Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

18 - CONVERTER

1. Conversores

1.1. Crie o pacote `br.com.etechoracio.training.view.converter`.

1.2. Crie a classe `CpfConverter` no pacote recém-criado.

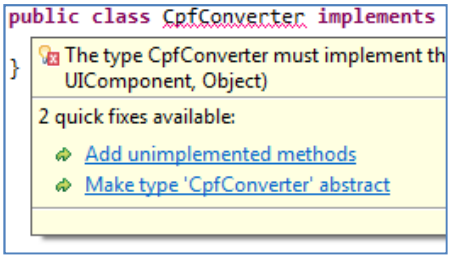
1.3. Anote a classe com `@Componente(value="cpfConverter")`.

```
@Component(value="cpfConverter")
public class CPFConverter {

}
```

1.4. A classe deverá implementar a *interface* `Converter`.

```
@Component(value="cpfConverter")
public class CPFConverter implements Converter {

}
```

1.5. Posicione o mouse sobre o erro informado e adicione os métodos não implementados.



1.6. Dentro do método `getAsObject()` implemente o código abaixo:

```
String cpf = value;

if (cpf != null && !cpf.equals("")) {
    cpf = value.replaceAll("[^0-9]", "");
}

return cpf;
```

Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

1.7. Dentro do método `getAsString()` implemente o código abaixo:

```
String cpf = String.valueOf(value);
if (cpf != null && cpf.length() == 11) {
    cpf = cpf.substring(0, 3) + "." +
          cpf.substring(3, 6) + "." +
          cpf.substring(6, 9) + "-" +
          cpf.substring(9, 11);
}
return cpf;
```

1.8. Idente a classe com `Ctrl` + `Shift` + `F` e salve.

1.9. Faça, também, os conversores `CEPConverter` e `TelefoneConverter`.

1.10. Abra a página **save.xhtml** através da combinação `Ctrl` + `Shift` + `R` para localização.

1.11. Localize o campo **CPF** e adicione a propriedade `converter` abaixo:

```
<h:outputText value="CPF:" />
<p:inputMask
       value="#{alunoMB.edit.cpf}"
       mask="999.999.999-99"
       style="width:140px;"
       label="CPF"
       required="true"
       converter="#{cpfConverter}"/>
```

1.12. Repita a operação para os campos **Telefone** e **CEP**.

1.13. Abra a classe **`Pc2Exercicio13Application`** através da combinação `Ctrl` + `Shift` + `R` para localização.

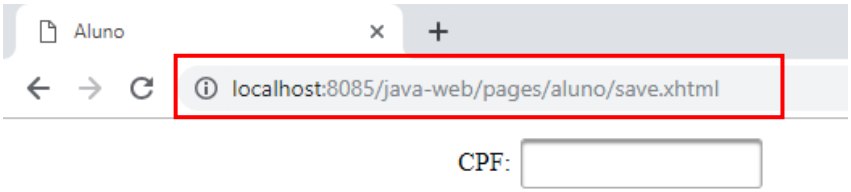
1.14. Para executar o código, clique com o botão direito do mouse na classe e selecione as opções ***Run As*** ➔ ***Java Application***.

1.15. Observe novamente a inicialização do *framework* Spring:

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.3.RELEASE)

