Configurando o Hibernate e Acessando o Postgres no Eclipse

(versão 1.0)

Prof. Julio Cesar Nardi.

Passo 1 – Crie um Dynamic Web Projet;

Passo 2 – Baixe os JARs do Hibernate em https://hibernate.org/orm/releases/5.5/.

Passo 3 – Descompacte o arquivo com os JARs do Hibernate em algum diretório na sua máquina.

Passo 4 – Vá até seu projeto no Eclipse → clique com o botão direito no nome do projeto → escolha *properties* → escolha Java Build Path → selecione *classpath* na caixa de seleção ao lado. Em seguida, escolha e inclua os JARs do Hibernate que estão no diretório do Passo 3. Para tanto, selecione o botão *Add External JARs…*.

Passo 5 – Baixe o JAR para conexão JDBC para o Postgres ou para o banco de dados que esteja utilizando. Para tanto, acesse a página https://jdbc.postgresql.org/download.html. Em seguida, copie este JAR para um diretório para utilizá-lo em seguinda.

Passo 6 - Vá até seu projeto no Eclipse → clique com o botão direito no nome do projeto → escolha *properties* → escolha Java Build Path → selecione *classpath* na caixa de seleção ao lado. Em seguida, escolha e inclua o JAR de conexão JDBC do Postgres que está no diretório do Passo 5. Para tanto, selecione o botão *Add External JARs…*.

Até o momento, já incluímos as bibliotecas necessárias do Hibernate e do Postgres em nosso projeto. Agora, vamos configurar o Hibernate para acessar seu banco de dados no Postgres.

Passo 7 – Esteja certo que o seu banco de dados esteja criado no Postgres.

Passo 8 – Vá até seu projeto no Eclipse → selecione o pacote *src/main/java* → clique com o botão direito neste diretório → selecione *New* → *Hibernate Configuration File* (cfg.xml).

Passo 9 – Abra o arquivo e verifique que este arquivo tenha, pelo menos o seguinte conteúdo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
                "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
                "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
        <property name="hibernate.connection.password">123</property>
        <property name="hibernate.connection.url">jdbc:postgresql:primeiroprojetoBD</property>
        <property name="hibernate.connection.username">postgres</property>
        <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
        <property name="hibernate.hbm2ddl.auto">update</property>

        <mapping class="primeiroprojeto.model.domain.Usuario"/>

    </session-factory>
</hibernate-configuration>
```

Obs.: Algumas informações deverão ser adaptadas de acordo com o seu projeto. Dentre elas, a senha e o usuário do banco de dados, o nome do banco de dados e as classes correspondentes a serem mapeadas.

Passo 10 – Crie a seguinte classe no diretório `primeiroprojeto.model.domain`. Observe que esta classe é a mesma cujo mapeamento foi incluído no arquivo **hibernate.cfg.xml**.

```
package primeiroprojeto.model.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;

@Entity
public class Usuario {

	@Id
	@GeneratedValue (strategy = GenerationType.SEQUENCE)
	private Long id;

	public Long getId() {
		return id;
	}

	public void setId(Long id) {
		this.id = id;
	}
}
```

Passo 11 – Crie a seguinte classe no pacote `primeiroprojeto.` Essa classe possui código de conexão com o banco de dados e um teste de criar e salvar um determinado objeto Usuario. Se tudo estiver OK, o esquema do banco de dados será criado, incluindo a tabela Usuario e um novo registro será incluído na tabela.

```
package primeiroprojeto;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import primeiroprojeto.model.domain.Usuario;

public class TesteConexao {

	public static void main(String[] args) {
		// TODO Auto-generated method stub


		StandardServiceRegistry ssr = new
StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();
	 Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();


	 SessionFactory factory = meta.getSessionFactoryBuilder().build();
	 Session session = factory.openSession();
	 Transaction t = session.beginTransaction();

	 session.save(new Usuario());

	 t.commit();

	 session.close();
	 factory.close();
	 ssr.close();


	}
}
```

Bom trabalho.