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.

Steps	Frameworks					Projects	86	Files	services	
1. Choose Project	Select the frameworks you want to use in your web application.				1 [IPAsecond				
 Name and Location Server and Settings 	Sorry Web MYC Sorry Web MYC Sorry State MYC Sorry			*		JFASECOND				
4. Frameworks				E		È 💽	h W	leb Pages		
					📋 🗄 🖳 WEB-INF					
					111	index subtral				
					111	index.xntmi				
	Server Library:	JSF 2.2			111	ė. 🃭	S	ource Pack	ages	
	Registered Libraries:	Searching Libraries			111	- T 4	_		- -	
	🗇 Greate New Library				111			<defaul< td=""><td>t package></td><td></td></defaul<>	t package>	
	JSF Folder or JAR:		Brows	ie	111	- ÷ 🔽	1.3	brarias		
	Library Name:				111	<u>ها</u> رو) LI	branes		
					111	👘 💽	C	onfiguratio	n Files	
					111		, -	oningen er en		
					111					
		<back next=""></back>	inish Cancel H	jelp						

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".

New File	Provider and Database	
Steps . Orose File Type . None and Lozaton . Provider and Database	Provider and Database Provider and Database Prostore (JN None: P.Accord V Sportly the persistence provider and database for entity classes. Everation: Provider and database for entity classes. Data Source: Data Source: Use gave Transaction APIs Table Generation Strategy:	Create Data Source
	< gade Hext > Cancel	

- 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*.

New Connection Wizard	New Connection Wizard
Locate Driver	Customize Connection
Driver: Java DB (Network)	Driver Name: Java DB (Network)
Driver File(s): C:\Program Files\glassfish-3.0.1\javadb\lb\glarbyclent.jar C:\Program Files\glassfish-3.0.3\javadb\lb\glarby.jar	dd Post: localhost Port: 1527 Database: jpaintro
	User Name: app Passgord: ●●●
	Test Connection JDBC URL: jdbc:derby://ocalhost:1527/jpaintro;create=true
< Back Next > Enish Cancel	Help Enish Cancel Help

- 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 <u>dados são perdidos</u>.

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.

New Connection Wizard	New File	×
Choose Database Schema	Steps	Provider and Database
For each database connection, the Services window only displays objects from one database schema. Select the schema of the tables to be displayed.	Choose File Type Name and Location Provider and Database	Persistence Link Hame: J#AsecondPU Specify the persistence provider and database for entity dasses. Persistence Provider: Edgesetrik (JPA 2.1)(default) • Data Sarcer:
		V Use Java Transaction APIs
		Table Generation Strategy: O Greate (a) Drop and Create (C) None
Select schema: APP		
< Back Next > Enish Cancel Help		<back next=""> Emith Cance Help</back>

- 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.

Ш				-			Trudent in a market
Ш	Projects	Files	Services 8	8			Student. Java 🕷
Ш	🖃 🗐 Datal	bases				Sou	irce History 📴 💀 📲 🕶 🔍 🤯 🤯 🚰 🎬 🎧 🔗 🈓 💆 🖄 😐 🥥 📄
Ш	i 📥 🖬 J	ava DB				16	L */
Ш		inaintro				17	@Entity
Ш		a samala				18	<pre>public class Student implements Serializable {</pre>
		sample				19	<pre>private static final long serialVersionUID = 1L;</pre>
Ш	😐 🕪 🗖	rivers				20	0Id
Ш	📄 🕀 🔁 je	dbc:derby:	//localhost:15	i27/jpaintro;create=true [a	app on APP]	21	<pre>@GeneratedValue(strategy = GenerationType.AUTO)</pre>
Ш	🗼 🗄 🔣 je	dbc:derby:	//localhost:15	27/sample [app on APP]		22	private Long id;
Ш	🗄 👧 Web	Services				23	
Ш	serve	ers				24	public Long getId() {
Ш	Maye	n Penosita	riec			25	return 1a;
Ш		iii Reposito i	ines			20	- 1
ш						28	
Ш	Huds	on Builders				29	this.id = id:
	🗄 📷 Issue	Trackers				30	
	🗄 🗛 JS Te	st Driver				31	
						32	Roverride

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.





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

New JPA Controller Class	es from Entity Classes	
Steps 1. Oncour File Type 2. Birthy Coases 3. Generate JPA Controller Classes	Senerate JPA Controller Classes Specify the location of the JPA controller classes and related classes. Specify The location of the JPA controller classes and related classes. Specify The location of the JPA controller classes and related classes. Pacigage: JPA scored methods	Studentjava m Studentjava m Source Hetary Weight Hetary Y Public class StudentJpaController implements Seriel Public class StudentJpaController implements Seriel Public StudentJpaController StudentJpaController implements Seriel Public StudentJpaController StudentJpaController Seriel StudentJpaController Implements Seriel Public StudentJpaController StudentJpaController Seriel StudentJpaController Implements Seriel Public StudentJpaController Implements Public StudentJpaController Seriel Public EntityManagerFactory emf = null; Implements Public EntityManager() (Implic <t< th=""></t<>

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 .

teps	Choose File Type	Steps	Entity Classes		
L. Choose File Type 2	Broject PAecond Categories: Dic Types: Woh Dic Stronged Bern Saviserver Face Saviser	Choose File Type Entity Classes Generate JSF Pages and Classes	A <u>v</u> alable Entity Classes:]	Selected Entity Classes: JPAsecond.entities.Student
	Been Valation Spring Framework Spring Framework Spring Framework Spring Framework Spring Framework Spring Framework Spring Current Spring Spri			Add > < <u>R</u> emove Add All >>	
	Swatkener Objects			<< kemove All	☑ Include Referenced Classes
			(< gack Next >	Enish Cancel Help

- 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.

New	v JSF Pages from Entit	y Classes		x									
Steps		Generate JSF Pages a	nd Classes		1.1.4								
1. Ch 2. Ent	toose File Type tity Classes	Specify the package of existing or new EJBs and the package of JSF classes.			List								
3. Ge	enerate JSF Pages and	Project:	3PAsecond										
_		Location:	Source Packages										
		Session Bean Package:	JPAsecond.ejb 🔹		11/	1							
		JSF Classes Package:	JPAsecond.jsf 🔹		I d	FirstName	LastName						
		Specify the location of new	w JSF pages.				-						
		1SF Pages Folder:	Brg	vse	1	Paula	Prata	View Edit Destroy					
		Localization Bundle Name:	/Bundle										
		Override existing files			Crea	to New Student							
					orea	te New Oludein							
		Customize Template:			<u>Inde</u>	2							
			<back next=""> Enish Cancel</back>	Help									

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.