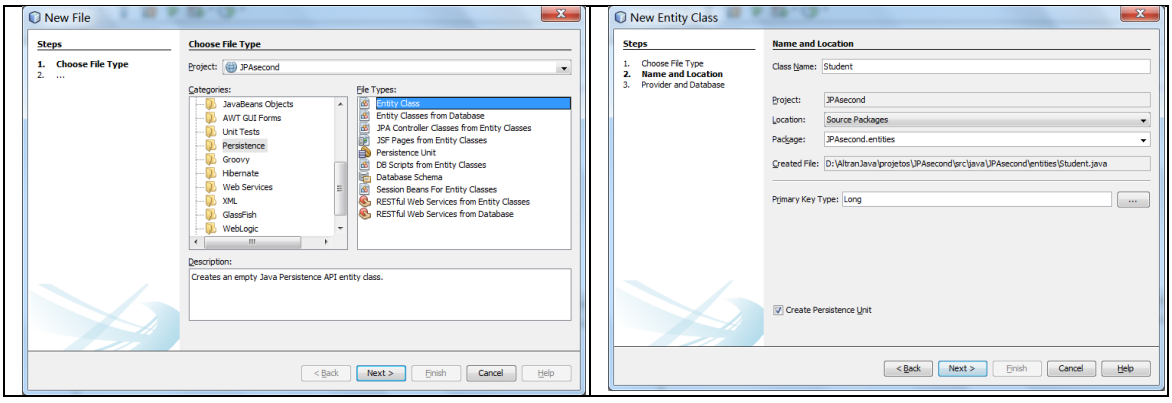
a) Criar um Web Application (JPAsecond) como anteriormente: New Project / Java Web / Web Application/ Next, atribuir o nome JPAsecond , Next / selecionar GlassFish Server, “Java EE 6 Web”, Next / selecionar a Framework JavaServer Faces com o ilustra a figura abaixo. Após terminar, observar a janela Projects onde foi criado o ficheiro index.xhtml.



b) Criar uma entity

- No menu File escolher New File. Selecionar no projeto JPAsecond, Persistence , Entity Class, Next. Para a nova entity class atribuir o nome Student, e o package JPAsecond.entities.

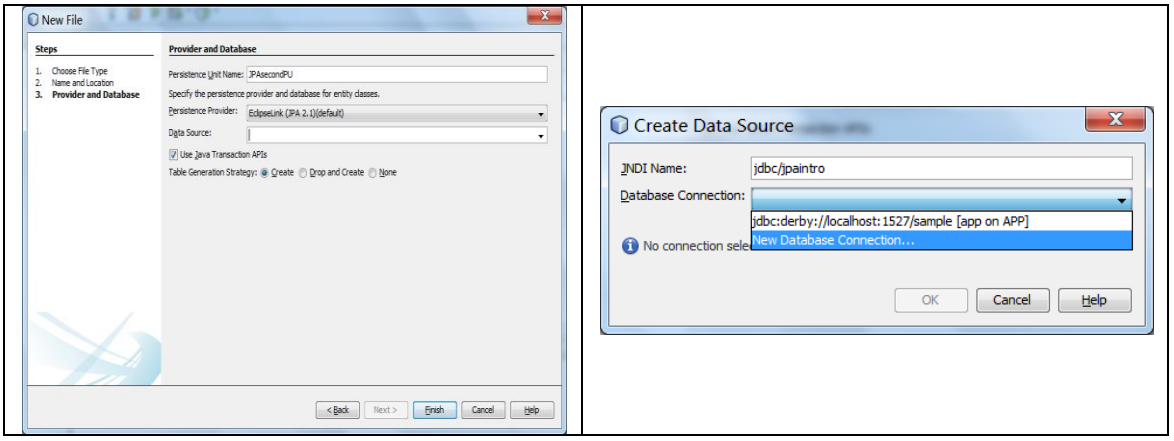
Projetos que usam JPA requerem uma unidade de persistência que será definida num ficheiro persistence.xml. Quando é criada a primeira entity, o NetBeans deteta que não existe o ficheiro e automaticamente seleciona a opção “Create Persistence Unit”.



