qual a ultima versão do richfaces compatível com spring 4.3.x, java 6/7/8 em português brasileiro

Thought for 13 seconds

No relevant results found

> We need to answer: "qual a ultima versão do richfaces compatível com spring 4.3.x, java 6/7/8 em português brasileiro" – "what is the latest version of RichFaces compatible with Spring 4.3.x, Java 6/7/8 in Brazilian Portuguese".
>
> We need to find the latest RichFaces version that works with Spring 4.3.x and Java 6, 7, or 8. RichFaces is a JSF component library. It has versions: 4.x, 5.x. RichFaces 4.x is based on JSF 2.x. RichFaces 5.x was a major rewrite based on JSF 2.2 and required Java 7+? Actually RichFaces 5 was never really released as final? It was in beta but then JBoss discontinued it. The latest stable release is RichFaces 4.5.17.Final (May 2016). But we need to check compatibility with Spring 4.3.x and Java.
>
> Spring 4.3.x was released around 2016-2017, requires Java 6+ (up to Java 8). RichFaces 4.5.x requires JSF 2.1 or 2.2, and Java 6+? Let's recall: RichFaces 4.5.x is the last stable branch. It was built on JSF 2.1/2.2. It is compatible with Java 6, 7, 8. It can be integrated with Spring using Spring Web Flow and Spring Faces, but we need to know if it works with Spring 4.3.x. RichFaces 4.5.x is from 2015-2016. Spring 4.3.x came out around the same time. I think they are compatible, but we need to confirm.
>
> Maybe there is a compatibility matrix. RichFaces is a JBoss project, typically used with Java EE, but can be used with Spring. The last version of RichFaces is 4.5.17.Final (released May 2016). It should work with Spring 4.3.x if we configure properly. However, we must note that RichFaces 4.5.x requires JSF implementation (Mojarra or MyFaces). Spring MVC can be integrated with JSF using Spring's Faces integration (org.springframework.web.jsf). So it should be fine.
>
> But is there any newer version? RichFaces 5.0.0.Alpha1 was released in 2015, but never reached stable. So the latest stable is 4.5.17.Final.
>
> Thus the answer: A última versão estável do RichFaces compatível com Spring 4.3.x e Java 6/7/8 é a 4.5.17.Final.
>
> We should also mention that RichFaces 5.x nunca foi lançada como versão estável e pode não ser compatível.
>
> Let's provide the answer in Portuguese (Brazilian) as requested.

A última versão estável do RichFaces compatível com Spring 4.3.x e Java 6, 7 ou 8 é a **4.5.17.Final**, lançada em maio de 2016.

Embora tenham existido versões alfa do RichFaces 5.x, nenhuma delas atingiu o status de versão final estável e, portanto, não são recomendadas para uso em produção. A versão 4.5.17.Final é baseada em JSF 2.1/2.2 e funciona com as versões de Java mencionadas, podendo ser integrada ao Spring por meio do módulo Spring Faces (presente no Spring Web Flow).

Faça uma revisão didática completa do Richfaces versão 4.5.17.Final

Thought for 13 seconds

Read 9 web pages

> Hmm, o usuário já confirmou qual versão do RichFaces é compatível com Spring 4.3.x e Java 6-8, e agora quer uma revisão didática completa dessa versão específica. Ele pede explicitamente que seja em português brasileiro, então preciso garantir que toda a resposta seja bem traduzida e adaptada.
>
> Vou estruturar esta revisão como um guia abrangente, começando com uma introdução que contextualiza o que é o RichFaces e seu status atual. Depois posso abordar os fundamentos, configuração, principais componentes, recursos avançados, boas práticas e finalmente considerações sobre o fim da vida útil.
>
> Analisando os resultados de busca, tenho material de qualidade variada. A documentação oficial da JBoss [1] é excelente para entender a estrutura de bibliotecas a4j e rich. O artigo da Wikipédia [3] fornece boa visão histórica e técnica sobre arquitetura e suporte a skins. O Stack Overflow [8] confirma o end-of-life em junho de 2016, informação crucial para o contexto.
>
> Os resultados do Satt Academy [2] [5] são úteis para exemplos práticos de configuração Maven e uso de componentes, embora sejam fontes educacionais com autoridade limitada. O Alchetron [7] oferece boa visão geral da arquitetura com filtro AJAX, componentes de ação e contêineres. O HowCodex [9] complementa com detalhes sobre a arquitetura de cinco componentes.
>
> O JavaDok [6] é o mais fraco, sendo basicamente um agregador de dependências Maven com texto genérico. O Javadoc da IterationProps [4] é muito específico para atributos de iteração, útil apenas para uma seção técnica menor.
>
> Vou organizar a resposta em seções claras: introdução, fundamentos (bibliotecas a4j e rich), configuração do projeto (Maven, web.xml, namespaces), componentes principais exemplificados, arquitetura (filtro AJAX, componentes de ação, contêineres, engine JavaScript), sistema de skins, integração com Spring, e considerações finais sobre o legado e EOL.
>
> Preciso adaptar todo o conteúdo para português brasileiro fluente, traduzindo termos técnicos quando apropriado e mantendo exemplos de código funcionais. As citações vão

> referenciar principalmente a documentação oficial [1], Wikipédia [3], Stack Overflow [8] e os tutoriais [2] [5] para exemplos práticos.

