ANÁLISE DE ADERÊNCIA DE BIBLIOTECAS JAVA SERVER FACES EM RELAÇÃO A HTML5

Luiz AngeloHeinzen[1]
[1]Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brazil
laheinzen@gmail.com

Resumo
O HTML5 apresentou novos recursos à linguagem HTML com o objetivo de enriquecer a experiência dos usuários. O presente artigo tem por objetivo analisar o HTML gerado pelas bibliotecas de componentes JSF PrimeFaces e RichFaces no tocante a estes novos recursos. Conclui-se que essas bibliotecas não geram HTML5 ativamente, usando componentes não HTML5 próprios da biblioteca sempre que possível, mas permitindo a alteração desse comportamento através de funcionalidades do JSF. A biblioteca PrimeFaces se mostrou mais amigável a geração de HTML5 do que a biblioteca RichFaces.

Palavras-chave: HTML 5. Java Server Faces. Richfaces. Primefaces.

ANALYSIS OF JAVA SERVER FACES LIBRARIES ADHERENCE IN RELATION TO HTML5

Abstract
HTML5 has brought new features to the HTML aiming to enrich the user's experience. The main goal of this article is a comparison between two JSF component libraries: PrimeFaces and RichFaces JSF, exploring features related to HTML generation. The tests showed that these libraries don't generate HTML5 in an active way. Instead, they give preference to their own non-HTML5 components whenever possible, allowing to use JSF features as a way to change this behavior. The PrimeFaces library was more HTML5 generation friendly than RichFaces.

Keywords: *HTML5. Java Server Faces. RichFaces. PrimeFaces.*

1. Introdução

A popularidade do Java pode ser atestada ao acessar o índice de popularidade de linguagens de programação elaborada pela TIOBE Software BV (TIOBE, 2014), que mostra Java como a 2ª mais popular linguagem de desenvolvimento. Segundo Franco (2011, p. 13) "o ambiente mais utilizado atualmente para o desenvolvimento de aplicações corporativas na plataforma Java é o ambiente web.".

De acordo com Oracle (2013), proprietária do Java, a tecnologia Java Server Faces (JSF) é um componente do lado do servidor para construção de aplicações web com tecnologia Java. Segundo Carmisini (2012) "o principal objetivo do JSF é tornar mais ágil o desenvolvimento de aplicações web, aumentado a produtividade e minimizando a complexidade da manutenção".

Uma das razões para o aparecimento das bibliotecas de componentes JSF é que o JSF disponibiliza apenas um conjunto básico de componentes para a criação de páginas, enquanto aplicações web modernas precisam de componentes mais avançados (PITOŇÁK, 2011).

O HTML5 é a mais recente versão do HTML e traz novos elementos, atributos e funcionalidades, para proporcionar uma maior riqueza a experiência do usuário. Entre essas novidades há a definição de novos tipos de dados (email e URL, por exemplo) bem como

atributos novos para os elementos (sinalizar que um campo é obrigatório, ou que recebe automaticamente o foco, por exemplo). Tais novidades do HTML5 podem ser usadas em vez dos componentes diferenciados disponibilizados pelas bibliotecas de componentes.

A especificação do JSF 2.2, lançada em maio de 2013, tinha como um dos objetivos principais ser mais amigável ao HTML5. O objetivo desde trabalho é analisar a aderência de bibliotecas JSF ao HTML5 no tocante aos novos tipos de entrada, novos elementos e novos atributos. Portanto foram analisadas as bibliotecas de componentes compatíveis com o JSF 2.2: PrimeFaces 4.0 e RichFaces 4.3.5.

A metodologia aplicada foi criar páginas JSF com as bibliotecas, buscando a geração de código HTML que fizesse uso das funcionalidades do HTML5. Foi utilizado o Netbeans, na versão 7.4, criando um projeto único, Maven, Java Enterprise Edition 7, servidor Glassfish. O projeto foi configurado para fazer uso do JSF 2.2, junto com as bibliotecas e suas dependências. O mesmo código fonte Java foi usado tanto nas classes modelo, quanto nos *managed beans*. Foram criadas então páginas JSF, ora com o PrimeFaces, ora com o RichFaces.

O resultado encontrado foi que alguns dos problemas que o HTML5 busca resolver, em especial os novos atributos e elementos, podem ser resolvidos com componentes básicos do JSF e/ou das bibliotecas. Quando da geração do HTML há uma preferência das bibliotecas em solucionar os problemas usando suas próprias respostas (componentes) ao invés de usar HTML5. Entretanto o desenvolvedor consegue reverter tal preferência: mais facilmente no PrimeFaces do que no RichFaces – que acaba se apoiando basicamente das funcionalidades do JSF 2.2.

No decorrer deste trabalho a seção 2 apresenta a Fundamentação Teórica. Já a seção 3 apresenta trabalhos correlatos, onde são apresentados outros trabalhos relacionados ao presente. A seção 4 apresenta a avaliação das bibliotecas, detalhando os resultados encontrados. Por fim a são apresentadas as conclusões encontradas pelo autor.

