Hi,

I've found some issues in **JSF 2.3-m04** regarding to `<f:convertDateTime />`:

1. The converter won't work with `type="localTime"` and `timeStyle`. This seems to be a typo in source code, the `dateStyle` is used instead of `timeStyle`:

```
// Mojarra 2.3-m04 source code
public class DateTimeConverter implements Converter, PartialStateHolder {

 ...
 private String dateStyle = "default";
 private String timeStyle = "default";
 ...

 private FormatWrapper getDateFormat(Locale locale) {
  ...
  } else if (type.equals("localTime")) {
     dtf = (null != pattern) ? DateTimeFormatter.ofPattern(pattern, locale) :
            DateTimeFormatter.ofLocalizedTime(getFormatStyle(dateStyle));
     from = LocalTime::from;
     ...
  }
  ...
}
```

As a consequence, you can write `<f:convertDateTime type="localTime" dateStyle="short" />` and it will work.

2. You can't format only time when `type="localDateTime"`:

Consider the following example:

```
<h:outputText value="#{myBean.localDateTime}">
 <f:convertDateTime type="localDateTime" timeStyle="short" />
</h:outputText>
```

Similar to first issue, the value of the `dateStyle` attribute will be used instead of `timeStyle`.

As a consequence, you can't "combine" `timeStyle` and `dateStyle` when `type="localDateTime"`. A `LocalDateTime` object holds both date and time. So we should be able to format the date part with one style and the time part with another style. For example:

```
<h:outputText value="#{myBean.localDateTime}">
 <f:convertDateTime type="localDateTime" dateStyle="medium" timeStyle="short" />
</h:outputText>
```

This will not work as expected. During the parsing and formatting process, the `timeStyle` attribute will be ignored. Only `dateStyle` will be used:

```
// Mojarra 2.3-m04 source code
public class DateTimeConverter implements Converter, PartialStateHolder {

 ...
 private String dateStyle = "default";
 private String timeStyle = "default";
 ...

 private FormatWrapper getDateFormat(Locale locale) {
  ...
```

```
  } else if (type.equals("localDateTime")) {
     dtf = (null != pattern) ? DateTimeFormatter.ofPattern(pattern, locale) :
            DateTimeFormatter.ofLocalizedDateTime(getFormatStyle(dateStyle));
     from = LocalDateTime::from;
  }
  ...
}
```

The fix may consist in calling the `DateTimeFormatter.ofLocalizedDateTime()` method with two parameters, something like `DateTimeFormatter.ofLocalizedDateTime(getFormatStyle(dateStyle), getFormatStyle(timeStyle))`.

3. You can't use the converter with `type="localTime"` or `type="localDateTime"` and `dateStyle` or `timeStyle` with a value of `long` or `full`. This will cause an `javax.faces.convert.ConverterException` exception which "hides" the "original" exception caused by the JDK-8085887 bug. Although the bug only mentions `LocalDateTime`, the same happens when using `LocalTime`.