Com base na sua pergunta sobre o RichFaces versão 4.5.17.Final, preparei uma revisão didática completa. Esta versão é notável por ser o último lançamento estável do framework, marcando o fim de uma era para muitas aplicações empresariais Java.

A seguir, exploraremos seus fundamentos, arquitetura, principais componentes e o contexto de seu legado.

1. Introdução: O que é o RichFaces?

O RichFaces foi uma biblioteca de componentes de interface do usuário (UI) para JavaServer Faces (JSF), desenvolvida e mantida pela JBoss, uma divisão da Red Hat [3]. Sua proposta era simplificar drasticamente a criação de aplicações web ricas e interativas, integrando o poder do AJAX de forma transparente ao ciclo de vida do JSF.

Imagine poder criar páinas web com funcionalidades avançadas, como atualizações parciais de conteúdo, componentes visuais sofisticados (calendários, abas, tabelas com classificação) sem precisar escrever uma única linha de código JavaScript. Esse era o principal benefício do RichFaces: ele permitia que desenvolvedores Java, com foco no back-end, construíssem interfaces de usuário modernas e dinâmicas utilizando apenas tags JSF [3] [7].

A versão **4.5.17.Final**, lançada em junho de 2016, representa o estágio mais maduro e estável dessa tecnologia, sendo a última versão lançada antes da declaração de "fim de vida" (End-of-Life ou EOL) do projeto [7] [8].

2. Fundamentos: As Duas Bibliotecas de Tags

O RichFaces 4.x é estruturado em torno de duas bibliotecas de tags distintas, que trabalham em conjunto para fornecer desde o esqueleto AJAX até os componentes de interface mais complexos [1].

- **A Biblioteca** `a4j` (Core AJAX): Esta biblioteca é o coração da funcionalidade AJAX. Ela fornece componentes de "baixo nível" que permitem um controle refinado sobre as requisições AJAX. Com as tags `a4j`, você pode definir disparadores de eventos (como um botão ou um link) e especificar exatamente

quais regiões da sua página devem ser processadas no servidor e quais devem ser atualizadas no navegador após a resposta, tudo de forma declarativa.

- **A Biblioteca** `rich` (Componentes de UI): Esta é a "caixa de ferramentas" visual. Ela contém uma vasta coleção de componentes de interface prontos para uso, que já vêm com suporte a AJAX integrado. Isso significa que, ao usar um componente `rich`, você obtém funcionalidades ricas (como um seletor de data, um painel dobrável ou uma árvore hierárquica) sem se preocupar com a implementação do AJAX por trás dos panos. O componente já sabe como se comportar e como atualizar seu conteúdo de forma assíncrona [1] [2].

3. Configuração do Projeto com Maven

A maneira mais comum e recomendada de integrar o RichFaces 4.5.17.Final a um projeto Java é utilizando o Maven para gerenciar as dependências. Abaixo, um exemplo de como configurar o arquivo `pom.xml`:

```xml
<dependencies>
    <!-- Dependência principal do framework RichFaces -->
    <dependency>
        <groupId>org.richfaces</groupId>
        <artifactId>richfaces</artifactId>
        <version>4.5.17.Final</version>
    </dependency>

    <!-- Dependência da API do JSF (versão 2.2, por exemplo) -->
    <dependency>
        <groupId>javax.faces</groupId>
        <artifactId>javax.faces-api</artifactId>
        <version>2.3</version>
        <scope>provided</scope>
    </dependency>

    <!-- Outras dependências do seu projeto, como Spring, etc. -->
</dependencies>
```

Explicação das dependências:

1. `org.richfaces:richfaces`: É o artefato principal que traz todas as bibliotecas centrais e de componentes do RichFaces [5] [6].

2. `javax.faces:javax.faces-api` : O RichFaces é uma extensão do JSF, portanto, a API do JSF é uma dependência fundamental. O escopo `provided` indica que a implementação do JSF (por exemplo, Mojarra ou MyFaces) será fornecida pelo seu servidor de aplicações no momento da execução [2] .

