01

Introdução ao desenvolvimento web com JSF

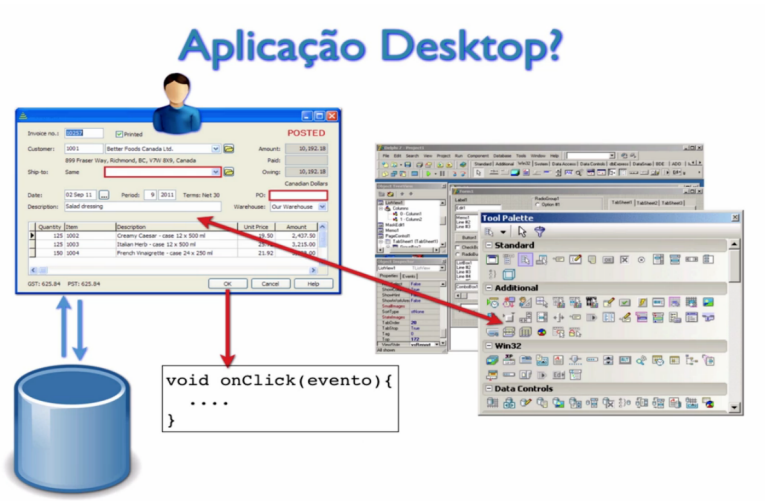
Transcrição

Os usuários habituaram-se com aplicações Desktop. Este tipo de aplicação é instalada no computador local e acessa diretamente um banco de dados ou gerenciador de arquivos.

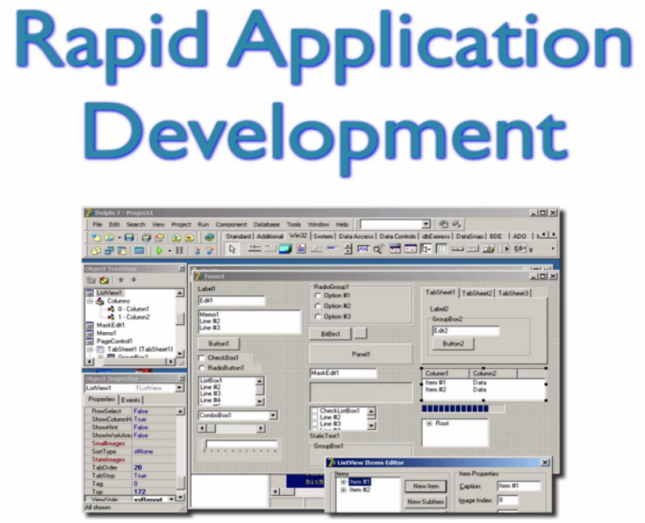
Para o desenvolvedor, a aplicação Desktop é construída com uma série de componentes que a plataforma de desenvolvimento oferece para cada sistema operacional. Esses componentes ricos e muitas vezes sofisticados estão associados a eventos ou procedimentos que executam lógicas de negócio.

Problemas de validação dos dados são indicados na própria tela sem que qualquer informação do formulário seja perdida. De uma forma natural, esses componentes lembram-se dos dados do usuário, inclusive entre telas e ações diferentes.

Nesse tipo de desenvolvimento são utilizados diversos componentes ricos, como por exemplo, calendários, menus diversos ou componentes *drag & drop* (arrastar e soltar). Eles ficam associados à eventos/ações e guardam automaticamente seu estado, pois mantém os valores digitados pelo usuário.



Essa é uma boa prática no desenvolvimento orientado a objetos e ganhou o nome de Desenvolvimento Rápido de Aplicações, ou simplesmente RAD, onde o ambiente de desenvolvimento e os componentes reutilizáveis favorecem a criação rápida de aplicações.



O cliente gordo e o desenvolvimento Desktop

Existem algumas desvantagens nessa abordagem, visto que cada usuário possui uma cópia, muitas das vezes integral, dessa aplicação. Qualquer alteração precisaria ser propagada para todas as outras máquinas. Estamos usando um "cliente gordo", com muita responsabilidade no lado do cliente.

Para piorar, as regras de negócio rodam no computador do usuário, sendo muito difícil depurar a aplicação. Em geral, enfrentamos problemas de manutenção e gerenciabilidade.

O desenvolvimento Web e o protocolo HTTP

Para resolver problemas como esse, surgiram as aplicações baseadas na web. Nessa abordagem há um servidor central, onde a aplicação é executada e processada, e todos os clientes podem acessá-la através do protocolo HTTP.

O usuário precisa basicamente instalar um navegador web, como Firefox ou Internet Explorer, que receberá o HTML, o CSS e o JavaScript, que são afinal tecnologias que ele entende.

Entre cada requisição (*request*), trafega o HTML do lado servidor (*Server Side*) para o computador do cliente (*Client Side*). Em nenhum momento a aplicação está salva no cliente. Todas as regras da aplicação estão sendo processadas no lado do servidor. Por isso, essa abordagem também foi chamada de "cliente magro".

