Universidade Federal do Paraná
Tutorial: Hibernate 5/Netbeans 12.6
Prof. Dr. Razer Anthom Nizer Rojas Montaño
razer@ufpr.br

Projeto Java Web com Hibernate sem o plug-in do Netbeans.

1) Versões usadas: JDK 11, Netbeans 12.6, Glassfish 6.2.3, Hibernate 5.6, PostgreSQL 12

2) Instalar o PostgreSQL e adicionar tabela tb_aluno em um banco de dados:

```
CREATE TABLE tb_aluno
(
    id_aluno serial PRIMARY KEY,
    nome_aluno character varying(50) NOT NULL
)
```

Esse comando cria uma *sequence* chamada ***tb_aluno_id_aluno_seq***, que será usada no Hibernate

3) Baixar biblioteca do Hibernate e Bean Validation

Hibernate 5.6: https://hibernate.org/orm/releases/5.6/

Bean Validation API 2.0.1:
https://mvnrepository.com/artifact/javax.validation/validation-api/2.0.1.Final

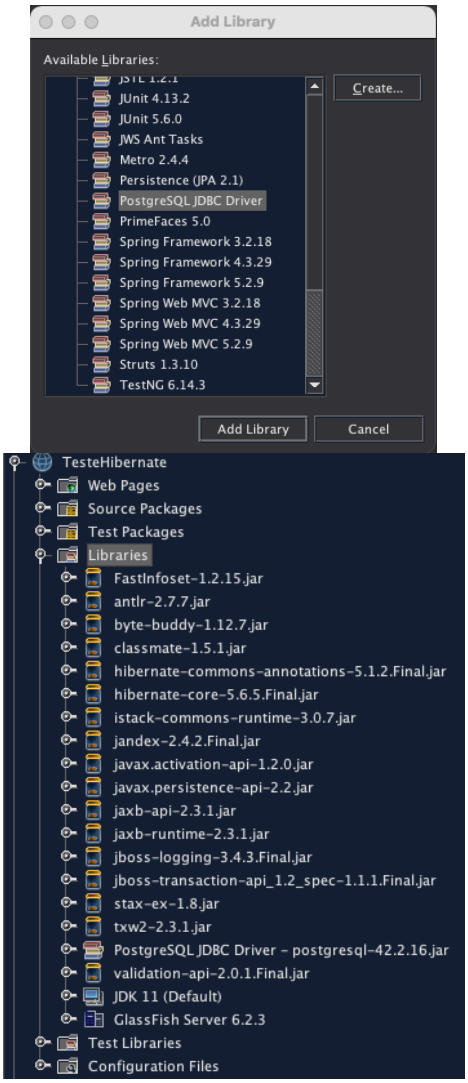
4) Descompactar os arquivos baixados para poder adicionar os arquivos .jar

5) Criar um novo projeto Web no Netbeans

6) Adicionar as bibliotecas:

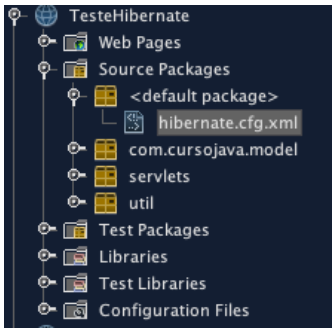
- Hibernate: todas que estiverem na pasta ***lib/required***
- Bean validation: ***validation-api-2.0.1.Final.jar***
- Driver do PostgreSQL: Botão direito em Libraries | Opção Add Library | Escolher PostgreSQL JDBC Driver

Prof. Dr. Razer Anthom Nizer Rojas Montaño
razer@ufpr.br



7) Criar o arquivo de configuração no *default package*, fora de qualquer pacote: **hibernate.cfg.xml**

- Acertar as configurações para acesso ao seu banco de dados

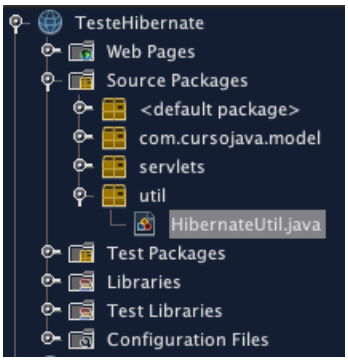



```
<?xml version='1.0' encoding='utf-8'?>
```

```
<!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.url">
            jdbc:postgresql://localhost:5432/razer
   </property>
   <property name="hibernate.connection.username">postgres</property>
      <property
name="hibernate.connection.password">postgres</property>
      <property name="hibernate.connection.pool_size">1</property>
      <property name="hibernate.dialect">
            org.hibernate.dialect.PostgreSQLDialect
      </property>
      <property name="hibernate.current_session_context_class">
            thread
      </property>
      <property name="hibernate.cache.provider_class">
            org.hibernate.cache.NoCacheProvider
      </property>
<property name="hibernate.show_sql">true</property>

<!-- Class Mapping -->
<mapping class="com.cursojava.model.Aluno"/>

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

8) Criar a classe ***HibernateUtil*** no pacote ***util***

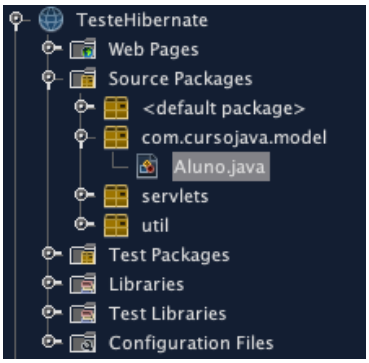
Prof. Dr. Razer Anthom Nizer Rojas Montaño
razer@ufpr.br




