15

(Para saber mais) Configurando o JSF para usar o TimeZone da nossa máquina!

Nosso sistema evoluiu e agora um novo requisito surgiu: a necessidade de salvar também o **horário** de lançamento do livro. Para isso, precisaremos fazer algumas alterações em nosso projeto:

1) Vamos alterar o padrão do conversor do `h:inputText` já que agora ele irá receber também o horário:

```
<h:inputText id="dataLancamento" value="#{livroBean.livro.dataLancamento.time}">
    <f:convertDateTime pattern="dd/MM/yyyy HH:mm" timeZone="America/Sao_Paulo" />
</h:inputText>
```

2) Também precisamos alterar o valor da saída, já que precisamos exibir o horário cadastrado. Ou seja, vamos alterar o valor do conversor também no `h:outputText`

```
<h:outputText value="#{livro.dataLancamento.time}">
    <f:convertDateTime pattern="dd/MM/yyyy HH:mm" timeZone="America/Sao_Paulo" />
</h:outputText>
```

3) Configurar a JPA para armazenar também o horário além da data.

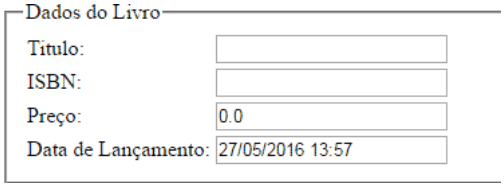
Na classe `Livro`, troque o valor da anotação **@Temporal** de `DATE` para `TIMESTAMP`:

```
@Entity
public class Livro implements Serializable {
    // outros atributos

    @Temporal(TemporalType.TIMESTAMP)
    private Calendar dataLancamento = Calendar.getInstance();

    // outros atributos
    // getters e setters
}
```

Reinicie o Tomcat e veja que agora que o campo **Data de lançamento** recebe também o horário conforme a imagem abaixo:



Mas, por que será que precisamos colocar o atributo `timeZone` em todos os nossos `f:convertDateTime`? Vamos conferir! **Remova** os atributos e recarregue a página. O que aconteceu?

Provavelmente o horário exibido agora no campo não é mais o mesmo horário marcado em seu computador (*timezone America/Sao_paulo*). Por que será que isso aconteceu?

Isso acontece porque o JSF usa por padrão o *timezone* **GMT** (UTC), como você pode conferir no código da classe `DateTimeConverter` clicando aqui (http://grepcode.com/file/repo1.maven.org/maven2/org.glassfish/javax.faces/2.2.12/javax/faces/convert/DateTimeConverter.jav Por isso precisamos informar o `timeZone` toda vez que criamos um novo converter. Mas será que não há nenhuma forma de centralizar essa informação?

Há sim e muito fácil! Podemos dizer ao JSF que queremos utilizar o mesmo *TimeZone* usado pelo nosso sistema operacional. E para isso vamos declarar um novo parâmetro no arquivo `web.xml` :

```
<context-param>
    <param-name>
        javax.faces.DATETIMECONVERTER_DEFAULT_TIMEZONE_IS_SYSTEM_TIMEZONE
    </param-name>
    <param-value>
        true
    </param-value>
</context-param>
```

Pronto! Reinicie o Tomcat e veja que o horário na aplicação está equivalente ao horário do sistema operacional mesmo sem utilizar o atributo `timeZone` .