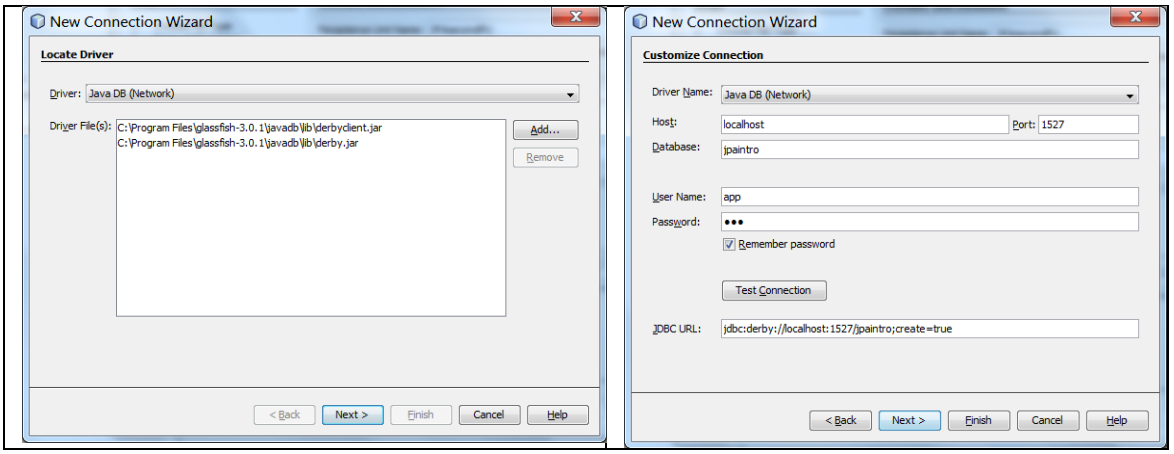
c) Criar a unidade de persistência

Após Next, aceitar o nome sugerido para a unidade de persistência JPAsecondPU. É agora necessário indicar um “persistence provider” e um Data Source que nos permita aceder à base de dados associada às classes entity. Para Persistence Provider usar o valor de omissão, EclipseLink. Para “Data Source” escolher a opção “New Data Source”.

- O acesso ao Data Source por JNDI permite obter uma conexão à base de dados. Atribuir o valor "jdbs/jpaintro" ao JNDI name do "data source" e escolher a opção New Database Connection".



- Ao criar uma nova conexão, escolher o Driver Java DB (Network) quem vem incluído no NetBeans.
- Após Next, configurar a conexão. Como o JavaDB está na mesma máquina o Host é "localhost", o porto de omissão do Java DB é o 1527 e queremos uma ligação a uma base de dados de nome "jpaintro". Especificar um username e password. Finalmente, como a base de dados ainda não existe vamos acrescentar ao JDBC URL o atributo *create = true* .



- Após Next, é pedido um database Scheme. Selecionar APP. Criado o data source e a connection, regressa-se à configuração da unidade de persistência.

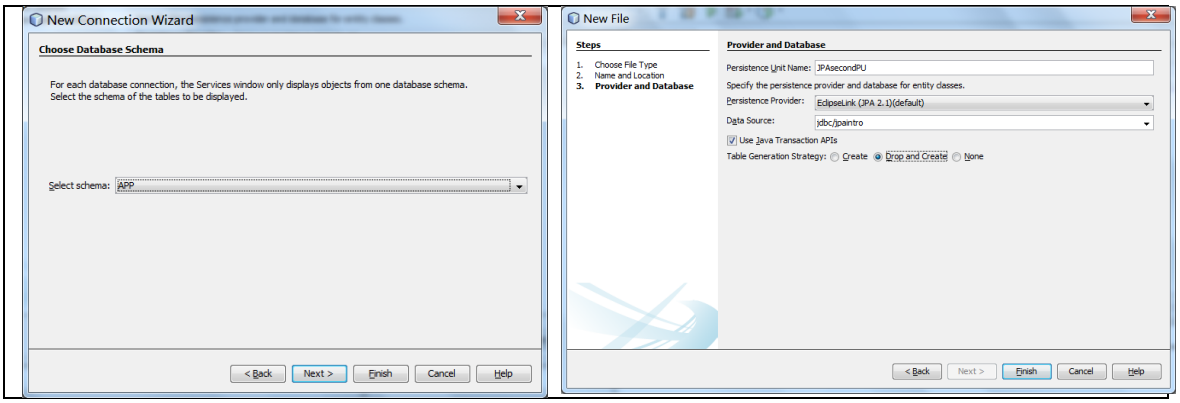
- Selecionar a opção "Use Java Transations API" o que permitirá usar a API de gestão de transações. (JTA – Java Transactions API).

