

Introdução ao JavaServer Faces 2.x

Adicionando Suporte JSF 2.x a uma Aplicação Web

Criando um Feijão Gerenciado

Utilizando o Assistente de Bean Gerenciado

Criando um Construtor

Adicionando Propriedades

Conectando Beans Gerenciados às Páginas

index.xhtml

response.xhtml

Aplicando um Modelo de Facelets

Criando o Arquivo de Modelo de Facelets

Criando Arquivos de Clientes de Modelo

Consulte Também

Tutoriais e Artigos NetBeans

Recursos Externos

Blogs

O JavaServer Faces (JSF) é um framework de interface de usuário (IU) para aplicações Java Web. Foi projetado para complacência, trabalha tarefa de escrever e manter as aplicações que são executadas em um servidor de aplicações Java e renderizar como UIs de volta a um cliente de destino. O JSF oferece facilidade de uso das seguintes formas:

- Facilita a construção de uma IU usando um conjunto de componentes de IU reutilizáveis
- Simplifica a migração de dados da aplicação para IU e provenientes dela
- Ajude a gerenciar o estado da IU nas solicitações do servidor
- Oferece um modelo simples para conectar os eventos gerados pelo cliente ao código da aplicação do servidor
- Permite personalizar os componentes de UI para que sejam facilmente construídos e reutilizados

Para obter uma descrição mais completa do framework JSF, consulte o Tutorial do Java EE 7, Capítulo 12: Desenvolvendo Tecnologia do JavaServer Faces .

Este tutorial demonstra como você pode aplicar o suporte do JSF 2.x a uma aplicação Web utilizando o NetBeans IDE. Comece adicionando o suporte ao framework JSF 2.x a uma aplicação Web básica e continue para executar as seguintes tarefas:

- crie um bean gerenciado pelo JSF para manipular os dados solicitados,
- conecte o bean gerenciado às páginas Web da aplicação e
- converta as pages Web em arquivos de modelo de Facelets.

O NetBeans IDE oferece, há muito tempo, suporte ao JavaServer Faces. A partir do lançamento JSF 2.0 e Java EE 6, o NetBeans IDE oferece suporte para JSF 2.0 e JSF 2.1. Para obter mais informações, consulte Suporte JSF 2.x no NetBeans IDE .

Para concluir este tutorial, você precisa dos seguintes recursos e softwares.

Software ou Recurso	Versão Necessária
NetBeans IDE	Pacote Java EE 7.2, 7.3, 7.4, 8.0
JDK (kit de desenvolvimento Java)	7 ou 8
GlassFish Server	Edição de código aberto 3.x ou 4
Projeto de Aplicação Web `jsfDemo`	WL


Observações:

- O pacote Java do NetBeans IDE também inclui o GlassFish Server, um servidor compatível com Java EE necessário para este tutorial.
- Para comparar seu projeto a uma solução que funcione, faça o download do projeto de amostra completo.


Adicionando Suporte JSF 2.x a uma Aplicação Web

Comece abrindo o `jsfDemo` projeto da aplicação Web no IDE. Depois de abrir o projeto no IDE, você pode adicionar suporte ao framework utilizando a janela Propriedades do projeto.

O IDE também permite criar novos projetos com o suporte JSF 2.x. Para obter mais informações, consulte Criando um Novo Projeto com o Suporte a JSF 2.x.

1. Clique no botão () para Abrir o Projeto na barra de ferramentas principal do IDE ou pressione Ctrl-Shift-O (⌘-Shift-O no Mac).
2. Na caixa de diálogo Abrir Projeto, navegue até o local do computador onde você armazenou o projeto descompactado do tutorial. Selecione-o e clique em Abrir Projeto para abri-lo no IDE.

Observação. Você será solicitado a resolver a referência às bibliotecas JUnit quando abrir o projeto NetBeans, caso não tiver instalado o plug-in JUnit quando instalou o IDE.

1. Execute o projeto para ver como ele é em um browser. Clique com o botão direito do mouse no nó `jsfDemo` do projeto, na janela Projetos e selecione Executar ou clique no botão () Executar Projeto na barra de ferramentas principal. O projeto é encapsulado e implantado no GlassFish Server e o browser é aberto para exibir a página de boas-vindas (`index.xhtml`).