27/03/2019 20:26:37 - Starting Pc2Exercicio13Application on LI03-22 with PID 6444 (F:\etec\Aulas\
27/03/2019 20:26:37 - No active profile set, falling back to default profiles: default
27/03/2019 20:26:37 - Bootstrapping Spring Data repositories in DEFAULT mode.
27/03/2019 20:26:38 - Finished Spring Data repository scanning in 66ms. Found 2 repository interf
27/03/2019 20:26:38 - @Bean method ScopeConfig.getScopeConfigurer is non-static and returns an ob
27/03/2019 20:26:38 - Bean 'org.springframework.transaction.annotation.ProxyTransactionManagement
27/03/2019 20:26:40 - Tomcat initialized with port(s): 8085 (http)
```

Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

1.16. Acesse o navegador de sua preferência com o endereço **localhost**, **porta** e **contexto** informados.

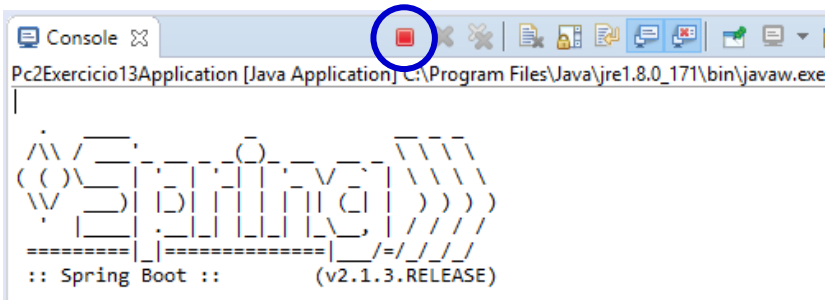


1.17. Preencha as informações e clique no botão **Cadastrar**.

1.18. Observe que no *console* do Eclipse aparecerá o erro abaixo:

```
Caused by: com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolationException: Column 'DT_CRIACAO' cannot be null
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance(Unknown Source)
        at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(Unknown Source)
        at java.lang.reflect.Constructor.newInstance(Unknown Source)
        at com.mysql.jdbc.Util.handleNewInstance(Util.java:406)
        at com.mysql.jdbc.Util.getInstance(Util.java:381)
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:1015)
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:956)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3491)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3423)
        at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:1936)
        at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2060)
        at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2542)
        at com.mysql.jdbc.PreparedStatement.executeInternal(PreparedStatement.java:1734)
        at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:2019)
        at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:1937)
        at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:1922)
        at org.hibernate.id.IdentityGenerator$GetGeneratedKeysDelegate.executeAndExtract(IdentityGenerator.java:94)
        at org.hibernate.id.insert.AbstractReturningDelegate.performInsert(AbstractReturningDelegate.java:57)
        ... 54 more
```

1.19. Pare a execução do projeto.



Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

2. PrePersist

O problema é que o campo **dataCriacao** é obrigatório na base de dados. Como não é um campo que NÃO será preenchido em tela, resolveremos o problema com a anotação `@PrePersist`.

2.1. Pare a execução do projeto e abra a classe **Aluno** dentro do pacote **model** e insira o método abaixo:

```
private void preencherDataCriacao(){
    if(dataCriacao == null) {
        dataCriacao = new Date();
    }
}
```

2.2. Anote-o com `@PrePersist` para realizar o preenchimento antes da inserção na base de dados.

```
@PrePersist
private void preencherDataCriacao(){
    if(dataCriacao == null) {
        dataCriacao = new Date();
    }
}
```

2.3. Idente a classe com `Ctrl` + `Shift` + `F` e salve.

2.4. Faça o mesmo procedimento na classe **Endereco** para o campo **UF** (valor padrão “SP”).

2.5. Abra a classe **`Pc2Exercicio13Application`** através da combinação `Ctrl` + `Shift` + `R` para localização.

2.6. Para executar o código, clique com o botão direito do mouse na classe e selecione as opções ***Run As*** ➔ ***Java Application***.

2.7. Preencha as informações e clique no botão **Cadastrar**.

2.8. Faça um *SELECT* na tabela **TBL_ALUNO** e **TBL_ENDERECO** para validar o cadastramento.

Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

3. Github

3.1. Acesse a pasta do projeto (via **Prompt de Comando**).

3.2. Verifique a situação dos arquivos no repositório Git.

```
git status
```

3.3. Faça com que os arquivos sejam rastreados pelo Git.

```
git add .
```

3.4. Verifique a situação dos arquivos no repositório Git novamente.

3.5. Execute o **commit** para gravar as mudanças no repositório com a mensagem **"Ex. 18 - Converter"**.

3.6. Verifique a situação do arquivo no repositório Git novamente.

3.7. Execute o **push** para incluir seu código no repositório remoto.