2. Fundamentação Teórica

Esta seção apresenta conceitos pertinentes a este trabalho. São abordados HTML5 e sua novidades, bem como as bibliotecas de componentes do JSF 2.2 PrimeFaces e RichFaces.

2.1 HTML5

De acordo com W3 (2013) em 1989 Tim-Berners Lee criou a World Wide Web, escreveu o primeiro servidor e o primeiro programa cliente em 1990, e escreveu a primeira versão da *Hyper Text Markup Language* (HTML). De acordo com W3 Schools (2013) a versão 4.01 do HTML foi liberada em 1999, e a internet mudou significativamente desde então. A versão 5 do HTML foi pensada para substituir o HTML, o XHTML e o HTML DOM Level 2. O HTML5 é fruto de um trabalho colaborativo entre o W3C e a Web Hypertext Application Technology Working Group (WHATWG).

De acordo com Mozilla (2013) o HTML5 "é uma nova versão da linguagem HTML, com novos elementos, atributos, e comportamentos e um conjunto maior de tecnologias que permite o desenvolvimento de aplicações e web sites mais diversos e poderosos. Este conjunto é chamado HTML5 e amigos e muitas vezes abreviado apenas como HTML5".

A especificação da HTML5 foi finalizada, mas ainda não é um padrão. Ou seja, ainda sujeita a alterações. Apesar disso os navegadores já suportam parte de sua especificação.

A seguir serão apresentadas as novidades do HTML5 consideradas relevantes para o presente trabalho. Será apresentado primeiro a categoria da novidade, seguida de uma tabela, descrevendo melhor cada uma das novidades, junto com o suporte dos navegadores. O conceito e dados sobre o suporte são baseados nos encontrados em W3 Schools (2013). Essa fonte lista o suporte das seguintes versões dos navegadores *desktop* Internet Explorer 11 (IE), Firefox 27 (FF), Chrome 33 (CH), Safari 7 (SF) e Opera 19 (OP).

Com o intuito de melhor atender as necessidades da Internet, a especificação do HTML5 disponibilizou novos tipos de entradas. Esse novos tipos de entrada trazem um melhor controle e validação dos dados fornecidos pelos usuários. Quando o navegador não reconhecer o tipo de entrada, o mesmo irá se comportar como campo do tipo *text*. No quadro 1 vemos os novos tipos, junto com uma descrição sucinta e o suporte ao novo tipo.

Tipo	*Descrição*	*Suporte Navegadores*				
		IE	FF	CH	SA	OP
color	Seletor de cor			*		*
date	Seletor de data			*	*	*
datetime	Data e hora, considerando fuso horário				*	*
datetime-local	Data e hora, sem considerar fuso horário				*	*
email	Campo que aceita apenas e-mails em formato válido	*	*	*		*
month	Seletor de mês e ano			*	*	*
number	Campo que aceita apenas números	*		*	*	*
range	Apresenta ao usuário um seletor, onde o mesmo pode escolher um número dentro de uma faixa. Especifica atributos extras: *max* (valor máximo permitido), *min* (valor mínimo permitido), *step* (intervalo entre os valores) e *value* (valor padrão).	*	*	*	*	*
search	Campo de busca			*	*	
tel	Campo que deve conter números de telefone					
time	Campo para informar hora			*	*	*
url	Define que o campo deve conter uma entrada URL, com validação ao enviar o formulário, devendo conter o protocolo, (http:// por exemplo)	*	*	*		*
week	Campo para selecionar um ano e número de semana			*	*	*

Quadro 1. Suporte a novos tipos de entradas nos navegadores Internet Explorer 11 (IE), Firefox 27 (FF), Chrome 33 (CH), Safari 7 (SF) e Opera 19 (OP) (Adaptado de W3 Schools, 2012)

O HTML5 disponibilizou três novos elementos que podem ser usados em formulários. Nem todos os navegadores dão suporte a esses tipos. Caso o mesmo não suporte, deverá tratar o elemento como um campo de texto simples. O quadro 2 apresenta os novos elementos, com uma breve descrição e o suporte dos navegadores a esses elementos.

Elemento	*Descrição*	*Suporte Navegadores*				
		IE	FF	CH	SA	OP
datalist	Especifica uma lista de valores para um elemento do tipo *input*.	*	*	*		*
keygen	Usado para definir um campo que gere um par (pública/privada) de chaves em um formulário. As chaves são geradas quando o formulário é enviado, sendo que a chave privada é armazenada localmente e a chave pública é enviada ao servidor,		*	*	*	*
output	Usado para mostrar o resultado de um cálculo.		*	*	*	*

Quadro 2. Suporte a novos elementos de formulários nos navegadores Internet Explorer 11 (IE), Firefox 27 (FF), Chrome 33 (CH), Safari 7 (SF) e Opera 19 (OP) (Adaptado de W3 Schools, 2012)

Foram adicionados novos atributos para os formulários em HTML5, permitindo ao desenvolvedor ditar, de maneira mais avançada, o comportamento do formulário. O quadro 3 introduz os elementos, com uma breve explicação e o suporte a estes.