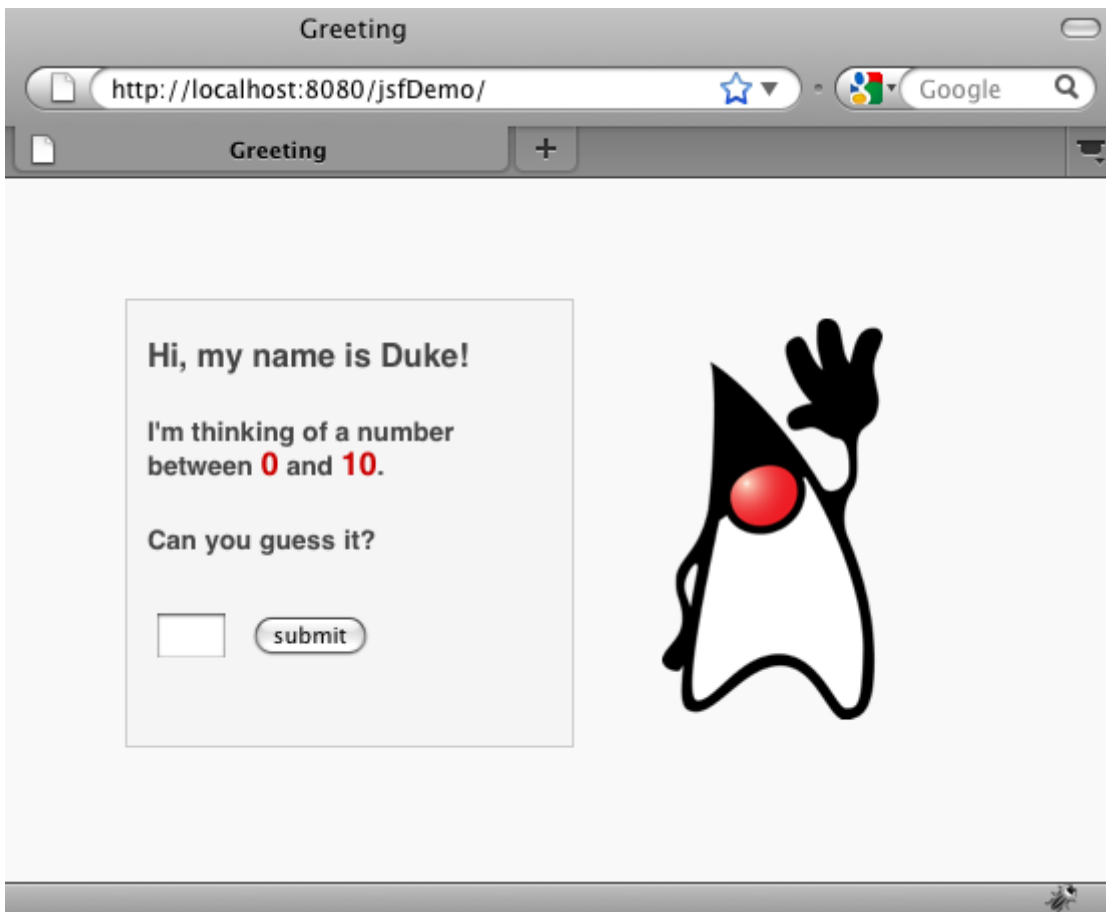


Figure 1. Executar o projeto para exibi-lo em um browser

1. Clique no botão Submeter. A página de resposta (`response.xhtml`) é exibida desta forma:

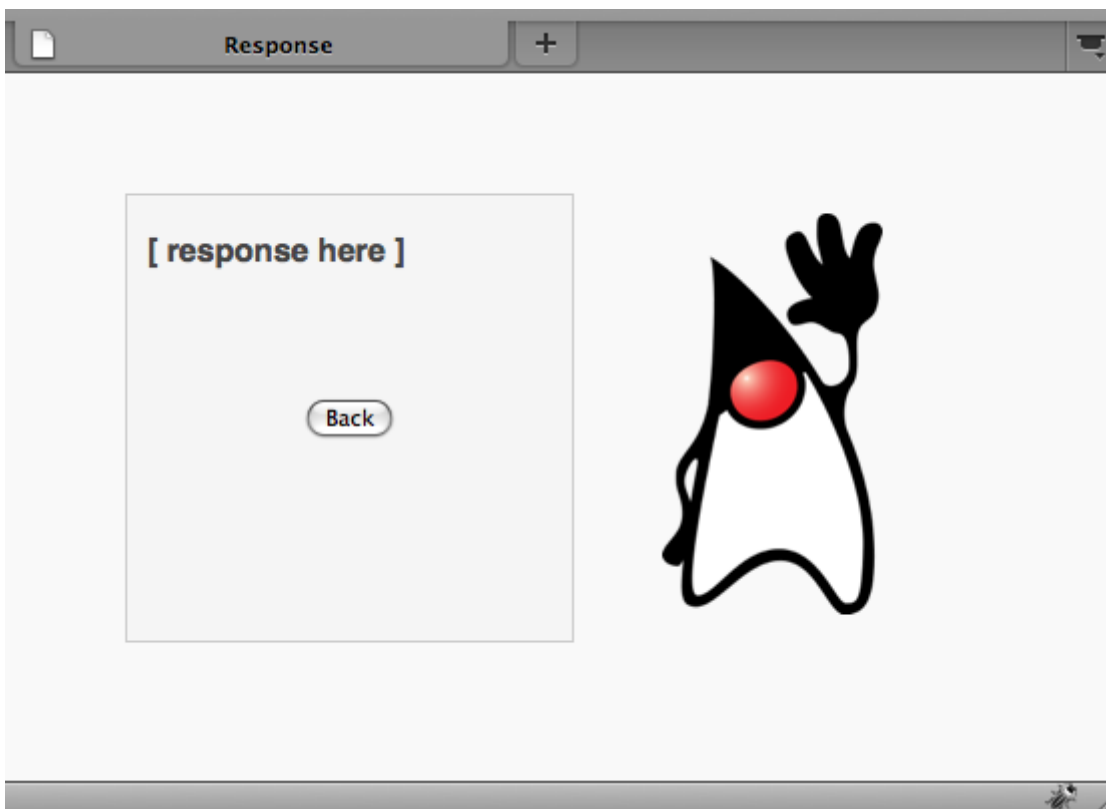


Figure 2. A página de boas-vindas e de resposta, atualmente, são páginas estáticas. No momento, as páginas de boas-vindas e resposta são estáticas e, juntamente com o arquivo `stylesheet.css` e a imagem `duke.png`, são os únicos arquivos da aplicação acessíveis do browser.

1. Na janela Projetos (Ctrl-1; ⌘-1 no Mac), clique com o botão direito do mouse no nó do projeto e escolha Propriedades para abrir a janela Propriedades do Projeto.
2. Selecione a categoria Frameworks e, em seguida, clique no botão Adicionar.
3. Selecione JavaServer Faces na caixa de diálogo Adicionar um Framework. Clique em OK.

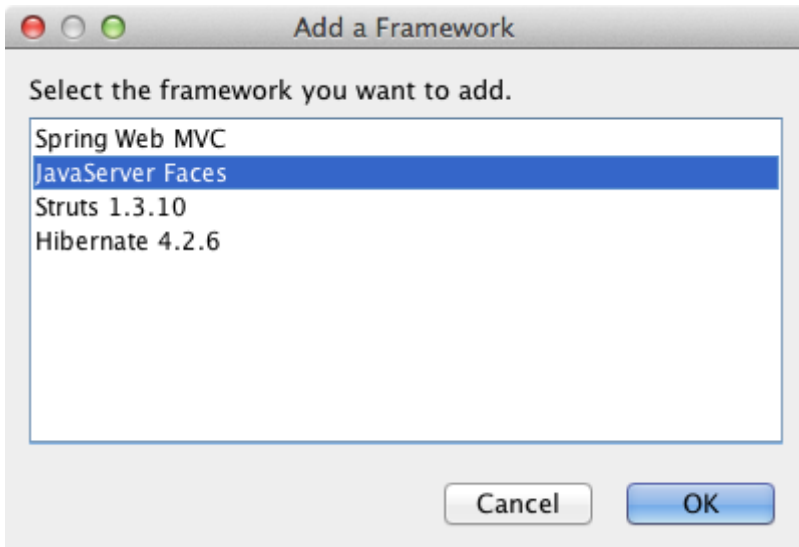


Figure 3. Adicionar suporte JSF a um projeto existente

Após selecionar JavaServer Faces, diversas opções de configuração ficarão disponíveis. Na guia Bibliotecas, você pode especificar como o projeto acessa as bibliotecas JSF 2.x. A versão de JFS disponível dependerá da versão do IDE e do GlassFish Server. A opção default é utilizar as bibliotecas fornecidas com o servidor (o GlassFish Server). No entanto, o IDE também integra as bibliotecas JSF 2.x. (É possível selecionar a opção Bibliotecas Registradas se desejar que seu projeto as utilize.)

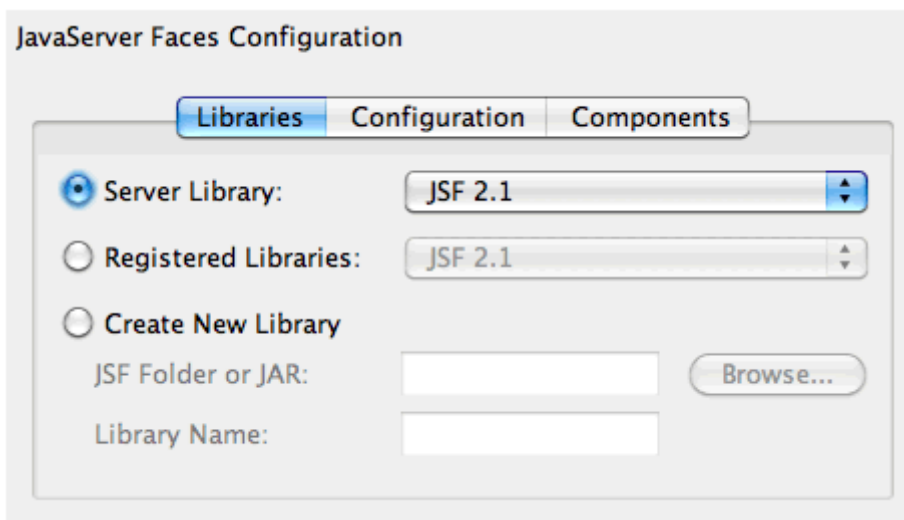


Figure 4. Especificar o acesso a bibliotecas do JSF 2.x

1. Clique na guia Configuração. É possível especificar como o servlet Faces é registrado no descritor de implantação do projeto. Também é possível indicar se você deseja utilizar Facelets ou páginas JSP no projeto.

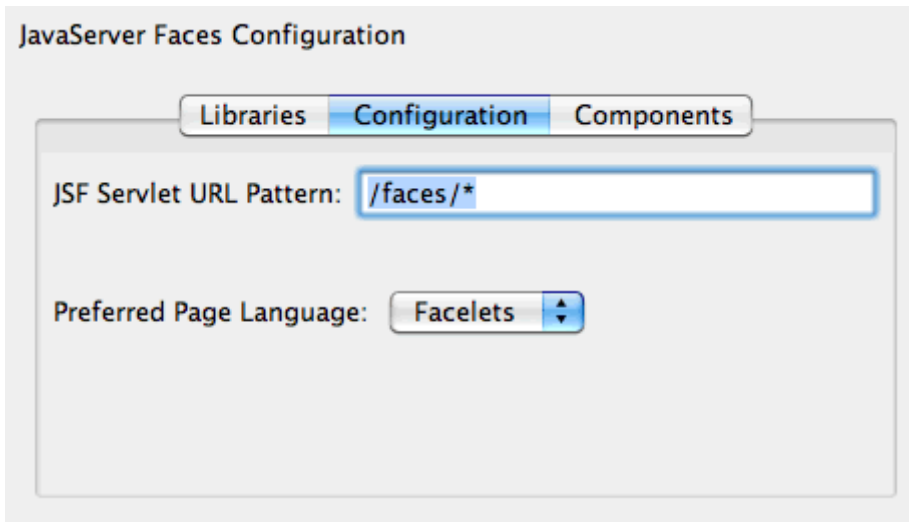


Figure 5. Especificar idioma preferencial e opções de servlet do Faces

Você também pode configurar facilmente seus projetos para usar vários conjuntos de componentes do JSF, na guia Componentes. Para usar um conjunto de componentes, é preciso fazer o download das bibliotecas necessárias e usar o Gerenciador de bibliotecas Ant para criar uma nova biblioteca contendo as bibliotecas do conjunto de componentes.

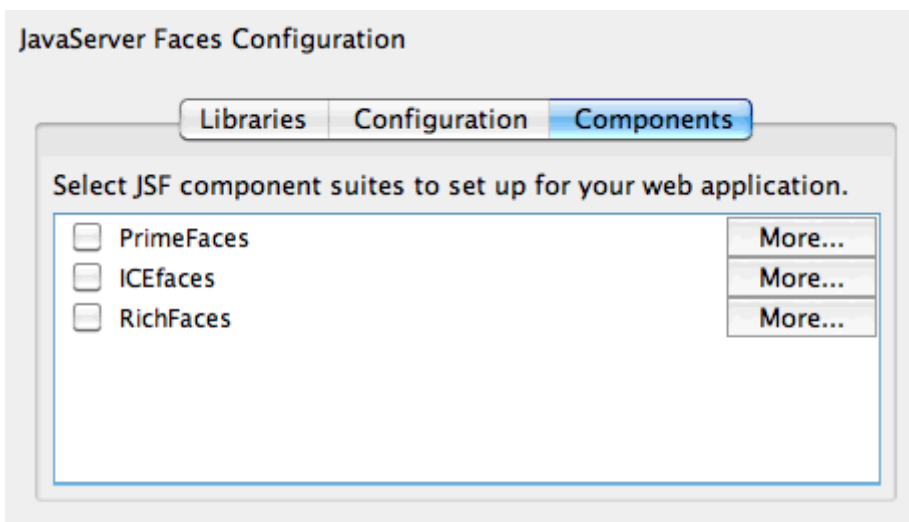


Figure 6. Especificar idioma preferencial e opções de servlet do Faces

1. Clique em OK para concluir as alterações e sair da janela Propriedades do Projeto.

Depois de adicionar o suporte JSF ao seu projeto, o descritor de implantação `web.xml` do projeto é modificado para que tenha a aparência a seguir: (Alterações em **negrito**.)

```
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
  *<context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <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>/faces/*</url-pattern>
  </servlet-mapping>
</web-app>
```

```
</servlet-mapping>*
<welcome-file-list>
    <welcome-file>*faces/*index.xhtml</welcome-file>
</welcome-file-list>
</web-app>
```

Importante: confirme se `web.xml` contém só uma entrada `<welcome-file>` e se a entrada contém `'faces/'` conforme mostrado no exemplo. Isso assegura que a página de boas-vindas do projeto (`index.xhtml`) passe pelo servlet do Faces antes de ser exibida no browser. Isso é necessário para renderizar apropriadamente os componentes da biblioteca de tags do Facelets.