Na estratégia de geração de tabelas é possível, escolher entre 3 opções: Create (as tabelas são criadas quando é feito o deploy da aplicação) Drop and Create (as tabelas são apagadas e reconstruídas quando é feito o deploy da aplicação) None (não são criadas tabelas).

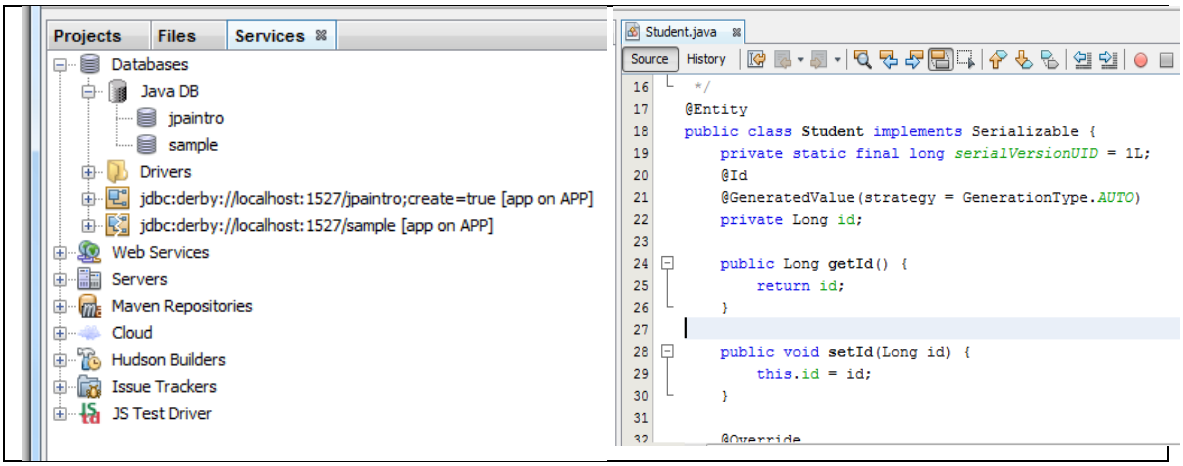
- Selecionar “Drop and Create” pois numa fase de desenvolvimento permite alterar os campos da entity sem alterar o esquema da base de dados. De notar que neste caso em cada novo deploy, os dados são perdidos.

Foi criado um novo “Data Source”, uma nova conexão à base de dados e uma nova unidade de persistência.

- Clicar em Finish para o NetBeans concluir a geração da entity Student.



- Na janela “Services” observar a base de dados jpaintro e a conexão criada.
- Na janela “projectos”, em package “Source Packages” “jpaseconf.entities” observar o ficheiro Student.java gerado.



Notas:

- A entity é uma classe java normal que implementa a interface Serializable.
- Para que uma objeto possa ser persistente numa base de dados a sua classe tem de ter a anotação @Entity.
- A anotação @Id indica o campo que é a chave primária.
- A chave primária pode ser gerada automaticamente. Existem várias estratégias de geração.

2 – Interagir com JPA entities através do Entity Manager

a) Adicionar novos campos à entity

A entity gerada, Student.java, possui um único campo (a chave primária) os métodos get e set e faz a sobreposição de alguns métodos herdados da classe Object (equals, toString e hasCode).

- Adicionar pelo menos mais dois campos (por exemplo nome e curso) e respetivos getters e setters. Pode fazê-lo manualmente seguindo as normas standard de codificação ou de forma automática através de "insert code".