```
package util;

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

public class HibernateUtil {
  private static StandardServiceRegistry registry;
  private static SessionFactory sessionFactory;

  public static SessionFactory getSessionFactory() {
      if (sessionFactory == null) {
          try {
              registry = new StandardServiceRegistryBuilder().
                                                  configure().build();
              MetadataSources sources = new MetadataSources(registry);
              Metadata metadata = sources.getMetadataBuilder().build();
              sessionFactory = metadata.getSessionFactoryBuilder().
                                                  build();
          }
          catch (Exception e) {
            e.printStackTrace();
            if (registry != null) {
                StandardServiceRegistryBuilder.destroy(registry);
            }
          }
      }
      return sessionFactory;
  }
  public static void shutdown() {
      if (registry != null) {
          StandardServiceRegistryBuilder.destroy(registry);
      }
  }
}
```

9) Criar a classe persistente ***Aluno*** no pacote ***com.cursojava.model***:

Prof. Dr. Razer Anthom Nizer Rojas Montaño
razer@ufpr.br

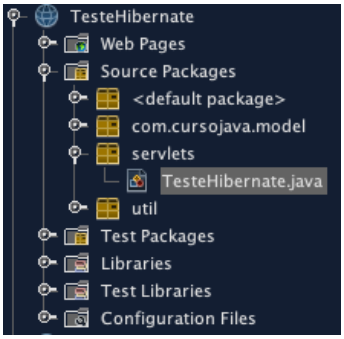



```
package com.cursojava.model;

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

@Entity
@Table(name="tb_aluno")
@SequenceGenerator(name="seq", sequenceName="tb_aluno_id_aluno_seq",
allocationSize=1)
public class Aluno {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator="seq")
    @Column(name = "id_aluno")
    private int id;
    @Column(name = "nome_aluno")
    private String nome;

    public Aluno() {
    }
    public Aluno(int id, String nome) {
        this.id = id;
        this.nome = nome;
    }
    public Aluno(String nome) {
        this.nome = nome;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

10) Criar a Servlet TesteHibernate no pacote servlets:

Prof. Dr. Razer Anthom Nizer Rojas Montaño
razer@ufpr.br

- Alterar somente o método ***processRequest()*** conforme abaixo. Manter o ***doGet()*** e ***doPost()***.
- Preste atenção nas importações.




```
package servlets;

import com.cursojava.model.Aluno;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.util.List;
import org.hibernate.Session;
import org.hibernate.Transaction;
import util.HibernateUtil;
@WebServlet(name = "TesteHibernate", urlPatterns =
{"/TesteHibernate"})
public class TesteHibernate extends HttpServlet {

    protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
            throws ServletException, IOException {

        Aluno a = new Aluno("Razer");
        Aluno b = new Aluno("Jaime");

        Transaction transaction = null;
        try (Session session = HibernateUtil.getSessionFactory().
                                                openSession()) {

            transaction = session.beginTransaction();

            session.save(a);
            session.save(b);

            transaction.commit();
        } catch (Exception e) {
```

```
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }

        response.setContentType("text/html;charset=UTF-8");
        try ( PrintWriter out = response.getWriter();
              Session session = HibernateUtil.getSessionFactory().
                                                openSession()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Alunos</title>");
            out.println("</head>");
            out.println("<body>");

            List <Aluno> alunos = session.createQuery(
                                    "FROM Aluno", Aluno.class).list();
            for (Aluno x: alunos) {
                out.println("<h2>" + x.getNome() + "</h2>");
            }
            out.println("</body>");
            out.println("</html>");
        }
        catch (Exception e) {
            e.printStackTrace();
            if (transaction != null) {
                transaction.rollback();
            }
        }
    }

  // Aqui são mantidos os métodos doGet() e doPost()

}
```

11) Execute a servlet.