O servlet do Faces é registrado no projeto e a página de boas-vindas `index.xhtml` passa pelo servlet do Faces quando é solicitada. Observe também que foi adicionada uma entrada no parâmetro de contexto `PROJECT_STAGE` . A definição desse parâmetro como `'Desenvolvimento'` fornece informações úteis quando a aplicação é depurado.

Consulte http://blogs.oracle.com/rlubke/entry/jsf_2_0_new_feature2 para obter mais informações.

É possível localizar as bibliotecas JSF expandindo o nó Bibliotecas do projeto na janela Projetos. Se você estiver usando as bibliotecas default incluídas com o GlassFish Server 3.1.2 ou GlassFish Server 4, este será o `javax.faces.jar` visível no nó GlassFish Server. (Se você estiver usando uma versão mais antiga do GlassFish, verá as bibliotecas `jsf-api.jar` e `jsf-impl.jar` em vez de `javax.faces.jar`.)

O suporte JSF 2.x do IDE inclui principalmente vários assistentes específicos do JSF e a funcionalidade especial fornecida pelo editor de Facelets. Você ira explorar esses recursos funcionais nas etapas a seguir. Para obter mais informações, consulte Suporte JSF 2.x no NetBeans IDE.

’ Criando um Bean Gerenciado

É possível utilizar os beans gerenciados do JSF para processar dados do usuário e retê-los entre as solicitações. Um bean gerenciado é um POJO (Objeto Java Simples Antigo) que pode ser utilizado para armazenar dados e é gerenciado pelo contêiner (por exemplo, o GlassFish Server) utilizando o framework JSF.

Um POJO é essencialmente uma classe Java que contém um construtor público sem argumentos e está em conformidade com as convenções de nomenclatura do JavaBeans para suas propriedades.

Ao observar a página estática produzida ao executar o projeto, você precisará de um mecanismo que determine que o número inserido pelo usuário corresponde ao número selecionado atualmente e que ele retorne uma view apropriada para esse resultado. Utilize o assistente de Bean Gerenciado para criar um bean gerenciado para essa finalidade. As páginas de Facelets que você criará na próxima seção precisarão acessar o número digitado pelo usuário e a resposta gerada. Para ativar esta opção, adicione as propriedades `userNumber` e `response` ao Bean gerenciado.

- Utilizando o Assistente de Bean Gerenciado
- Criando um Construtor
- Adicionando Propriedades

Utilizando o Assistente de Bean Gerenciado

1. Na janela Projetos, clique com o botão direito do mouse no nó do projeto `jsfDemo` e selecione **Novo > Bean Gerenciado pelo JSF**. (Se o Bean Gerenciado não estiver listado, selecione **Outros**. Em seguida, selecione a opção **Bean Gerenciado pelo JSF** na categoria **JavaServer Faces**. Clique em **Próximo**.)
2. No assistente, informe o seguinte:
 - **Nome da Classe:** `UserNumberBean`
 - **Pacote:** `guessNumber`
 - **Nome:** `UserNumberBean`
 - **Escopo:** `Session`

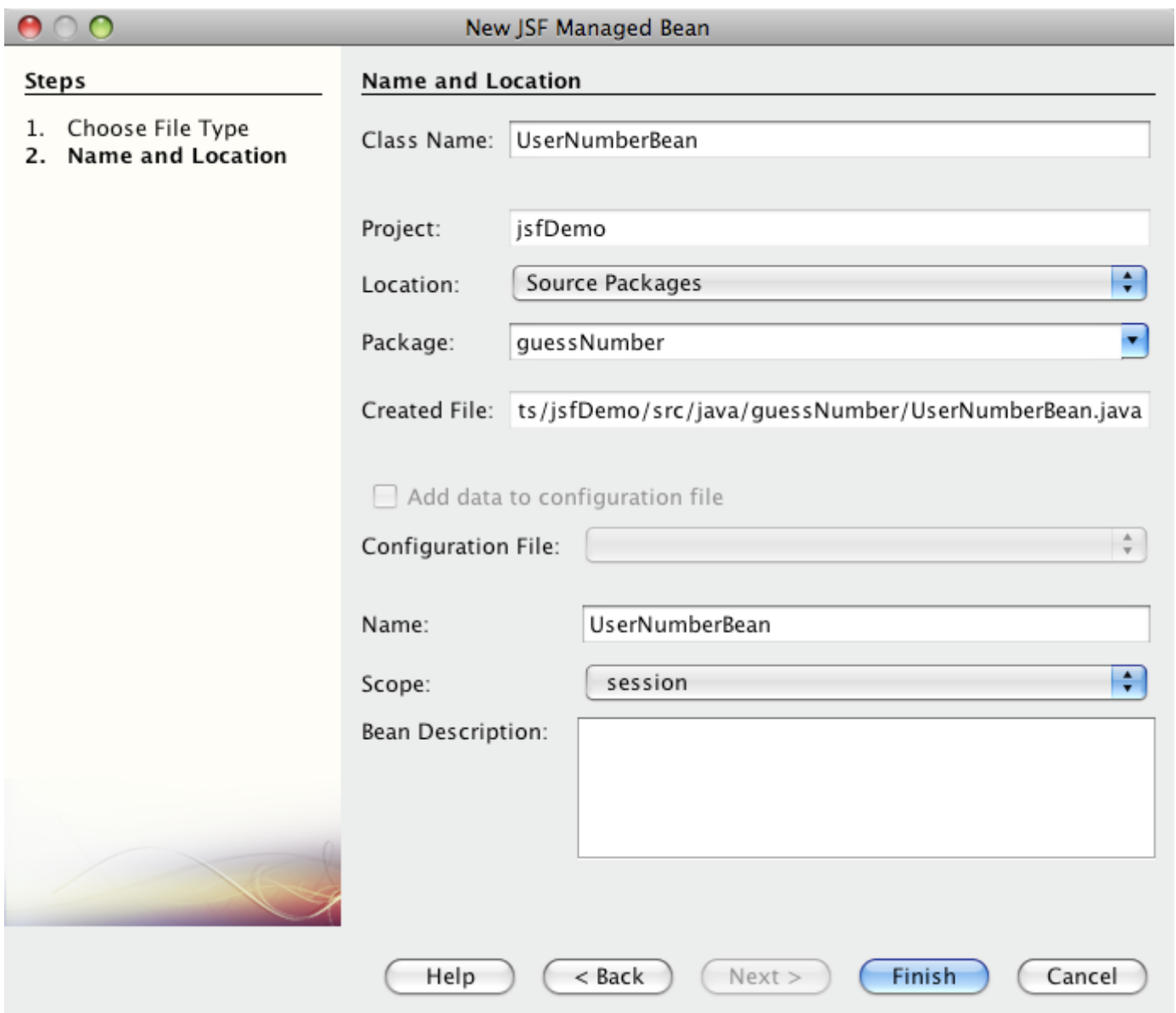


Figure 7. Utilizar o Ben Gerenciado pelo JSF para criar um novo Bean gerenciado

1. Clique em **Finalizar**. A classe `UserNumberBean` é gerada e aberta no editor. Observe as anotações a seguir (mostradas em **negrito**):

```

package guessNumber;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

/**
 *
 * @author nbuser
 */
@ManagedBean(name="UserNumberBean")
@SessionScoped
public class UserNumberBean {

    /** Creates a new instance of UserNumberBean */
    public UserNumberBean() {
    }

}

```

Como você está utilizando o JSF 2.x, você pode declarar todos os componentes específicos do JSF utilizando as anotações. Nas versões anteriores, era necessário declará-los no arquivo de configuração do Faces (`faces-config.xml`).

Para exibir o Javadoc para todas as anotações JSF 2.1, consulte Especificações de Anotações de Bean Gerenciado pelo Faces.

’ Criando um Construtor

O construtor `UserNumberBean` deve gerar um número aleatório entre 0 e 10 e armazená-lo em uma variável de instância. Isso forma parcialmente a lógica de negócios da aplicação.