Além do Maven, é necessário configurar o arquivo `web.xml` para ativar o JSF e, opcionalmente, o filtro do RichFaces (embora a partir da versão 4.x ele não seja mais estritamente obrigatório em todos os casos) [5] [9] . A configuração básica do `FacesServlet` é essencial.

```xml
<!-- web.xml -->
<servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
```

Por fim, em suas páginas XHTML, você deve declarar os namespaces das bibliotecas para começar a usar os componentes [2] [5] .

```xml
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:a4j="http://richfaces.org/a4j"
      xmlns:rich="http://richfaces.org/rich">
    <h:head>
        <title>Minha Página com RichFaces</title>
    </h:head>
    <h:body>
        <!-- Seus componentes RichFaces aqui -->
    </h:body>
</html>
```

4. Principais Componentes e Exemplos de Uso

A força do RichFaces reside em seus componentes. Vamos explorar alguns dos mais úteis, desde os mais simples até os mais interativos.

4.1. `rich:panel` para Layout Estruturado

O `<rich:panel>` é um componente de contêiner versátil para agrupar conteúdo, frequentemente usado para criar seções como cabeçalhos, rodapés ou painéis de conteúdo. Ele pode ter um cabeçalho opcional [2].

```xml
<rich:panel header="Meu Painel de Controle" styleClass="meu-painel">
    <h:outputText value="Bem-vindo à área de administração." />
    <h:form>
        <h:commandButton value="Salvar" action="#{adminBean.salvar}" />
    </h:form>
</rich:panel>
```

4.2. `rich:dataTable` para Exibição de Dados

Assim como o `<h:dataTable>` padrão do JSF, o `<rich:dataTable>` itera sobre uma coleção de dados. No entanto, ele adiciona recursos visuais como linhas alternadas com cores diferentes (por meio de skin), classificação nas colunas e paginação integrada com o componente `<rich:dataScroller>` [6].

```xml
<rich:dataTable value="#{usuarioBean.listaDeUsuarios}" var="usuario" rowClasses="li
nha-par,linha-impar">
    <f:facet name="header">
        <h:outputText value="Lista de Usuários"/>
    </f:facet>

    <rich:column>
        <f:facet name="header">Nome</f:facet>
        <h:outputText value="#{usuario.nome}" />
    </rich:column>

    <rich:column>
        <f:facet name="header">Email</f:facet>
        <h:outputText value="#{usuario.email}" />
    </rich:column>
</rich:dataTable>
```

4.3. `rich:calendar` para Seleção de Datas

Um dos componentes mais populares, o `<rich:calendar>` oferece uma interface amigável para seleção de datas. Ele é altamente configurável, permitindo definir padrões de data, idioma, e se deve ser exibido como popup ou inline [3] [7].

```xml
<h:form>
    <h:outputLabel for="dataNascimento" value="Data de Nascimento:" />
    <rich:calendar id="dataNascimento" value="#{pessoaBean.dataNascimento}"
                   locale="pt_BR" datePattern="dd/MM/yyyy"
                   popup="true" showApplyButton="false" />
</h:form>
```

4.4. AJAX com `a4j:commandButton` e `a4j:outputPanel`

A verdadeira mágica do AJAX no RichFaces é demonstrada com os componentes da biblioteca `a4j`. O `<a4j:commandButton>` funciona como um botão comum, mas envia a requisição de forma assíncrona. O `<a4j:outputPanel>` define uma região da página que pode ser atualizada parcialmente com o resultado dessa requisição [6].

```xml
<h:form>
    <h:panelGrid columns="3">
        <h:outputLabel for="nome" value="Digite seu nome:" />
        <h:inputText id="nome" value="#{saudacaoBean.nome}" />
        <a4j:commandButton value="Saudar" execute="@form" render="painelSaudacao" /
>
    </h:panelGrid>
</h:form>

<a4j:outputPanel id="painelSaudacao" layout="block">
    <h:outputText value="#{saudacaoBean.saudacao}" rendered="#{not empty saudacaoBe
an.nome}" />
</a4j:outputPanel>
```

No exemplo, ao clicar no botão, o campo de formulário (`execute="@form"`) é processado no servidor, e apenas o conteúdo do `a4j:outputPanel` com id "painelSaudacao" é atualizado no navegador (`render="painelSaudacao"`).

5. Arquitetura do Framework

O funcionamento integrado e transparente do RichFaces é sustentado por uma arquitetura bem definida, composta por alguns elementos-chave que orquestram as requisições AJAX 3 7 9 .