Atributo	*Descrição*	*Suporte Navegadores*				
		IE	FF	CH	SA	OP
autocomplete	Para definir se um formulário deve ser auto preenchido, com base em dados informados	*	*	*	*	

	anteriormente pelo usuário, ou não.					
novalidate	Usado para definir que um formulário não deve ser validado ao ser enviado	*	*	*		*

Quadro 3. Suporte a novos atributos para formulários nos navegadores Internet Explorer 11 (IE), Firefox 27 (FF), Chrome 33 (CH), Safari 7 (SF) e Opera 19 (OP) (Adaptado de W3 Schools, 2012)

Também foram criados atributos para os elementos de entrada. Os novos atributos ajudam a refinar a aparência e comportamento dos elementos. A tabela 4 elenca os atributos, descrevendo brevemente o atributo, juntamente com o suporte em navegadores.

O atributo *autocomplete* está presente tanto para formulários quanto para elementos. Os atributos *autocomplete* do formulário e dos campos podem ser combinados para ligar/desligar o auto preenchimento de campos individuais. Ou seja: você pode definir que delisgar o *autocomplete* em um formulário, mas ligar para campos individuais dentro do formulário, como também pode ligar no formulário e desligar para elementos individuais.

Atributo	*Descrição*	*Suporte Navegadores*				
		IE	FF	CH	SA	OP
autocomplete	Define se o campo deve ser ou não auto preenchido com base em dados anteriormente informados.	*	*	*	*	
autofocus	O campo com esse atributo recebe automaticamente o foco quando a página é carregada.	*	*	*	*	*
form	Indica o(s) formulário(s) a que um campo pertence.		*	*	*	*
formaction	URL de processamento do formulário ao usar o elemento a qual o atributo pertence. Válido para os elementos *submit* e *image*.	*	*	*	*	*
formenctype	Codificação a ser usada ao enviar ao usar o elemento a qual o atributo pertence. Usado com os tipos *submit* e *image*, valido apenas quando o método for o *POST*.	*	*	*	*	*
formmethod	Método HTTP a ser chamado ao enviar o formulário usando o elemento ao qual o atributo pertence. Usado com os tipos *submit* e *image*.	*	*	*	*	*
formnovalidate	A sua presença indica que o campo não deve ser validado quando o formulário ao qual pertence for enviado. Pode ser usado com campos do tipo *submit*.	*	*	*		*
formtarget	Especifica (através de nome ou palavra chave) onde mostrar a resposta recebida ao enviar o formulário – se em numa nova página, por exemplo. Usado com campos do tipo *submit* e *image*.	*	*	*	*	*
height/width	Apenas para elementos *image*, define altura e largura.	*	*	*	*	*
list	Faz referência a uma *datalist* que contém os opções pré-definidas para o campo.	*	*	*		*
min/max	Define valores mínimo e máximo (respectivamente) para um campo. Válido para: *number*, *range*, *date*, *datetime*, *datetime-local*, *month*, *time* e *week*.	*		*	*	*
multiple	Quando presente indica que o campo aceita múltiplos valores. Válido para campos do tipo *email* e *file*.	*	*	*	*	*
pattern	Define uma expressão regular ao qual a entrada do campo será validada. Válido para campos do tipo *text*, *search*, *url*, *tel*, *email* e *password*.	*	*	*		*
placeholder	Indica um texto, normalmente indicando a entrada esperada, que deve aparecer enquanto o usuário não entra com o valor do campo. Válido para campos do tipo *text*, *search*, *url*, *tel*, *email* e *password*.	*	*	*	*	*

required	Indica campo de preenchimento obrigatório. Válido para os campos do tipo *text*, *search*, *url*, *tel*, *email*, *password*, *date*, *datetime*, *datetime-local*,*month*, *time*, *week*, *number*, *checkbox*, *radio* e *file*.	*	*	*		*
step	Define o intervalo legal para um campo. Válido para campos do tipo *number*, *range*, *date*, *datetime*, *datetime-local*, *month*, *time* e *week*.	*		*	*	*

Quadro 4. Suporte a novos atributos para entradas em formulários nos navegadores Internet Explorer 11 (IE), Firefox 27 (FF), Chrome 33 (CH), Safari 7 (SF) e Opera 19 (OP) (Adaptado de W3 Schools, 2012)

Alguns dos novos atributos para elementos em formulários se destinam a sobrescrever o mesmo atributo presente no formulário. São os atributos *formaction, formtype*, *formmethod*, *formnovalidate* e *formtarget* que sobrescrevem, respectivamente, os atributos *action, type*, *method*, *novalidate* e *target* respectivamente. Com isso o desenvolvedor pode definir comportamentos diferentes para um mesmo formulário.