1. Defina um construtor para a classe `UserNumberBean` . Insira o código a seguir (alterações exibidas em **negrito**).

```

public class UserNumberBean {

    *Integer randomInt;*

    /** Creates a new instance of UserNumberBean */
    public UserNumberBean() {
        *link:http://docs.oracle.com/javase/7/docs/api/java/util/Random.html[+Random+] randc
        randomInt = new Integer(randomGR.link:http://docs.oracle.com/javase/7/docs/api/java/
        System.out.println("Duke's number: " + randomInt);*
    }

}

```

O código acima gera um número aleatório entre 0 e 10, e o número é aplicar no log do servidor.

1. Corrigir importações. Para isso, clique no indicador de dica (💡) aplique na margem esquerda do editor, em seguida, selecione a opção para importar `java.util.Random` para a classe.
2. Executar o projeto novamente (clique no botão (▶) Executar Projeto ou pressione F6; fn-F6 no Mac). Quando você executa seu projeto, o arquivo de log do servidor é aberto automaticamente na janela de saída.



Figura 8. O arquivo de log do servidor é aberto automaticamente na janela de saída. Observe que você não vê o "número do Duke: " listado na saída (como deveria ser indicado no construtor). Um objeto `UserNameBean` não foi criado porque o JSF utiliza *instanciação lenta* por default. Ou seja, os Beans em determinados escopos são criados e inicializados somente quando a aplicação precisa deles.

O Javadoc da anotação `@ ManagedBean` afirma:

Se o valor do atributo `eager()` for `true` e o valor de `managed-bean-scope` for "application", o runtime must instanciar essa classe quando uma aplicação para reclamação. Essa instanciação e o armazenamento da instância devem ocorrer antes das consequências serem processadas. Se `_eager` não estiver especificado ou for `false`, ou se `managed-bean-scope` for diferente de "aplicação", ocorrerá uma instanciação "lenta" default e o armazenamento com escopo do Bean gerenciado.

1. Como o `UserNameBean` não está escopo da sessão, implemente-o na interface `Serializable`.

```
@ManagedBean ( name = " UserNameBean " )
@SessionScoped
public class UserNameBean * implementa Serializable * {
```

Utilize o indicador de dados (💡) para importar `java.io.Serializable` para a classe.

Adicionando Propriedades

Como páginas de Facelets que você criará na próxima seção precisarão, acessando o número digitado pelo usuário e a resposta gerada. Para facilitar essa tarefa, dedicada como propriedades `userNumber` e `response` à classe.

1. Comece declarando um `Integer` denominado `userNumber`.

```
@ManagedBean ( name = " UserNumberBean " )
@SessionScoped
public class UserNumberBean implementa Serializable {

    Integer randomInt;
    * Número inteiro do usuário; *
```

1. Clique com o botão direito do mouse no editor e selecione Inserir Código (Alt-Insert; Ctrl-I no Mac). Escolha Getter e Setter.

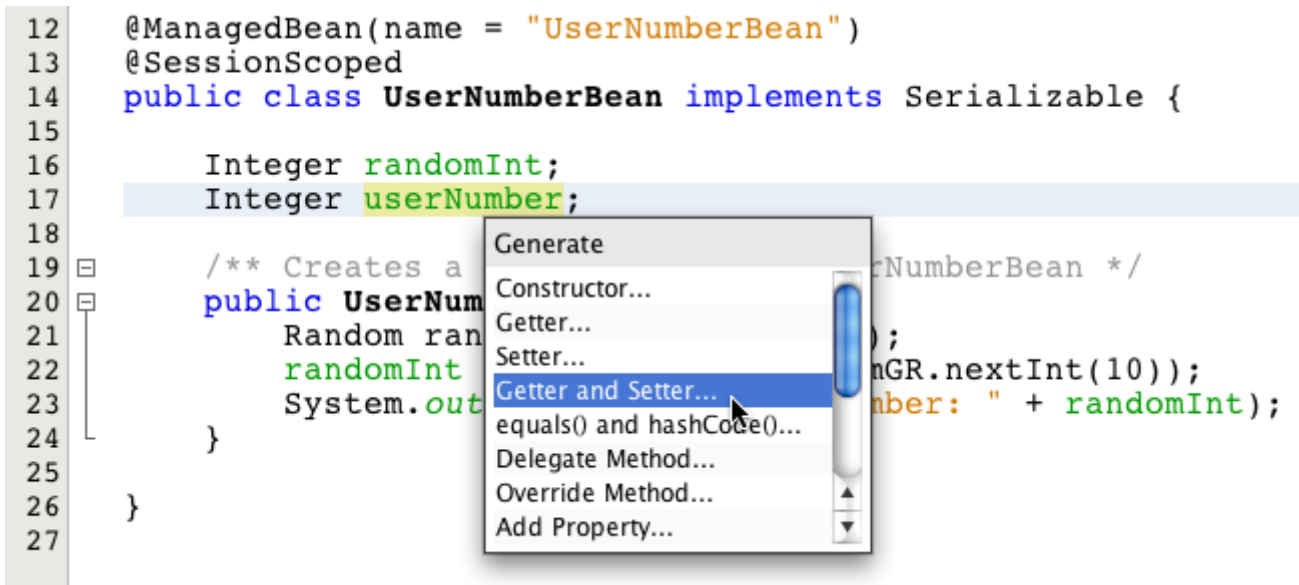


Figura 9. Utilizar o IDE para gerar métodos de acesso para propriedades

1. Selecione uma Opção `userNumber : Integer`. Clique em Gerar.

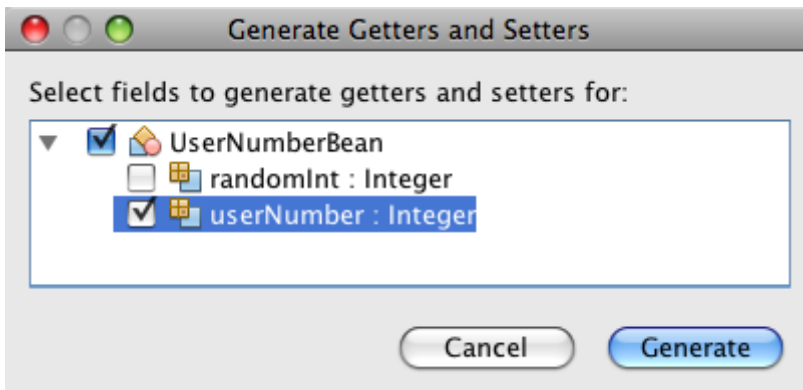


Figura 10. Utilizar o IDE para gerar métodos de acesso para propriedades

Observe que os métodos `getUserNumber()` e `setUserNumber(Integer userNumber)` são adequados à classe.

1. Crie uma propriedade `response`. Declare uma `String` denominada `response`.

```
@ManagedBean ( name = " UserNumberBean " )
@SessionScoped
public class UserNumberBean implementa Serializable {

    Integer randomInt;
```

```
Número inteiro do usuário;  
* Resposta da string ; *
```

1. Crie um método getter para `response` . (Esta aplicação não necessariamente de um setter.) Você pode usar um código gerado por janela pop-up do IDE mostrado acima, na etapa 2, para gerar o código do modelo. Neste tutorial, basta colar o método para seguir a classe.

```
public String getResponse () {  
    if ((userNumber != null) & & & (userNumber.link:http://download.oracle.com/javas  
  
        // invalida a sessão do usuário  
        FacesContext context = FacesContext.getCurrentInstance ();  
        HttpSession session = (HttpSession) context.getExternalContext (). getSession (false  
        session.invalidate ());  
  
        return "Yay! Você acertou!";  
    } outro {  
  
        return " < p > Desculpe," + userNumber + " não é. </ p > "  
            + " < p > Adivinhe novamente ... </ p > ";  
    }  
}
```

O método acima realiza duas funções: 1. Testa se o número informado pelo usuário (`userNumber`) é igual ao número aleatório gerado para a sessão (`randomInt`) e retorna uma resposta `String` apropriada. 2. Isso invalida a sessão do usuário se o usuário adivinhar o número correto (isto é, se `userNumber` for igual a `randomInt`). Isso é necessário para que um novo número seja gerado, caso o usuário queira jogar novamente.

1. Clique com o botão direito do mouse no editor e selecione Corrigir Importações (Alt-Shift-I; ⌘-Shift-I no Mac). As instruções de importação são criadas automaticamente para:

- `javax.servlet.http.HttpSession`
- `javax.faces.context.FacesContext`

Você pode pressionar Ctrl-Espaço nos itens do editor para chamar as sugestões da funcionalidade autocompletar código e o suporte da documentação. Pressione Ctrl-Espaço no `FacesContext` para exibir a descrição da classe do Javadoc.