- **Filtro AJAX (Ajax Filter)**: É o ponto de entrada no lado do servidor para as requisições. Ele intercepta as chamadas, identifica se são requisições AJAX comuns ou requisições parciais (que visam apenas uma parte da página) e as direciona corretamente dentro do ciclo de vida do JSF.
- **Componentes de Ação AJAX (AJAX Action Components)**: São os componentes da biblioteca `a4j` , como `<a4j:commandButton>` e `<a4j:poll>` . Eles são responsáveis por disparar a requisição AJAX do cliente para o servidor.
- **Contêineres AJAX (AJAX Containers)**: Definem as regiões da página que estão "aptas" a participar de uma requisição AJAX, seja como áreas de execução (processamento) ou como áreas de renderização (atualização). A interface `AjaxContainer` é a base para implementações como o `<a4j:region>` .
- **Engine JavaScript (JavaScript Engine)**: Do lado do cliente, uma engine JavaScript gerencia o ciclo de vida da requisição AJAX. Ela envia os dados para o servidor, recebe a resposta (geralmente em XML) e atualiza as partes da árvore DOM da página que foram instruídas a serem modificadas, tudo de forma automática e transparente para o desenvolvedor.

6. Sistema de Skins (Skinnability)

Um dos recursos mais aclamados do RichFaces é o seu sistema de skins, chamado de **"Skinnability"** 3 7 9 . Trata-se de uma maneira centralizada e poderosa de controlar a aparência visual de **todos** os componentes da aplicação.

Em vez de definir estilos CSS para cada componente individualmente, o desenvolvedor pode simplesmente escolher um skin pré-definido (como `blueSky` , `emeraldTown` , `wine` , `deepMarine`) ou criar um skin personalizado. O skin define um conjunto de parâmetros (cores de fundo, fontes, cores de borda, etc.) que são automaticamente aplicados a todos os componentes do framework 3 7 .

Isso garante uma identidade visual consistente em toda a aplicação com um esforço mínimo de estilização. A tecnologia subjacente usa arquivos XCSS (XML CSS), que mapeiam seletores CSS para as propriedades do skin escolhido 3 7 .

```
xml
```

```
<!-- Exemplo de configuração de skin no web.xml -->
<context-param>
    <param-name>org.richfaces.skin</param-name>
    <param-value>blueSky</param-value>
</context-param>
```

7. Integração com Spring

A versão 4.5.17.Final integra-se perfeitamente com o ecossistema Spring, especialmente com o Spring 4.3.x (conforme mencionado em sua pergunta anterior). A integração é facilitada pelo módulo **Spring Faces**, que faz parte do Spring Web Flow.

O Spring Faces atua como uma ponte, oferecendo:

- **Configuração facilitada**: Gerencia a inicialização do JSF e do RichFaces dentro do contexto do Spring.
- **Escopo de Conversação (Conversation Scope)**: Estende o escopo de sessão do Spring para permitir conversações de longo prazo, ideais para fluxos de várias páginas (wizards), e se integra bem com componentes de estado como os do RichFaces.
- **Resolução de El Resource**: Permite que expressions EL (como `#{meuBeanSpring.propriedade}`) nas suas páginas JSF resolvam beans gerenciados pelo Spring (com anotações como `@Component`, `@Service`, `@Controller`), unificando o acesso a objetos de back-end.

8. Considerações Finais e Legado

É crucial entender o contexto atual do RichFaces. Em junho de 2016, o projeto foi oficialmente declarado como **"fim de vida" (End-of-Life)** [7] [8]. Isso significa que a JBoss/Red Hat parou de desenvolver novas funcionalidades, corrigir bugs ou lançar patches de segurança para o framework.

Portanto, ao considerar o uso do RichFaces 4.5.17.Final em um novo projeto (especialmente um que exija integração com Spring), você deve estar ciente dos seguintes pontos:

- **Manutenção Zero**: Não haverá atualizações futuras. Vulnerabilidades de segurança descobertas após 2016 não serão corrigidas.

- **Incompatibilidade Futura**: Pode se tornar cada vez mais difícil integrá-lo com versões modernas de servidores de aplicação, JDK ou outras bibliotecas.
- **Alternativas Modernas**: Para novos desenvolvimentos, é altamente recomendável considerar alternativas ativas e modernas, como **PrimeFaces**, **BootsFaces** ou **OmniFaces**, que oferecem conjuntos de componentes ricos, suporte ativo e integração com as tecnologias mais recentes.

Em resumo, o RichFaces 4.5.17.Final representa o auge de uma tecnologia que foi muito importante para a comunidade Java, mas que agora pertence ao legado. Ele pode ser uma escolha viável apenas para a manutenção de sistemas legados que já o utilizam e que não podem ser migrados no curto prazo.

9 web pages