2.2. Bibliotecas de componentes JSF

De acordo com PrimeTek (2014) o PrimeFaces é uma *suíte* de componentes JSF de código aberto com várias extensões. . É mantido pela PrimeTek Informatics, e existe há mais de cinco anos. A última versão do PrimeFaces é a 4.0, compatível com o JSF 2.2.

De acordo com Katamreddy (2011) o PrimeFaces é bem melhor que muitas outras bibliotecas de componente JSF por causa das seguintes razões: uma rica gama componentes de interface gráfica; nenhuma configuração XML extra é requerida e não tem outras dependências; AJAX embutido baseado no padrão JSF 2.0 Ajax APIs (*Application Programming Interface*); e Skinning com mais de 25 temas; documentação excelente com exemplos de código.

Como curiosidade, o autor principal do PrimeFaces lista em sua página que o ICEFaces (outra biblioteca popular) é cliente do Primefaces . O motivo é que, conforme PrimeFaces (2012), a versão 3 do ICEFaces, através de seus componentes ACE, faz uso do mesmo código fontes dos componentes do PrimeFaces, inclusive atribuindo a autoria em sua licença.

De acordo com RichFaces (2014) o projeto RichFaces é um *framework* de componentes avançados para integrar facilmente capacidades AJAX (*Asynchronous JavaScript and XML*) em aplicativos comerciais que usam JSF.

É uma biblioteca de componentes do JSF de código aberto, mantida pela JBoss, mesma criadora do servidor de aplicações JBOSS Application Server; também de código aberto e com base no Java EE. Além disso, a JBOSS é uma subsidiária da RedHat, mantenedora da distribuição Linux que leva o nome da empresa. A versão 3.10 do RichFaces foi lançada em Outubro de 2007 (SMIRNOV, 2007). Ou seja: vem sendo desenvolvida há mais de 6 anos. A versão mais recente, a usada no trabalho é a 4.3.5., compatível com JSF 2.2.

O RichFaces estende as capacidades AJAX do JSF com características avançadas para o desenvolvimento de aplicações web empresariais. Faz uso de várias funcionalidades do JSF 2, como ciclo de vida, validação, conversões e o gerenciamento dos recursos dinâmicos e estáticos. Possui validação no lado do cliente (navegador); um Kit de Desenvolvimento de componentes, para desenvolver componentes próprios; e permite skinning dos componentes. (LEATHEM; FRYC; ROGERS, 2014)

3. Trabalhos correlatos

Para a elaboração deste artigo foi feita uma pesquisa em busca de trabalhos que tivessem objetivos próximos ao do atual: comparar a aderência das bibliotecas JSF no que toca a geração de páginas HTML5. O objetivo era garantir que o comparativo realizado neste trabalho seja um estudo inédito. A pesquisa não encontrou trabalhos com o mesmo objetivo. A maioria dos

trabalhos encontrados faziam apenas comparativos entre as diferentes bibliotecas JSF, sem levar em conta o HTML5. Diante da carência de fontes sobre o assunto, ficou evidente a originalidade deste trabalho. A seguir serão apresentados os trabalhos correlatos encontrados cujo objetivo é o mais próximo deste.

Carmisini (2012) faz um comparativo de bibliotecas de componentes JSF. O trabalho analisou as bibliotecas Tobago 1.5.7, PrimeFaces 3.4 e RichFaces 4.2.0. Dentre os critérios não estava a aderência ao HTML5. Após analisar a disponibilidade de componentes, documentação, suporte, configuração de uso e frequência nas atualizações o autor define que a melhor biblioteca é o PrimeFaces, seguido do RichFaces e, por último, Tobago.

Em Pitoňák (2011) o autor do trabalho era empregado da RedHat e trabalha na equipe desenvolvedora do RichFaces. O objetivo principal era comparar as bibliotecas ICEFaces 2, OpenFaces 3.0 e, PrimeFaces 2.2.1 com RichFaces 4.1 e descobrir como elas interagem. O autor comparou as bibliotecas nos seguintes critérios: componentes básicos e AJAX, componentes de entrada, componentes de saída, painéis, componentes de interação, gráficos e validação. Ao final concluiu que, em termos de funcionalidade, as bibliotecas são muito similares, mas que nem sempre um mesmo tipo de componentes é implementado em todas as bibliotecas. O autor não considerou a aderência ao HTML5 em suas análises.