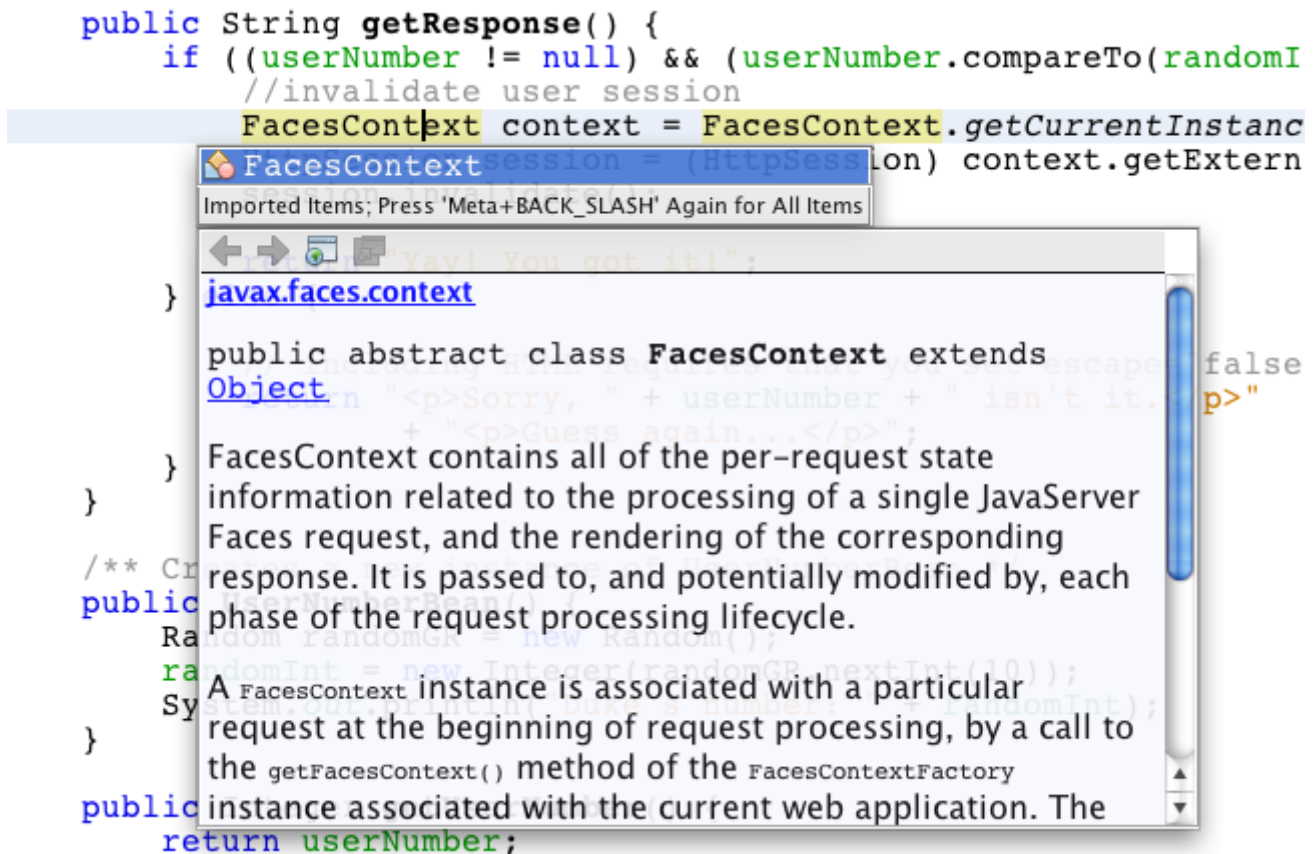



Figure 11. Pressione Ctrl-Espaço para chamar a funcionalidade autocompletar código e o suporte à documentação

Clique no ícone () do Web browser na janela da documentação para abrir o Javadoc em um Web browser externo.

Conectando Beans Gerenciados às Páginas

Uma das principais finalidades do JSF é remover a necessidade de escrever códigos clichês para gerenciar POJOs e suas interações com views da aplicação. Você viu um exemplo disso na seção anterior, na qual o JSF instanciou um objeto `UserNumberBean` quando a aplicação foi executada. Este conceito é denominado Inversão de Controle (IoC), que permite que o contêiner se responsabilize pelo gerenciamento de partes da aplicação que, do contrário, exigiriam que o desenvolvedor escrevesse códigos repetitivos.

Na seção anterior, você criou um bean gerenciado que gera um número aleatório entre 0 e 10. Você também criou duas propriedades, `userNumber` e `response`, que representam o número informado pelo usuário e a resposta a uma tentativa do usuário, respectivamente.

Nesta seção, você irá explorar como é possível utilizar `UserNumberBean` e suas propriedades em páginas Web. O JSF permite que você faça isso utilizando a sua linguagem de expressão (EL). A linguagem de expressão é utilizada para vincular os valores da propriedade aos componentes da IU do JSF contidos nas páginas Web da aplicação. Esta seção demonstra como você pode aproveitar a funcionalidade de navegação implícita do JSF 2.x para navegar entre o índice e as páginas de resposta.

O IDE oferece suporte a esta tarefa por meio das funcionalidades autocompletar código e documentação, que podem ser chamadas pressionando Ctrl-Espaço nos itens do editor.

Comece fazendo alterações em `index.xhtml` e, em seguida, em `response.xhtml`. Em ambas as páginas, substitua os elementos do form HTML por seus equivalente JSF, conforme estão definidos na biblioteca de tags HTML JSF. Em seguida, utilize a linguagem de expressão JSF para vincular os valores da propriedade aos componentes da IU selecionada.

- `index.xhtml`
- `response.xhtml`

'index.xhtml

1. Abra a página `index.xhtml` no editor. Clique duas vezes no nó `index.xhtml` da janela Projetos, ou pressione Alt-Shift-O para utilizar a caixa de diálogo Ir para Arquivo.

As páginas de índice e resposta já contêm os componentes de IU do JSF necessários para este exercício. Basta eliminar os comentários existentes e fazer comentários nos elementos HTML que estiverem sendo usados.

1. Comente no elemento do form HTML. Para isso, realce o elemento de form HTML, como na imagem a seguir, e pressione Ctrl-/ (⌘-/ no Mac).

*Observação: *para realçar, clique e arraste o elemento no editor com o mouse ou, utilizando o teclado, mantenha Shift pressionado e pressione as teclas de seta.

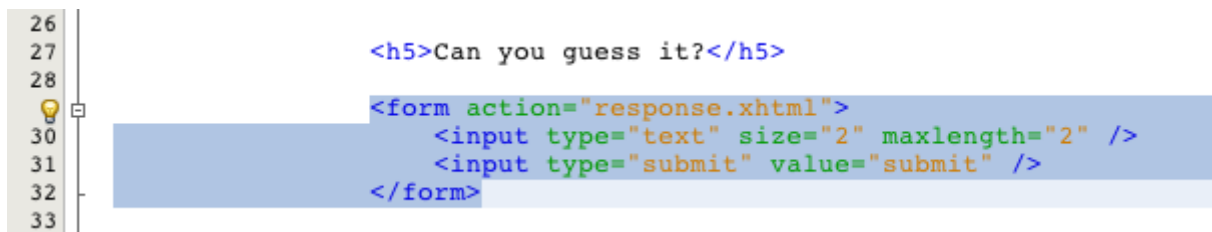


Figure 12. Realce o código, em seguida, pressione Ctrl-/ para comentá-lo. Utilize Ctrl-/ (⌘-/ no Mac) para alternar entre comentários no editor. É possível aplicar esse atalho do teclado em outros tipos de arquivo, como Java e CSS.

1. Elimine o comentário do componente do form HTML JSF. Realce o componente, conforme indicado na imagem a seguir, e pressione Ctrl-/ (⌘-/ no Mac).

Observação. Pode ser necessário pressionar Ctrl-/ duas vezes para remover os comentários do código.

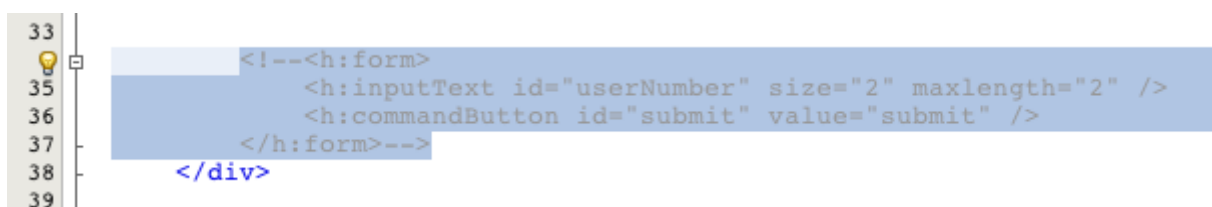


Figure 13. Realce o código comentado e, em seguida, pressione Ctrl-/ para eliminar o comentário. Após eliminar o comentário do componente de form HTML JSF, o editor indicará que as tags `<h:form>`, `<h:inputText>` e `<h:commandButton>` não foram declaradas.