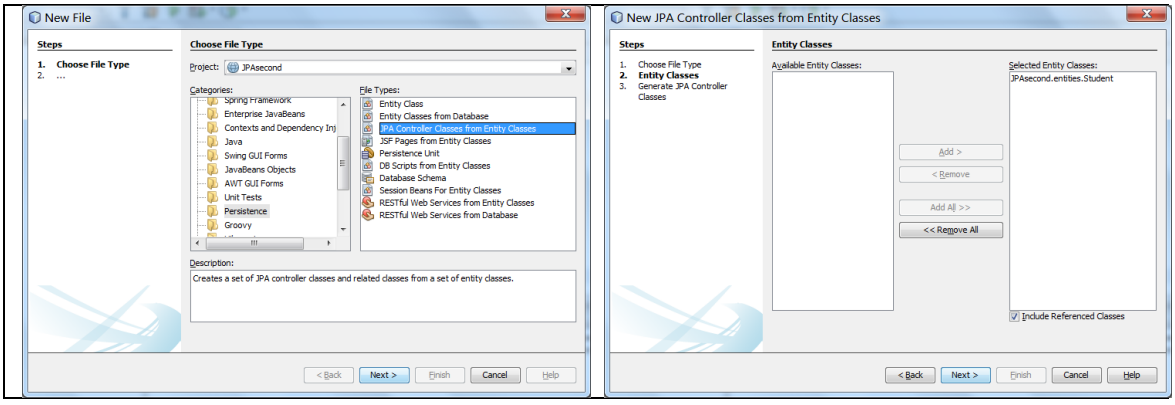
- Adaptar o método toString aos novos campos.

b) Gerar uma JPA "controller class" a partir de uma entity

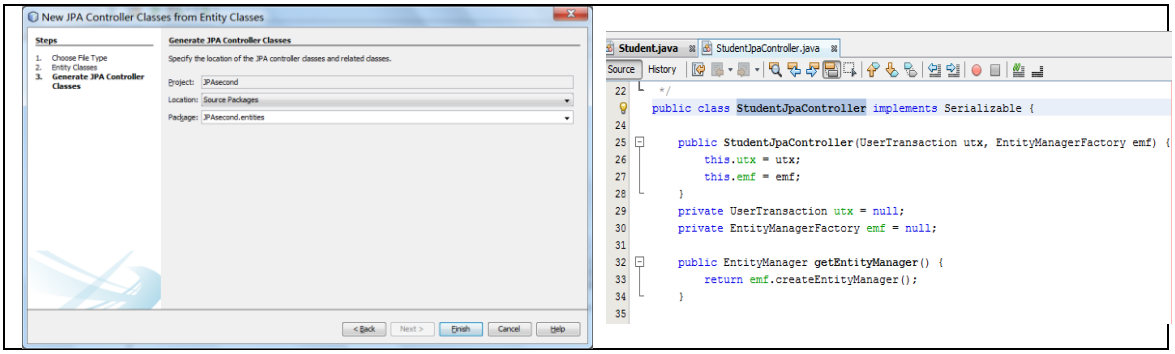
O padrão de desenho DAO (Data Access Object) consiste em colocar toda a funcionalidade de acesso à base de dados em classes separadas, isoladas de outras camadas da aplicação como a definição da interface, e da lógica de negócio.
O NetBeans permite gerar, a partir de entities, classes de controlo que seguem o padrão DAO.

- No menu File, escolher New File, projeto JPAsecond, selecionar Persistence/ JPA Controller Classes from Entity Classes.

- Após Next, selecionar a entity para a qual queremos gerar uma classe de controle.



- Selecionar o projeto e o package (caso ainda não o esteja). Após terminar, a classe StudentJpaController.java é gerada.



Observar que na classe foram gerados métodos para as operações CRUD (Create, Read, Update, Delete) clássicas.

O método *create(Student student)* permite criar uma instância da entity. Este método invoca o método *persist()* da EntityManager, que irá criar os dados da entity na base de dados.

São criados vários métodos para leitura. O método findStudent(Long id) invoca o método find() da EntityManager para obter da base de dados a instância correspondente ao id dado como parâmetro.

- Estude o método *findStudentEntities(boolean all, int maxResults, int firstResult).* Este método usa API *Criteria* para construir o query à base de dados.

O método de atualização é o método *edit(Student student)* e invoca o método *merge()* da EntityManager para actualizar a base de dados com o valor que recebe como parâmetro.