Em Estrada (2013) a autora busca a melhor biblioteca JSF para ser usada no desenvolvimento de um sistema nutricional para a Escola Superior Politécnica de Chimborazo, no Equador e é a tese de graduação da autora na instituição. Os aspectos usados na análise foram: facilidade de aprendizado (suporte e documentação), qualidade, componentes (disponibilidade e aplicabilidade ao problema), facilidade de desenvolvimento (instalação, configuração, uso, suporte AJAX, linhas de código). Não houve preocupação com a aderência ao HTML5 no desenvolver do trabalho. Ao final do trabalho a avaliação da autora determinou que a melhor biblioteca era o IceFaces, seguido do MyFaces e, por último, RichFaces. O trabalho também contemplava o desenvolvimento do sistema, que foi desenvolvido então usando o IceFaces.

4. Avaliação das bibliotecas

O suporte da tecnologia JSF ao HTML5 se dá de duas maneiras: passagem de elementos (*pass-through elements*) e passagem de atributos (*pass-through attributes*). A passagem de elementos permite que o desenvolvedor use elementos do HTML5 com atributos do JSF. A passagem de atributos é usada com componentes do JSF e permite o uso de atributos do HTML5 em componentes JSF, que são então renderizados, sem qualquer tipo de processador por parte do JSF, no HTML final. O resultado final alcançado permite que desenvolvedor misture componentes do JSF com elementos HTML5 livremente. (Oracle, 2013)

Para avaliar a aderência das bibliotecas JSF ao HTML foram criadas páginas JSF com os componentes das mesmas e foi avaliado o HTML gerado. Caso o JSF não fizesse uso do HTML desejado, o código JSF foi alterado até que se alcançasse o resultado esperado.

O que se verificou foi que, na maior parte dos casos, as bibliotecas não fazem uso das funcionalidades do HTML, preferindo usar componentes e atributos próprios. Como exemplo temos os campos cujo valor esperado é uma cor: o ideal seria o uso do *color* do HTML5, mas o PrimeFaces prefere usar um componente próprio.

No quadro 5 encontram-se os *imports* que foram usados nos testes, bem como nos trechos de códigos que fazem parte do presente trabalho.

Conteúdo	*Namespace*
Tags HTML	xmlns:h="http://xmlns.jcp.org/jsf/html"
JSF	xmlns:f="http://xmlns.jcp.org/jsf/core"
Passagem de elementos	xmlns:pe="http://xmlns.jcp.org/jsf"
Passagem de atributos	xmlns:pa="http://xmlns.jcp.org/jsf/passthrough"

PrimeFaces	xmlns:p="http://primefaces.org/ui"
RichFaces (AJAX)	xmlns:a="http://richfaces.org/a4j"
RichFaces (componentes)	xmlns:r="http://richfaces.org/rich"

Quadro 5. *Imports* usados nas páginas JSF

4.1 Novos tipos de entrada do HTML5

No *Primefaces*, é facilmente obtido o equivalente HTML5 ao se usar o componente *inputText*. Um dos atributos do *inputText* é o tipo do campo (*type*) que aceita um *string* com valor. Isso permite que se escolha qual o tipo do elemento a ser usado no renderizador HTML. Por exemplo, ao configurar a página JSF com o código apresentado no quadro 6, obtem-se o código HTML apresentado no quadro 7.

```
<p:inputText id="qq" type="qualquerCoisa" value="#{umBean.umObj.qualquer}"/>
```

Quadro 6: Codigo PrimeFaces usando o atributo *type* do *inputText*

```
<input id="j_idt7:qq" name="j_idt7:qualquerCoisa" type="qualquerCoisa"
class="ui-inputfield ui-inputtext ui-widget ui-state-default ui-corner-all"
role="textbox" aria-disabled="false" aria-readonly="false" aria-
multiline="false">
```

Quadro 7: HTML gerado pelo PrimeFaces ao usar o atributo *type* do *inputText*

Como podemos observar o PrimeFaces usa o valor informado no atributo *type* como o valor do mesmo atributo no HTML. Com isso é possível fazer uso dos novos tipos de entrada do HTML – aqueles apresentados no quadro 1. A integração com o *managed bean* funciona perfeitamente, gravando e recuperando valores. Em nossos testes a maioria dos novos tipos tem seu valor retornado como *string*. Mas é possível usar também outros tipos nativos, como *int* e *float*.

Já o RichFaces não apresenta a mesma facilidade. A escolha intuitiva, em nossa opinião, seria o componente *inplaceInput*. Mas ao usar tal componente, o RichFaces o renderiza como sendo um elemento HTML do tipo *span*, e não do *input*. Se usarmos a passagem de atributo especificando *type="email"*, por exemplo, para um *inplaceInput,* o resultado final é que não há sequer como entrar com um valor no campo.

Verificamos os demais componentes de entrada do RichFaces em busca de algum que gerasse um campo do tipo *input* como saída HTML para que pudéssemos fazer apenas a passagem do atributo *type* para gerar o mesmo comportamento do PrimeFaces. Em nossos testes, somente o *inputNumberSpinner* gerou um *input.* Mas o problema encontrado foi que o RichFaces faz validação automática. Ou seja: ele até gera um campo do tipo *color*, por exemplo, mas não deixa seguir adianta pois o mesmo espera um valor inteiro no campo.