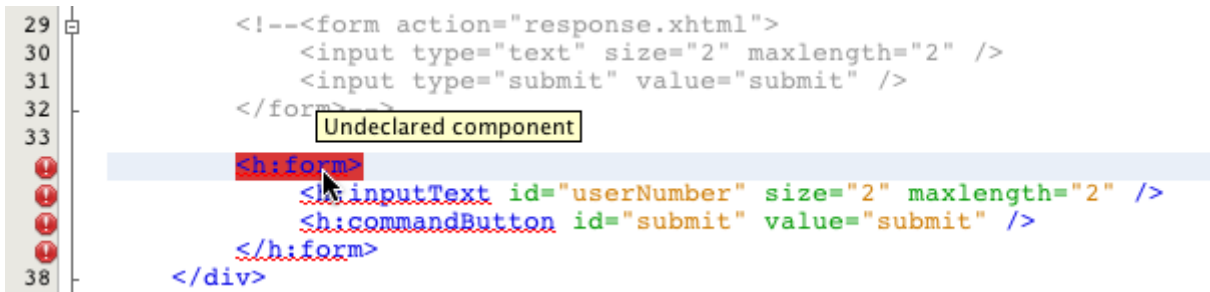


Figure 14. O editor fornecerá mensagens de erro para os componentes não declarados

1. Para declarar esses componentes, utilize a funcionalidade autocompletar código do IDE para adicionar o namespace da biblioteca de tag à tag `<html>` da página. Coloque o cursor em qualquer uma das tags não declaradas, pressione **Alt-Enter** e clique em **Inserir** para adicionar a biblioteca de tags sugerida. (Se houver várias opções, certifique-se de selecionar a tag exibida no editor antes de clicar em **Inserir**.) O namespace da biblioteca de tags HTML JSF será adicionado à tag `<html>` (mostrado em **negrito** abaixo), e os indicadores de erro desaparecerão.

Observação. Se o IDE não fornecer a opção de adicionar a biblioteca de tags, será necessário modificar manualmente o elemento `<html>` .

```
<html xmlns="http://www.w3.org/1999/xhtml"
      *xmlns:h="http://xmlns.jcp.org/jsf/html"*>
```

1. Utilize a linguagem de expressão JSF para vincular a propriedade `userNumber` do `UserNumberBean` ao componente `inputText` . O atributo `value` pode ser utilizado para especificar o valor atual do componente renderizado. Digite o código exibido em **negrito** abaixo.

```
<h:form>
  <h:inputText id="userNumber" size="2" maxlength="2" *value="#{UserNumberBean.userNumber}>
```

A linguagem de expressão JSF utiliza a sintaxe `#{}` . Dentro desses delimitadores, especifique o nome do bean gerenciado e a propriedade do Bean que deseja aplicar, separados por um ponto (`.`). Agora, quando os dados do form forem enviados ao servidor, o valor será salvo automaticamente na propriedade `userNumber` utilizando o setter da propriedade (`setUserNumber()`). Além disso, quando a página for solicitada e um valor para `userNumber` já tiver sido definido, o valor será exibido automaticamente no componente `inputText` renderizado. Para obter mais informações, consulte o Tutorial do Java EE 7: 12.1.2 Usando o EL para Beans Gerenciados de Referência.

1. Especifique o destino da solicitação chamada ao clicar no botão do form. Na versão HTML do form, você pode fazer isso utilizando o atributo `action` da tag `<form>` . Com o JSF, você pode utilizar o atributo `action` do `commandButton` . Além disso, devido à funcionalidade de navegação implícita do JSF 2.x, basta especificar apenas o nome do arquivo de destino, sem a sua extensão.

Digite o código exibido em **negrito** abaixo.


```
<h:form>
    <h:inputText id="userNumber" size="2" maxLength="2" value="#{UserNumberBean.userNumber}"
    <h:commandButton id="submit" value="submit" *action="response"* />
</h:form>
```

O runtime do JSF procura um arquivo denominado `response`. Ele supõe que a extensão do arquivo é a mesma utilizada pelo arquivo que originou a solicitação (`index*.xhtml*`) e procura o arquivo `response.xhtml` no mesmo diretório do arquivo de origem (por exemplo, `webroot`).

*Observação: *o JSF 2.x tem o objetivo de tornar as tarefas dos desenvolvedores muito mais fácil. Se você utilizou o JSF 1.2 para este projeto, você teria que declarar uma regra de navegação em um arquivo de configuração do Faces que tem a seguinte aparência:

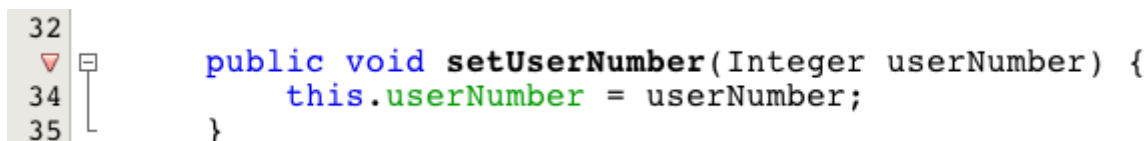
```
<navigation-rule>
    <from-view-id>/index.xhtml</from-view-id>

    <navigation-case>
        <from-outcome>response</from-outcome>
        <to-view-id>/response.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
```

As etapas de 7 a 12 a seguir são opcionais. Se quiser construir o projeto rapidamente, passe para `response.xhtml`.

1. Verifique se a expressão EL acima chama o método `setUserNumber()` quando a solicitação é processada. Para isso, utilize o depurador Java do IDE.

Altere para a classe `UserNumberBean` (Pressione Ctrl-Tab e selecione o arquivo na lista.) Defina um ponto de interrupção na assinatura do método `setUserNumber()`. É possível fazer isso clicando na margem esquerda. É exibido um indicador vermelho, mostrando que um ponto de interrupção do método foi definido.



```
32
33  public void setUserNumber(Integer userNumber) {
34      this.userNumber = userNumber;
35  }
```

Figure 15. Clique no margem esquerda do editor para definir pontos de interrupção

1. Clique no botão Depurar Projeto (🐞) na barra de ferramentas principal do IDE. A sessão de depuração é iniciada, e a página de boas-vindas do projeto é aberta no browser.

Observações.

- Talvez seja necessário confirmar a porta do servidor para depurar a aplicação.
- Se uma caixa de diálogo Depurar Projeto for exibida, selecione a opção 'Java do Servidor' default e clique em Depurar.

1. No browser, digite um número no form e clique no botão "Submeter".

- Volte ao IDE e inspecione a classe `UserNumberBean`. O depurador será interrompido no método `setUserNumber()`.

```

32  public void setUserNumber(Integer userNumber) {
33      |  this.userNumber = userNumber;
34      |
35      }

```

Figure 16. O depurador é suspenso de acordo com pontos de interrupção

- Abra a janela Variáveis do Depurador (Selecione Janela > Depuração > Variáveis, ou pressione Ctrl-Shift-1). Você verá os valores das variáveis no ponto em que o depurador foi suspenso.

Variables		
Name	Type	Value
▼ this	UserNumberBean	#6880
▼ randomInt	Integer	#6882
value	int	8
userNumber		null
response		null
▼ userNumber	Integer	#6881
value	int	4

Figure 17. Monitorar valores das variáveis utilizando a janela Variáveis do Depurador

Na imagem acima, o valor '4' foi fornecido à variável `userNumber` na assinatura `setUserNumber()`. (O número 4 foi inserido no form.) 'this' se refere ao objeto `UserNumberBean` criado para a sessão do usuário. Abaixo dele, você verá que o valor da propriedade `userNumber` é atualmente `null`.

- Na barra de ferramentas do Depurador, clique no botão Fazer Step Into (📌). O depurador executará a linha na qual ele está atualmente suspenso. A janela Variáveis se será atualiza, indicando as alterações na execução.

Variables		
Name	Type	Value
▼ this	UserNumberBean	#6880
▼ randomInt	Integer	#6882
value	int	8
▼ userNumber	Integer	#6881
value	int	4
response		null
▼ userNumber	Integer	#6881
value	int	4

Figure 18. A janela de variáveis é atualizada ao percorrer o código

A propriedade `userNumber` agora está definida com o valor inserido no form.

- Escolha Depurar > Finalizar Sessão do Depurador (Shift-F5; Shift-Fn-F5 no Mac) no menu principal para interromper o depurador.

'response.xhtml

- Abra a página `response.xhtml` no editor. Clique duas vezes no nó `response.xhtml` da janela Projetos ou pressione Alt-Shift-O para utilizar a caixa de diálogo Ir para Arquivo.

2. Comente no elemento do form HTML. Realce as tags HTML `<form>` de abertura e fechamento e o código entre elas, depois pressione Ctrl-/ (⌘-/ no Mac).

*Observação: *Para realçar, clique e arraste o mouse no editor ou, utilizando o teclado, mantenha Shift pressionado e utilize as teclas de seta.

1. Elimine o comentário do componente do form HTML JSF. Realce as tags `<h:form>` de abertura e fechamento e o código entre elas, depois pressione Ctrl-/ (⌘-/ no Mac).

Neste estágio, o código entre as tags `<body>` terão a seguinte aparência:

```
<body>
  <div id="mainContainer">

    <div id="left" class="subContainer greyBox">

      <h4>[ response here ]</h4>

      <!--<form action="index.xhtml">

        <input type="submit" id="backButton" value="Back"/>

      </form>-->

      <h:form>

        <h:commandButton id="backButton" value="Back" />

      </h:form>

    </div>

    <div id="right" class="subContainer">

      
      <!--<h:graphicImage url="/duke.png" alt="Duke waving" />-->

    </div>
  </div>
</body>
```

Após eliminar o comentário do componente de form HTML JSF, o editor indicará que as tags `<h:form>` e `<h:commandButton>` não foram declaradas.

1. Para declarar esses componentes, utilize a funcionalidade autocompletar código do IDE para adicionar o namespace da biblioteca de tag à tag `<html>` da página.

