01

Phaselistener

Transcrição

Vamos dar uma olhada no pacote `br.com.alura.livraria.util` do projeto `livraria`. Dentro do pacote existem duas classes: o `Autorizador`, que é um `PhaseListener`, e faz todo o controle para realização do login do usuário. A outra classe é o `LogPhaseListener`, que imprime qual fase estamos passando, dentro das fases do JSF.

No `LogPhaseListener`, estamos observando todas as fases:

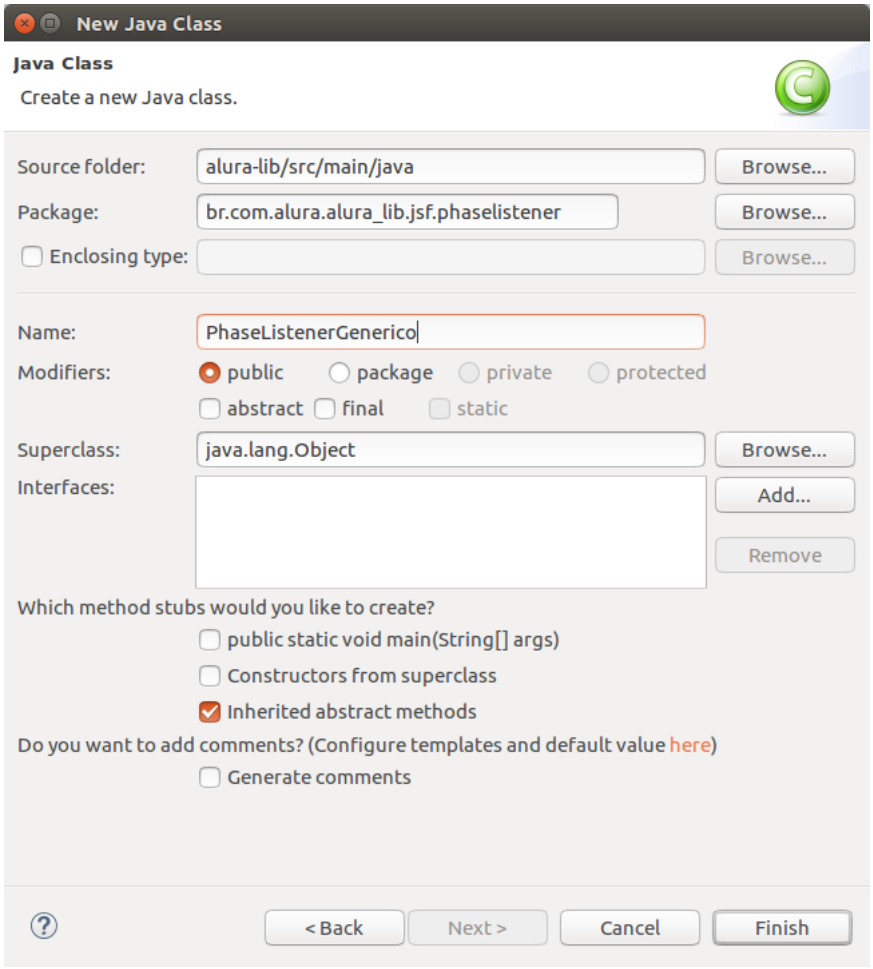
```
@Override
public PhaseId getPhaseId() {
    return PhaseId.ANY_PHASE;
}
```

E se precisássemos observar uma fase específica? Se após a execução de um método de um *bean*, por exemplo, quiséssemos realizar alguma ação? Existe uma fase chamada `INVOKE_APPLICATION` que poderíamos utilizar. Porém seria necessário criar outro `PhaseListener`.

Ou seja: criar uma classe, implementar a interface `PhaseListener`, inserir a implementação desejada. Depois seria necessário alterar o arquivo `faces-config.xml`, dentro do diretório `WEB-INF`, e declarar o *phase listener* nesse arquivo.

Ter um `PhaseListener` pode ser comum em determinado momento de uma aplicação. E não seriam necessários vários. Basta que seja observado todas as fases, e após chegar em uma determinada, todos que desejam ouvir aquela fase sejam notificados. O objetivo é levar um `PhaseListener` genérico para o projeto `alura-lib`.

Na biblioteca, vamos criar a classe `PhaseListenerGenerico` no pacote `br.com.alura.alura_lib.jsf.phaselistener`:



A classe implementa `PhaseListener` e no método `getPhaseId()`, estão sendo ouvidas todas as fases. Em seguida cada uma das "pessoas" que estiverem interessadas em saber em que fase estamos passando (antes ou depois de determinada fase) serão notificadas.

Vamos resolver isso, mas antes, para finalizar as configurações, no `beforePhase()` vamos por o mesmo código que temos no `LogPhaseListener` do projeto `livraria`:

```
public class PhaseListenerGenerico implements PhaseListener {

    private static final long serialVersionUID = 4332180564534271099L;

    @Override
    public void afterPhase(PhaseEvent event) {

    }

    @Override
    public void beforePhase(PhaseEvent event) {
        System.out.println("FASE: " + event.getPhaseId());
    }

    @Override
    public PhaseId getPhaseId() {
        return PhaseId.ANY_PHASE;
    }
}
```

O próximo passo é apagar o `LogPhaseListener` do projeto `livraria`, pois não precisamos mais dele. Ele será substituído pelo `PhaseListenerGenerico`. Para isso, vamos instalar a biblioteca no repositório local, e em seguida realizar o "Maven > Update Project..." no projeto `livraria`, ou utilizar o atalho "Ctrl + F5".

No arquivo `faces-config.xml`, é necessário alterar o nome da classe:

```
<lifecycle>
    <phase-listener>br.com.alura.livraria.util.Autorizador</phase-listener>
    <phase-listener>br.com.alura.alura_lib.jsf.phaselistener.PhaseListenerGenerico</phase-listener>
</lifecycle>
```

Vamos dar um "Clean" no Tomcat e iniciá-lo, para ver se tudo continua funcionado. E a partir disso será feita a implementação que irá notificar os interessados em saber se estamos antes ou depois de determinada fase. Ao acessar uma página da aplicação é possível ver no Console que tudo continua funcionando:

```
FASE: RESTORE_VIEW 1
/livro.xhtml
FASE: APPLY_REQUEST_VALUES 2
FASE: PROCESS_VALIDATIONS 3
FASE: UPDATE_MODEL_VALUES 4
FASE: INVOKE_APPLICATION 5
FASE: RENDER_RESPONSE 6
```

Na próxima aula será implementada a parte de notificar os interessados.