A única maneira de se fazer uso dos novos elementos HTML com o RichFaces é através da biblioteca HTML do JSF com passagem de atributos do JSF 2.2. O quadro 8 apresenta o código JSF usado para gerar um campo de email no PrimeFaces.

```
<p:inputText id="email" type="email" value="#{umBean.usuario.email}"
placeholder="Digite o email"/>
```

Quadro 8: Código PrimeFaces para gerar uma campo de entrada de email

O quadro 9 mostra como obter o mesmo resultado, um campo de email, no RichFaces.

```
<h:inputText id="email" pa:type="email" value="#{umBean.usuario.email}"
pa:placeholder="Digite o email" />
```

Quadro 9: Código RichFaces para gerar uma campo de entrada de email

Com relação aos novos tipos, as bibliotecas tem funcionalidades equivalente as novas do HTML5. O *color*, por exemplo tem o *colorPicker* no PrimeFaces. O comportamento de ambos difere, trazendo inclusive valores diferentes: o *colorPicker* traz uma *string* sem o # na frente do valor, já o componente nativo traz o # junto do valor escolhido.

4.2. Teste dos novos elementos do formulário

Não foi encontrada forma de usar os novos elementos sem recorrer à passagem de atributos. Como as bibliotecas não implementam um componente para formulário, fazendo uso do renderizador HTML do JSF, o comportamento em ambas é igual.

O *datalist* pode ser usado criando o mesmo no JSF e fazendo a passagem de atributos. Para isso basta criar o elemento no JSF e, no elemento que deve receber os valores, usar *pa:list* para definir os valores. Os valores do *datalist* podem ser estáticos ou dinâmicos (através de uma *ui:repeat*). No quadro 5 temos um exemplo de como usar o elemento dinamicamente, onde a linha 6 apresenta a vinculação da *datalist*. No quadro 6 vemos o código fonte do bean usado para gerar os valores.

```
1.  <r:panel header="Novos tipos usando o RichFaces">
2.     <h:form>
3.         <datalist id="emails">
4.             <ui:repeat value="#{umBean.emailsValidos}" var="email">
5.                 <option value="#{email}"/>
6.             </ui:repeat>
7.         </datalist>
8.         <h:panelGrid columns="2" style="width: 900px">
9.             <h:outputText value="Email"/>
10.            <h:inputText id="eMail"
11.                                  pa:type="email"     pa:list="emails"
12.                 value="#{umBean.admin.email}"
13.        </h:panelGrid>
14.    </h:form>
15. </r:panel>
```

Quadro 5: Código RichFaces gerando *datalist* dinamicamente

O *keygen* não foi possível usar. O elemento até é gerado na página, mas não foi possível obter a chave gerada através de um bean.

O mesmo resultado foi encontrado com *output*. Não foi possível acessar dinamicamente os valores que estavam nos *beans* para efetuar o cálculo. O valor era buscado apenas na primeira vez em que se executava. Nos testes com *JQuery* também não foi possível obter o valor.

```
package beans;
import java.util.ArrayList;
import java.util.List;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;

@ManagedBean
@RequestScoped
public class UmBean {

    public UmBean(List<String> emailsValidos) {
        this.emailsValidos = emailsValidos; }

    public List<String> getEmailsValidos() {
        return emailsValidos; }

    public void setEmailsValidos(List<String> emailsValidos) {
        this.emailsValidos = emailsValidos; }

    private List<String> emailsValidos;
```

```
    public UmBean() {
        emailsValidos = new ArrayList<String>();
        emailsValidos.add("laheinzen@gmail.com");
        emailsValidos.add("fernando.santos@udesc.br");}
}
```

Quadro 6: Código do Bean que gera os valores dinamicamente

4.3. Teste dos novos atributos para formulários

Mais uma vez entra em jogo a passagem de atributos e o uso do renderizador HTML do JSF, o que acaba igualando as bibliotecas nesse aspecto.

O atributo *autocomplete* foi gerado com sucesso: ao usar o formulário passando o atributo em questão (*<h:form pa:autocomplete="off" .../>*) o HTML gerado recebe o atributo conforme definido. Já o *novalidate* só funciona quando a validação deveria ser feita pelo navegador. O *novalidate* não tem efeito caso o atributo JSF *required* seja configurado: nesse caso o JSF não valida, retornando para o usuário.

4.4. Teste dos novos atributos para elementos

No *Primefaces*, o componente *inputText* tem dois atributos que equivalem, diretamente a atributos HTML5: *autocomplete* e *placeholder*. O *inputTextArea* tem o *autocomplete*. Todos estes geram o mesmo atributo no HTML.

Os atributos *heigth/width* aparecem tanto no JSF padrão *grapichImage*, quanto no componente do PrimeFaces. Ambos geram um HTML que contem esses mesmos atributos no HTML.

