10

Mãos a obra: Adicionando funcionalidades à biblioteca

Vamos criar uma anotação para qualificar os mapas de escopos que temos no `JSF` :

```
@Qualifier
@Target({ElementType.METHOD, ElementType.PARAMETER, ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
public @interface ScopeMap{
    ScopeMap.Scope value();

    enum Scope {
        REQUEST,SESSION,APPLICATION;
    }
}
```

Crie também uma classe para produzir o objeto `FacesContext`

```
public JSFFactory {

    @Produces
    @RequestScoped
    public FacesContext getFacesContext(){
        return FacesContext.getCurrentInstance();
    }

    @Produces
    @RequestScoped
    public Flash getFlash(){
        return getExternalContext().getFlash();
    }

    @Produces
    @ScopeMap(Scope.REQUEST)
    public Map<String,Object> getRequestMap(){
        return getExternalContext().getRequestMap();
    }

    @Produces
    @ScopeMap(Scope.SESSION)
    public Map<String, Object> getSessionMap(){
        return getExternalContext().getSessionMap();
    }

    @Produces
    @ScopeMap(Scope.APPLICATION)
    public Map<String,Object> getApplicationMap(){
        return getExternalContext().getApplicationMap();
    }

    private ExternalContext getExternalContext(){
        return getFacesContext().getExternalContext();
    }
```

```
}
```

E uma classe para facilitar a criação de mensagens para o `JSF`

```
public MessageHelper {

    private FacesContext context;
    private Flash flash;

    @Inject
    public MessageHelper(FacesContext context, Flash flash){
        this.context = context;
        this.flash = flash
    }

    public MessageHelper onFlash(){
        flash.setKeepMessage(true);
        return this;
    }

    public void addMessage(FacesMessage message){
        addMessage(null, message);
    }

    public void addMessage(String clientId, FacesMessage message){
        context.addMessage(clientId, message);
    }

}
```

Feito isso instale no `jar` no repositório local: Clique com o botão direito sobre o projeto `Alura-lib` e selecione `Run as > Maven Install`.

No projeto `Livraria` vamos utilizar essas novas funcionalidades.

Primeiramente atualize o projeto: Clique com o botão direito do mouse sobre o projeto `Livraria` e selecione `Maven > Update project...`.

Feito isso vamos receber nossas dependências da classe `LoginBean` no nosso construtor.

```
    @Inject
    public LoginBean( UsuarioDao usuarioDao, @ScopeMap(Scope.SESSION) Map<String, Object> sessionMap
        this.usuarioDao = usuarioDao;
        this.sessionMap = sessionMap;
        this.helper = helper;
    }
```

Altere a classe `LoginBean` para usar os atributos que foam injetados.