Isso facilita bastante a manutenção e a gerenciabilidade, pois temos um lugar - central -, onde a aplicação é executada, mas ainda é preciso conhecer bastante de HTML, CSS e JavaScript para definir a interface com o usuário. Também não há mais eventos, mas sim um modelo bem diferente orientado a requisição e resposta. Além disso, ainda é preciso conhecer o protocolo HTTP. Assim, toda essa responsabilidade fica a cargo do desenvolvedor.

Comparando as duas abordagens, podemos ver vantagens e desvantagens em ambas. No lado da aplicação puramente Desktop, temos um estilo de desenvolvimento orientado ao evento, usando componentes ricos, porém com problemas de manutenção e gerenciamento. Do outro lado, as aplicações web são mais fáceis de gerenciar e manter, mas precisamos lidar com HTML e conhecer o protocolo HTTP, e seguir o modelo *request-response* (requisição-resposta).

Mesclar desenvolvimento Desktop e Web

O ideal seria mesclar os dois estilos, aproveitando as vantagens de cada um. Seria um desenvolvimento Desktop para WEB central e com componentes ricos. Isso é justamente a ideia dos *frameworks* web baseados em componentes.

Frameworks Web baseados em componentes

No mundo Java há algumas opções como JavaServer Faces (JSF), Apache Wicket, Vaadin, Tapestry ou GWT, do Google. Todos eles são *frameworks* web baseados em componentes.

Analisando um pouco melhor o JSF, percebemos que ele é na verdade um padrão ou especificação que faz parte do ***Java Enterprise Edition*** (Java EE). Por ser uma especificação, ou *Java Specification Request* (JSR), ele é mantido dentro do *Java Community Process* (JCP).

Vamos procurar então o JSF no site da JCP (http://www.jcp.org). Digitando 314 na busca das JSRs (buscar das especificações), é mostrado como resultado a última versão do JSF. Navegando no resultado, podemos ver todos os documentos disponíveis para os interessados na implementação dessa especificação JSF.

Baseado nesta especificação, há várias implementações. A mais famosa, e também implementação referencial (RI), é a Oracle Mojarra (http://javaserverfaces.java.net/), aquela que mostra como o JSF deveria se comportar. Outra implementação famosa é da *Apache Software Foundation*, e se chama MyFaces (http://myfaces.apache.org/).

Neste caso baixaremos a implementação Mojarra no link indicado para utilizá-lo depois em nosso projeto. Acessando o site javaserverfaces.java.net (http://javaserverfaces.java.net/), na parte de *downloads*, podemos encontrar o *link (https://maven.java.net/content/repositories/releases/org/glassfish/javax.faces/2.1.16/javax.faces-2.1.16.jar)* para baixar o JAR do Mojarra.

Introdução ao JSF com Mojarra e PrimeFaces

Como dito antes, nosso projeto utilizará a implementação Mojarra do JSF. Ela já define o modelo de desenvolvimento e oferece alguns componentes bem básicos. Nada além de `inputs`, botões e `ComboBox` simples. Não há componentes sofisticados dentro da especificação. Isto é proposital, pois o mundo web evolui rápido (principalmente na questão das interfaces gráficas).

Para atender a demanda dos desenvolvedores por componentes mais sofisticados, existem várias extensões do JSF que seguem o mesmo ciclo e modelo da especificação. São exemplos dessas bibliotecas: **PrimeFaces**, **RichFaces** ou **IceFaces**.

Todos eles oferecem *ShowCases* na web para mostrar seus componentes e suas funcionalidades. Daremos uma olhada no **PrimeFaces**, acessando o site primefaces.org (http://www.primefaces.org).

Podemos ver no ***demo online*** uma lista de componentes disponíveis. Vamos dar uma olhada em alguns dos componentes, começando com o calendário (http://www.primefaces.org/showcase/ui/input/calendar.xhtml), que claro, não poderia faltar. Outro componente simples é o *carousel* (http://www.primefaces.org/showcase/ui/data/carousel.xhtml), que apresenta os dados em painéis fáceis de navegar. Já mais sofisticada é a agenda (https://www.primefaces.org/showcase/ui/data/schedule.xhtml) onde podemos cadastrar eventos. Por último não pode faltar os componentes *Drag and Drop* (http://www.primefaces.org/showcase/ui/dnd/dataGrid.xhtml).

Para este treinamento usaremos **Mojarra** com o **PrimeFaces**.

O primeiro projeto e os componentes poderosos do Primefaces

Para mostrar a simplicidade dos componentes ricos, preparamos para você um projeto base no Eclipse. Nos próximos vídeos veremos em detalhes a configuração desse projeto. Queremos com ele mostrar a simplicidade do desenvolvimento de interfaces ricas para web (***Rich Internet Application***) com componentes JSF.

