

## Mãos na massa: Preços do produto

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

---

1) O tipo do preço de cada livro está limitado a 3 opções (**ebook**, **impresso** e **combo**), portanto, crie a Enum `TipoPreco`, no pacote `br.com.casadocodigo.loja.models`, com os valores `EBOOK`, `IMPRESSO` e `COMBO`:

```
public enum TipoPreco {  
  
    EBOOK, IMPRESSO, COMBO;  
  
}
```

2) Crie a classe `Preco` no pacote `br.com.casadocodigo.loja.models` com o atributo `valor`, do tipo `BigDecimal`, e `tipo`, do tipo `TipoPreco`. Gere também seus *getters* e *setters* desses atributos:

```
public class Preco {  
  
    private BigDecimal valor;  
    private TipoPreco tipo;  
  
    public BigDecimal getValor() {  
        return valor;  
    }  
  
    public void setValor(BigDecimal valor) {  
        this.valor = valor;  
    }  
  
    public TipoPreco getTipo() {  
        return tipo;  
    }  
  
    public void setTipo(TipoPreco tipo) {  
        this.tipo = tipo;  
    }  
  
}
```

3) Agora, adicione esses preços na classe `Produto`. Como cada livro pode ter mais de um preço, utilize uma **lista**. Gere também o *getter* e *setter* desse atributo:

```
@Entity  
public class Produto {  
  
    private List<Preco> precos;  
  
    public List<Preco> getPrecos() {  
        return precos;  
    }  
  
}
```

```

    public void setPrecos(List<Preco> precos) {
        this.precos = precos;
    }

    // restante do código omitido

}

```

4) Anote a classe `Preco` com `@Embeddable`, que a permite ser persistida, desde que ela seja um atributo de uma entidade, e no caso ela é um atributo da classe `Produto`, que é uma entidade:

```

@Embeddable
public class Preco {

    // restante do código omitido

}

```

5) Para indicar que você irá armazenar uma lista de `Preco`, de uma classe `@Embeddable`, utilize a anotação `@ElementCollection`. Portanto, anote o atributo `precos` da classe `Produto`:

```

@ElementCollection
private List<Preco> precos;

```

6) Para adicionar os campos de preço na *view*, passe os tipos de preço para a JSP. Para isso, utilize o `ModelAndView`, que, além de carregar a página, permite enviar objetos para a *view*. No *controller* `ProdutosController`, altere o método `form()` para retornar um `ModelAndView` no lugar de uma string. Em seguida, crie um objeto dessa classe que retorna a página `produtos/form`.

Além de retornar a página, envie o objeto que representa os tipos de preço, através do método `addObject`, da classe `ModelAndView`, enviando uma string que indica o nome do objeto como primeiro parâmetro e os valores do objeto como segundo parâmetro:

```

@RequestMapping("/produtos/form")
public ModelAndView form() {

    ModelAndView modelAndView = new ModelAndView("produtos/form");
    modelAndView.addObject("tipos", TipoPreco.values());

    return modelAndView;
}

```

7) Percorra a lista de tipos para exibir os dados no formulário, usando o `forEach` da `JSTL`. Então, em `form.jsp`, antes do botão de cadastro, adicione:

```

<c:forEach items="${tipos}" var="tipoPreco" varStatus="status">
    <div>
        <label>${tipoPreco}</label>
        <input type="text" name="precos[${status.index}].valor">
        <input type="hidden" name="precos[${status.index}].tipo" value="${tipoPreco}">
    </div>
</c:forEach>

```

```
    </div>  
</c:forEach>
```

8) E não esqueça de importar a JSTL através da `taglib` , antes da *tag* `<html>` :

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```