Os atributos *min*/*max* são gerados ao usar um *spinner* no PrimeFaces usando os atributos com o mesmo nome. No RichFaces o *inputNumberSpinner* tem os atributos *minValue* e *maxValue* mas estes não se traduzem no *min*/*max* do HTML.

Para os demais atributos do PrimeFaces, temos que fazer uso da passagem de atributos.

Com relação ao RichFaces não há como se falar em passagem de atributos pois o mesmo depende das passagens (elementos e atributos) do JSF para gerar componentes HTML5. Com isso a passagem funciona da mesma maneira que para os novos tipos – uma vez que o tipo nada mais é do que um atributo dos campos *input*.

O *autocomplete* foi testado e se comportou conforme o esperado, assim como o *autofocus*. O único porém do *autofocus* é quando o mesmo é usado com elementos como o *colorPicker* do PrimeFaces: não é visível para o usuário que o foco está no controle.

O atributo *form* só pode ser usado se o formulário não tiver sido gerado através do JSF. A razão para isso é que o atributo pede o *id* do formulário e este é gerado pelo JSF.

O atributo *formtarget* funcionou como esperado. As mesmas observações feitas para o atributo *novalidate* do formulário valem para o *formnovalidate* dos campos: só vale para o caso da validação do próprio navegador HTML.

Os atributos *formenctype*, *formmethod* e *formmethod* conseguem ser gerados no HTML. Mas nossos testes se limitaram a testar a presença dos mesmos. Explica-se: o JSF é quem controla automaticamente esses aspectos, e mudar tais configurações pode prejudicar o funcionamento do JSF.

O atributo *list* funciona em conjunto com o *datalist* do formulário, e conforme mencionado na seção anterior, é possível usar a combinação elemento *datalist* no formulário e o atributo *list* no elemento, para um elemento cujo código JSF gere um campo do tipo *input*.

O atributo *pattern* foi testado e, como no caso do atributo *required*, que é detalhado em seguida, a mensagem de erro varia de navegador para navegador. "Por favor, satisfaça o formato requisitado" é a mensagem apresentada pelo Firefox.

No caso do atributo *required*, o mesmo está presente tanto no HTML5 quanto no JSF. Nos navegadores em que há suporte ao atributo HTML5, o comportamento varia: isso ocorre porque a implementação depende do navegador. O HTML5 dita que o atributo deve existir, mas não como ele deve se comportar. Ao tentar enviar um formulário, com um campo obrigatório não preenchido, o Chrome torna a borda do campo azul, apresenta um ícone de exclamação, de fundo amarelo e mostra a mensagem "Preencha este campo". Já o Firefox dispensa o ícone, mas coloca a borda em azul, e altera a mensagem para "Por favor, preencha este campo". O Opera não traz nenhuma mensagem, apenas coloca o fundo como azul. O Internet Explorer coloca a borda azul e traz a mensagem "Este campo é obrigatório". O Safari não dá suporte ao atributo.

Foi testada a redundância: mescla dos atributos HTML5 e JSF. Não altera o comportamento nos navegadores que dão suporte, mas no Safari o comportamento muda, conforme a ordem dos atributos. Se o atributo do JSF vier por último, ocorre a validação do servidor. Caso contrário, não. Foi usado o serviço *BrowserStack* (www.browserstack.com) e o mesmo comportamento ocorre em outros navegadores onde não há suporte ao atributo.

Já para o caso do atributo do JSF, o comportamento é sempre o mesmo uma vez que a implementação decorre da biblioteca, não dependendo do navegador a ser utilizado.

Considerações Finais

Ao final do trabalho concluiu-se que as novidades do HTML5 já são contempladas nas bibliotecas e que estas fazem uso dos componentes e funcionalidades por padrão. Principalmente no que tange aos novos elementos, o PrimeFaces permite uma abordagem mais intuitiva, exigindo apenas que se mude o atributo *type* dos elementos *inputText*. O RichFaces não tem o mesmo comportamento, exigindo que se faça uso das funcionalidades de passagem de atributos e elementos.

Em relação aos novos atributos, são poucos os que não exigem que se faça uso da passagem de atributos o que acaba tornando as bibliotecas parecidas.

Pelo fato de ser mais fácil gerar os novos elementos e fazer uso de mais atributos que se traduzem em equivalentes HTML (*min*, *max* e *placeholder*, por exemplo) este trabalho conclui que o PrimeFaces é o mais amigável em relação a criação de páginas que fazem uso dos novos recursos introduzidos no HTML5.

Mas existem vantagens nessa abordagem das bibliotecas em preferirem seus componentes e funcionalidades, uma vez que isso significa não ter que depender do suporte por parte do navegador e há uma padronização no comportamento do aplicativo, ao contrário do HTML5. Um exemplo disso é a imprevisibilidade da mensagem ao não fornecer um campo requerido: a mensagem apresentada é de responsabilidade do navegador, não podendo ser personalizada.