As bibliotecas necessárias (JARs) já estão disponíveis na pasta *lib* e também temos uma página inicial criada, a **olamundo.xhtml**, como primeiro exemplo. Ela já está pronta para os componentes do *PrimeFaces* e tem um corpo que mostra um simples cabeçalho HTML.

Podemos iniciar o servidor e chamar a página no navegador `http://localhost:8080/jsf/olamundo.xhtml` (http://localhost:8080/jsf/olamundo.xhtml). Aparecerá a nossa página de exemplo.

De volta ao Eclipse, vamos experimentar alguns componentes do *PrimeFaces* para ilustrar o uso. O primeiro será o calendário. Na página **olamundo.xhtml**, que está dentro da pasta **WebContent** do projeto **jsf** vamos digitar `<calendar />` logo após o `<br />` para referenciar o componente.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:f="http://java.sun.com/jsf/core"
     xmlns:h="http://java.sun.com/jsf/html"
```

```
        xmlns:p="http://primefaces.org/ui">

    <h:head>
        <title>Primeiro Exemplo usando JSF - olamundo.xhtml</title>
    </h:head>

    <h:body>
        <h2>Exemplo Ola Mundo com JSF 2.0 e PrimeFaces</h2>
        <br />
        <p:calendar />
    </h:body>
    </html>
```

Agora é só salvar e está pronto para testar no navegador. Temos então um *input* que renderiza automaticamente um calendário.

Como próximo componente vamos experimentar um `Panel`. Novamente, super simples de usar. Basta colocar `p:panel` e definir um cabeçalho e um corpo. Ao testar aparece o *panel*, mas ele está fixo na página, e não pode ser arrastado. Vamos voltar ao Eclipse e ativar essa funcionalidade com o componentes `p:draggable`. Aqui é preciso criar uma ligação entre *panel* e *draggable* pela ID do componente.

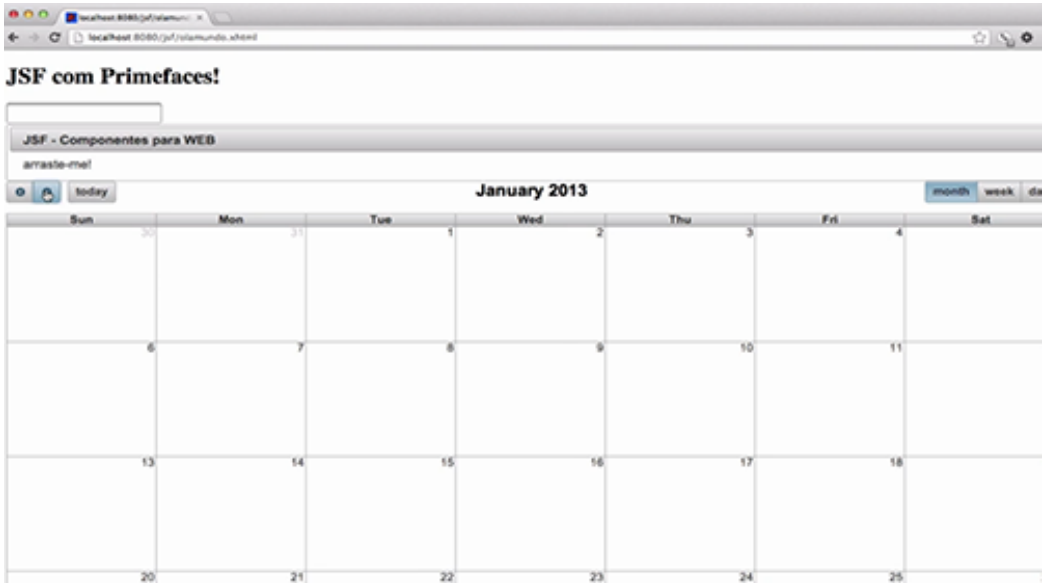
```
    <!-- código omitido -->

        <p:panel id="panel"  header="JSF - Componentes para Web">
            arraste-me!
        </p:panel>
        <p:draggable for="panel" />

    <!-- código omitido -->
```

Novamente vamos testar no navegador e agora sim podemos arrastar o *panel* livremente. Repare que em nenhum momento nos preocupamos com JavaScript, CSS ou HTML. Usamos apenas componentes para definir uma interface bastante agradável.

Por último, como vimos a agenda no *showcase* do PrimeFaces. Vamos testar esse componente também. Ele se chama `p:schedule` e também é de simples utilização.

```
        <p:schedule />
```

Ao atualizar no navegador, aparece a agenda sofisticada.


JSF com Primefaces!
JSF - Componentes para WEB
arraste-me!
today
January 2013
month
week
Sun
Mon
Tue
Wed
Thu
Fri
Sat