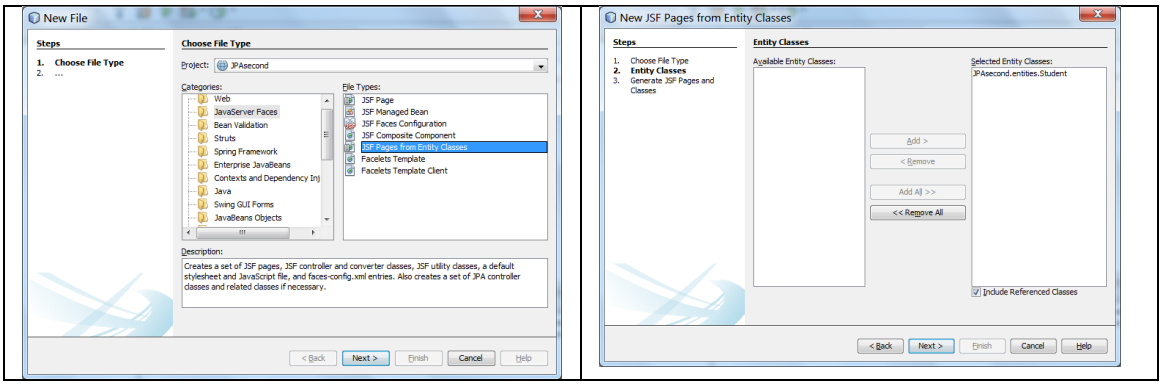
- O método para apagar é o método *destroy(Long id)* . Caso o objeto, identificado pela valor da chave primária que recebe como parâmetro, exista será eliminado da base de dados ao invocar o método *remove()* da EntityManager .

Neste momento temos todo o código necessário para efetuar as operações CRUD sobre a entity Student na base de dados. Basta invocar os métodos da classe StudentJpaControler.

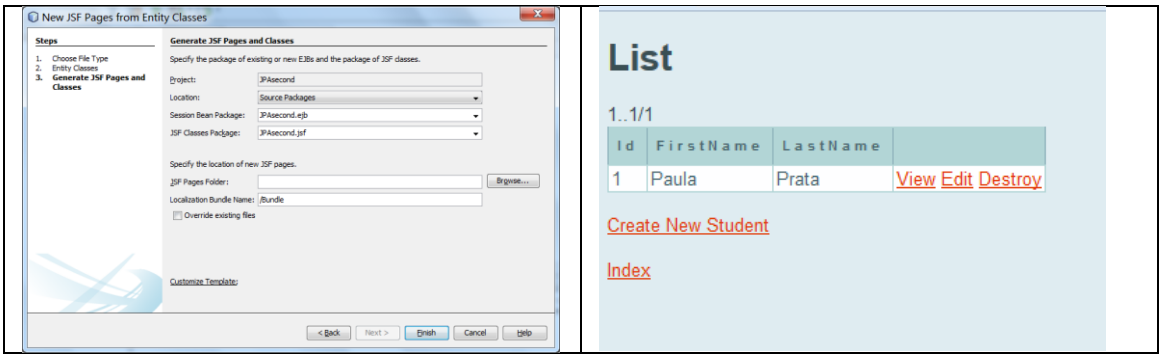
No entanto o processo pode ser ainda mais automatizado. Vamos ver com criar uma aplicação completa a partir de uma base de dados já existente. Neste caso, vamos usar uma base de dados muito simples, apenas tem uma tabela, mas o processo será o mesmo para bases de dados complexas.

3 - Gerar uma aplicação JSF completa a partir de JPA entities

- No projeto, selecionar File / new File / Java Server Faces / JSF from Entity Classes e Selecionar a entity Student .



- Especificar um package para os “managed Beans” a criar e outro para as classes JSF. Deixar em branco a localização das páginas JSF, por omissão serão criadas na pasta Web Pages. Após terminar, executar a aplicação. Foi criada uma aplicação web completa que permite efetuar as operações CRUD na(s) tabela(s) da base de dados considerada.



Na janela projeto observe as classes que foram criadas. O NetBeans cria, na pasta Web Pages, uma pasta para cada entity (neste caso apenas existia uma entity).

Observe que a classe StudentJpaControler não foi usada. O Wizard criou automaticamente uma classe de controlo para a entity.

4 – Construa uma aplicação web para criar, consultar, atualizar e apagar valores da base de dados exemplo do NetBeans “sample”

5 – Crie uma base de dados simples em MySQL e de seguida uma aplicação web para criar, consultar, atualizar e apagar valores dessa base de dados.