Ao final das análises realizadas neste trabalho, pode-se constatar que o HTML5, apesar de promissor, ainda tem problemas, como o não suporte a alguns elementos em certos navegadores, bem como uma não padronização dos comportamentos. É razoável esperar que as bibliotecas tentassem fazer uso do HTML quando o navegador desse suporte, usando seus componentes quando não: mas isso não ocorreu.

O presente trabalho pode ser complementado fazendo a mesma análise sob a ótica de dispositivos móveis: testar o suporte dos navegadores móveis e também se o HTML gerado pelo JSF gera páginas que fazem uso do design responsivo – se adequam dinamicamente ao dispositivo. Além disso, tal pesquisa pode ser novamente no futuro, quando o suporte a HTML5 estiver mais consolidado. Por fim, outro trabalho válido seria testar as funcionalidades que não foram analisadas neste trabalho: os elementos gráficos, os elementos multimídia e as APIS.

Referências

CARMISINI, A; VAHLDICK, A. Comparativo entre Frameworks de Javaserver Faces: **A**pache Tobago, Primefaces e Richfaces. **Revista Eletrônica do Alto Vale do Itajaí**, Ibirama, Nº 02

págs. 10 – 18. Dezembro. 2012. Disponível em: http://revistas.udesc.br/index.php/reavi/article/view/2889/2183. Acesso em 30 mar. 2014.

ESTRADA, MIRIAM E. J. **Análisis Comparativo sobre los Frameworks Myfaces, IceFaces y RichFaces Aplicado al Sistema Nutricional de la ESPOCH**. 2013. 218p. Tese de diplomação da Faculdade de Informática e Eletrônica – Escuela Superior Politécnica de Chimborazo, Riobamba, Equador. Disponível em: http://dspace.espoch.edu.ec/handle/123456789/2530. Acesso em 30 mar. 2014.

FRANCO, REBECA S. T. **Estudo comparativo entre frameworks Java para desenvolvimento de aplicações web**: JSF 2.0, Grails e Spring Web MVC. 2011. 89 p. Monografia de Especialização – Universidade Tecnológica Federal do Paraná. Curitiba, Paraná. Disponível em: http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/492/1/CT_JAVA_VI_2010_16.PDF. Acesso em 23 mar. 2014.

LEATHEM, BRIAN ; FRYC, LUKAS; ROGERS, SEAN. **RichFaces Developer Guide**: Develop applications using RichFaces 4. 2014. 92p. Disponível em: http://docs.jboss.org/richfaces/latest_4_3_X/Developer_Guide/en-US/ pdf/Developer_Guide.pdf. Acesso em 29 mar. 2014.

KATAMREDDY, SIVAPRASADREDDY. **PrimeFaces QuickStart Tutorial**: Part 1. 2011. Disponível em: http://java.dzone.com/articles/primefaces-quickstart-tutorial. Acesso em: 29 mar. 2014.

MOZILLA. **Introdução ao HTML5**. 2013. Disponível em: ttps://developer.mozilla.org/pt-BR/docs/HTML/HTML5/Introduction_to_HTML5. Acesso em 28 mar. 2014.

ORACLE. **The Java EE 7**: Tutorial. 2013. Disponível em: http://docs.oracle.com/javaee/7/tutorial/doc/javaeetutorial7.pdf. Acesso em 20 mar. 2014.

PITOŇÁK, PAVOL. **Comparison of AJAX JSF Libraries**: Functionality and Interoperability. 2011. 96p. Tese de Diplomação de Bacharelado em Informática – Universidade Masaryk, Brno, República Tcheca. Disponível em: http://is.muni.cz/th/207718/fi_m/thesis.pdf. Aceso em 30 mar. 2014.

PRIMEFACES. **IceFaces Copies PrimeFaces Line by Line.** 2012. Disponível em http://blog.primefaces.org/?p=1692. Acesso em 29 mar. 2014.

PRIMETEK. **PrimeFaces User Guide.** 2014. Disponível em http://primefaces.googlecode.com/files/primefaces_users_guide_4_0_edtn2.pdf. Acesso em 29 ago. 2014. RICHFACES. **Página do projeto RichFaces**. 2014. Disponível em http://richfaces.jboss.org/. Acesso em 29 ago. 2014.

SMIRNOV SERGEY, **What's new in RichFaces 3.1.0**. 2007. Disponível em: https://community.jboss.org/wiki/RichFacesWhatIsNewIn310. Acesso em 29 mar. 2014.

TIOBE. **TIOBE Index for March 2014.** 2014. Disponível em: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. Acesso em 28 mar. 2014.

W3. **Facts about W3C**. 2013. Disponível em http://www.w3.org/Consortium/facts. Acesso em 31 mar. 2014.

W3 SCHOOLS. **HTML5 Introduction**. 2013. Disponível em: http://www.w3schools.com/html/html5_intro.asp. Acesso em 28 mar. 2014.