Utilize o suporte para a funcionalidade autocompletar código do editor para adicionar os namespaces JSF necessários ao arquivo. Quando você seleciona uma tag JSF ou Facelets por meio da funcionalidade autocompletar código, o namespace necessário é automaticamente adicionado ao elemento raiz do documento. Para obter mais informações, consulte Suporte JSF 2.x no NetBeans IDE.

Coloque o cursor em qualquer uma das tags não declaradas e pressione Ctrl-Espaço. As sugestões da funcionalidade autocompletar código e o suporte da documentação serão exibidos.



Figure 19. Pressione Ctrl-Espaço para chamar uma janela pop-up com as sugestões da funcionalidade autocompletar código e a documentação

Clique em Inserir. (Se houver várias opções, certifique-se de selecionar a tag exibida no editor antes de clicar em Inserir.) O namespace da biblioteca de tags HTML JSF será adicionado à tag <html> (mostrado em **negrito** abaixo), e os indicadores de erro desaparecerão.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      *xmlns:h="http://xmlns.jcp.org/jsf/html"*>
```

1. Especifique o destino da solicitação que é chamada quando o usuário clica no botão do form. Você deseja definir o botão para que um usuário retorne à página de índice ao clicar nele. Para isso, utilize, o atributo `action` do `commandButton`. Digite o código exibido em **negrito**.

```
<h:form>

    <h:commandButton id="backButton" value="Back" *action="index"* />


</h:form>
```

Observação: *Ao digitar `action="index"`, você estará confiando na funcionalidade de navegação implícita do JSF. Quando um usuário clica no botão do form, o runtime do JSF procura um arquivo denominado `índice`. Ele supõe que a extensão do arquivo é a mesma extensão utilizada pelo arquivo que originou a solicitação (`response .xhtml*`) e procura o arquivo `index.xhtml` no mesmo diretório do arquivo de origem (por exemplo, `webroot`).

1. Substitua o texto estático "[response here]" pelo valor da propriedade `response` do `UserNumberBean`. Para isso, utilize a linguagem de expressão JSF. Digite (em **negrito**):

```
<div id="left" class="subContainer greyBox">
```

```
    < h4 > * < h: outputText  value = " # {UserNumberBean.response} " /> * </ h4 >
```

1. Executar o projeto (clique no botão () Executar Projeto ou pressione F6; fn-F6 no Mac). Quando uma página de boas-vindas for exibida no navegador, informe um número e clique em `submiter` . Você verá uma página de resposta com uma semelhante à seguinte (contanto que você não adivinha o número correto):

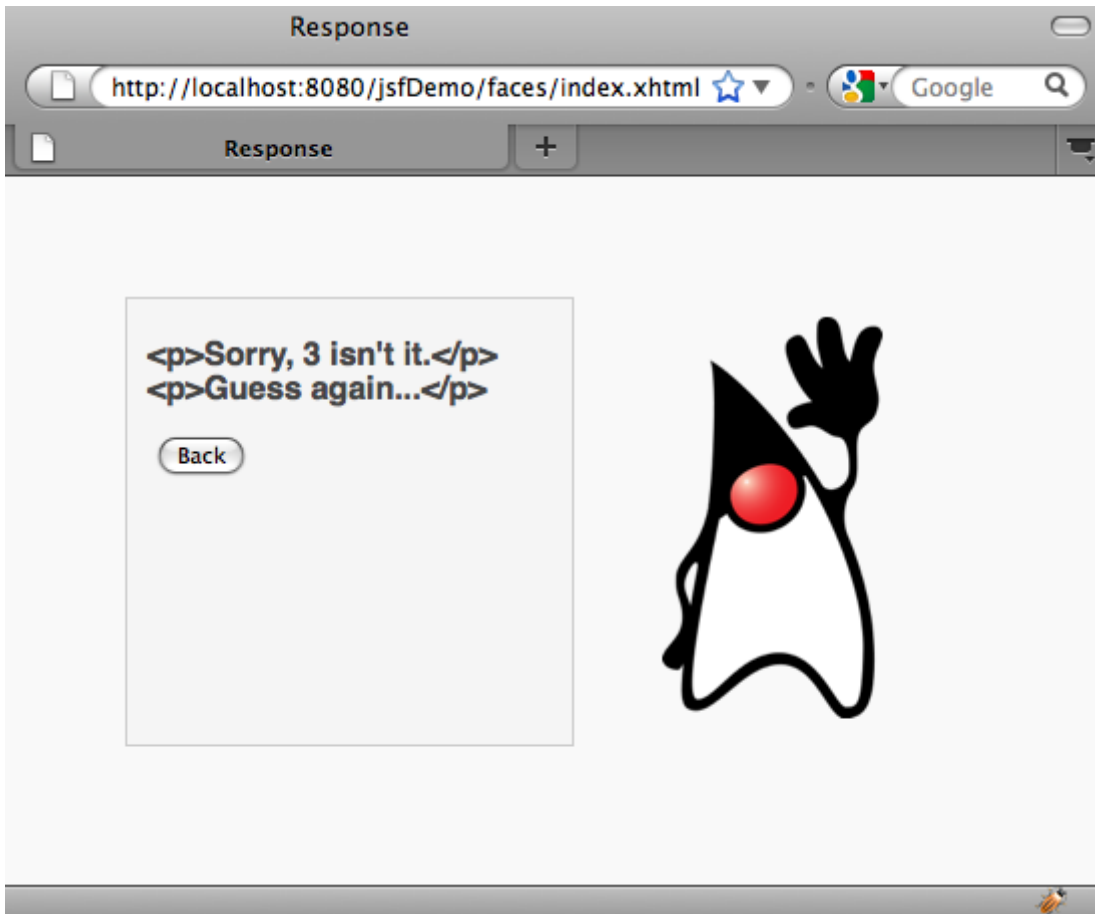


Figura 20. Exibir o status atual do projeto em um navegador
Há duas coisas erradas sem status atual da página de resposta:

1. As tags html `<p>` são exibidas na mensagem de resposta.
2. O botão Voltar não está aplicando no lugar correto. (Compare-a à versão original .)

As duas etapas a seguir corrigirão estes pontos, respectivamente.

1. Defina o atributo `escape` da tag `<h:outputText>` como `false` . Coloque o cursor entre `outputText` e `value` , insira um espaço e pressione Ctrl-Espaço para chamar a funcionalidade de autocompletar código. Papel para baixo para selecionar o atributo `escape` e inspecione a documentos.



Figura 21. Pressione Ctrl-Espaço para exibir os valores possíveis do atributo e os documentos. Conforme indicado pela documentação, o valor `escape` está definido como `true` padrão. Isso significa que os caracteres que normalmente são incluídos na análise como HTML incluídos na string, conforme ilustrado acima. Definir o valor como `false` permite aos caracteres que podem ser relacionados a analisar como HTML sejam renderizados como tal.

Clique em Inserir e digite `false` como valor.

```
< h4 > < h : outputText * escape = " false " * value = " #{UserNumberBean.response} " /> </
```

1. Defina, o atributo `prependId` da tag `<h:form>` como `false`. Coloque o cursor logo depois de 'm' em `<h:form>`, insira um espaço, em seguida, pressione Ctrl-Espaço para chamar a funcionalidade autocompletar código. Papel para baixo para selecionar o atributo `prependId` e inspecione a documentação. Clique em Inserir e digite `false` como valor.

```
< h : form * prependId = " false " * >
```

O JSF aplica IDs internos para manter o controle dos componentes da IU. No exemplo atual, se você inspecionar o código-fonte da página renderizada, você verá algo semelhante ao seguinte:

```
<form id="j_idt5" name="j_idt5" method="post" action="/jsfDemo/faces/response.xhtml" enctype="text/xml" >
  <input type="hidden" name="j_idt5" value="j_idt5" />
  <input *id="j_idt5:backButton" type="submit" name="j_idt5:backButton" value="Back" />
  <input type="hidden" name="javax.faces.ViewState" id="javax.faces.ViewState" value="7464" />
</form>
```

O ID do elemento do form é `j_idt5` e esse ID é *precedido* pelo ID do botão Voltar incluído no form (mostrado em **negrito** acima). Como o botão Voltar depende da regra de estilo do `#backButton` (definido em `stylesheet.css`), essa regra se torna um empecilho quando o ID do JSF é inserido como prefixo. Isso pode ser evitado ao definir `prependId` como `false`.

1. Execute o projeto novamente (clique no botão (▶) Executar Projeto ou pressione F6; fn-F6 no Mac). Insira um número na página de boas-vindas e clique em Submeter. A página de resposta agora exibe a mensagem de resposta sem as tags `<p>` e o botão Voltar está no lugar correto.

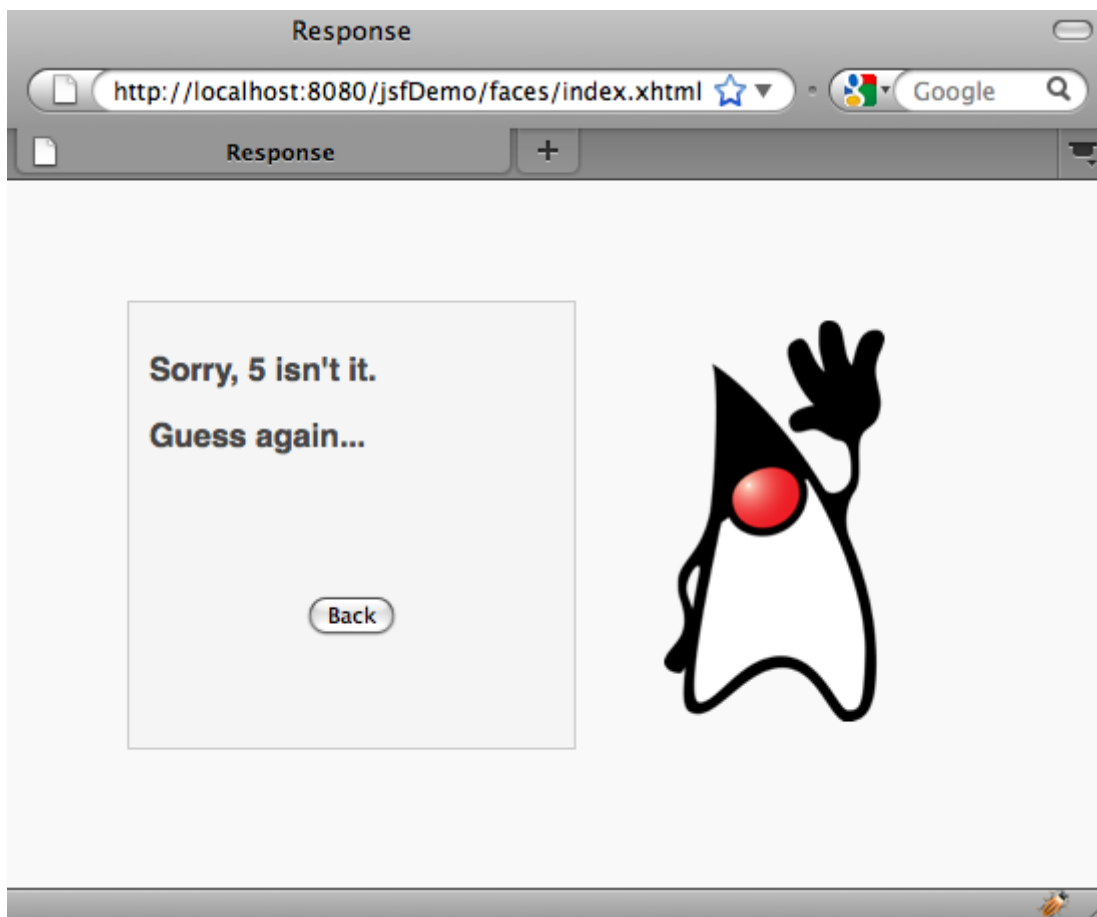


Figure 22. Exibir o status atual do projeto em um browser

1. Clique no botão Voltar. Como o valor atual da propriedade `userNumber` do `UserNumberBean` está vinculado ao componente JSF `inputText`, o número inserido anteriormente agora será exibido no campo de texto.
2. Inspecione o log do servidor na janela de Saída do IDE (Ctrl-4; ⌘-4 no Mac) para determinar qual é o suposto número correto.

Se, por alguma razão, você não puder ver o log do servidor, você poderá abri-lo alternando para a janela Serviços (Ctrl-5; ⌘-5 no Mac) e expandindo o nó Servidores. Em seguida, clique com o botão direito do mouse no GlassFish Server no qual o projeto está implantado e selecione Exibir Log do Servidor. Se você não conseguir ver o número no log do servidor, tente construir novamente a aplicação clicando com o botão direito do mouse no nó do projeto e selecionando Limpar e Construir Projeto.

1. Digite o número correto e clique em Submeter. A aplicação irá comparar a sua entrada com o número atualmente salvo e exibirá a mensagem apropriada.

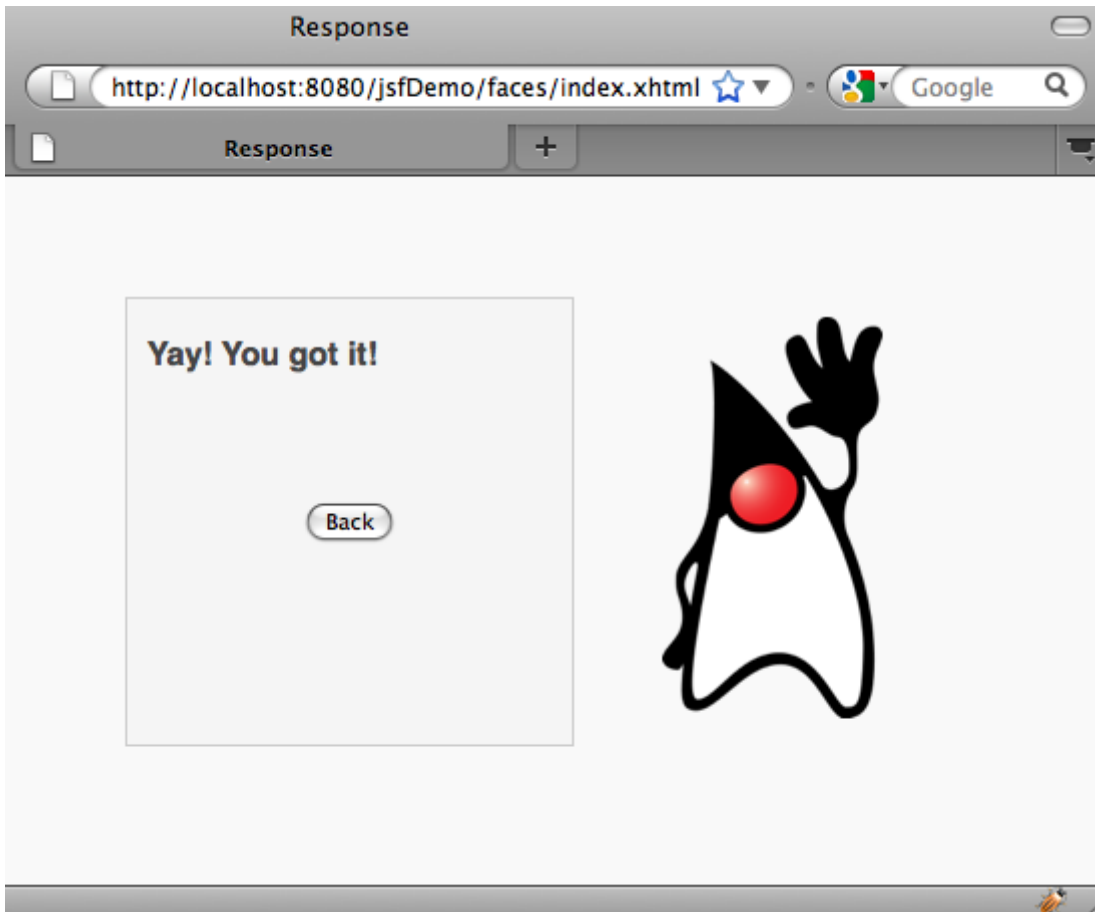


Figure 23. A resposta correta será exibida ao inserir o número correspondente

1. Clique novamente no botão Voltar. Observe que o número inserido anteriormente não será mais exibido no campo de texto. Lembre-se de que o método `getResponse()` do `UserNumberBean` invalida a sessão atual do usuário quando o número correto é descoberto.

Aplicando um Modelo de Facelets

O Facelets se tornou a tecnologia padrão de exibição para o JSF 2.x. O Facelets é um framework de modelos leve que suporta todos os componentes de IU do JSF e é usado para compilar e renderizar a árvore de componentes do JSF para views da aplicação. Também oferece suporte ao desenvolvimento quando ocorrem erros de EL, permitindo que você inspecione o rastreamento de pilha, a árvore de componentes e as variáveis com escopo.

Embora você talvez não tenha percebido, os arquivos `index.xhtml` e `response.xhtml` com os quais você está trabalhando até o momento no tutorial são páginas de Facelets. As páginas de Facelets utilizam a extensão `.xhtml` e, desde que você esteja trabalhando em um projeto JSF 2.x (As bibliotecas JSF 2.x incluem arquivos JAR Facelets.), as views poderão renderizar apropriadamente a árvore de componentes JSF.

O objetivo desta seção é familiarizar você com modelos de Facelets. Em projetos que contêm várias views, geralmente é mais vantajoso aplicar um arquivo de modelo que defina a estrutura e a aparência das diversas views. Quando você atende às solicitações, a aplicação insere dinamicamente o conteúdo preparado no arquivo de modelo e envia o resultado de volta ao cliente. Embora esse projeto contenha somente duas views (a página de boas-vindas e página de resposta), é fácil ver que elas contêm uma grande quantidade de conteúdo duplicado. É possível fatorar esse conteúdo duplicado em um modelo de Facelets e criar arquivos do cliente de modelo para manipular o conteúdo específico das páginas de boas-vindas e resposta.

O IDE oferece um assistente de modelo de Facelets para a criação de modelos de Facelets e um assistente de Cliente de modelo de Facelets para a criação de arquivos dependentes de um modelo. Esta seção utiliza esses assistentes.

Observação: O IDE também oferece um assistente de Página JSF que permite criar páginas de Facelets individuais para seu projeto. Para obter mais informações, consulte Suporte JSF 2.x no NetBeans IDE .

- Criando o Arquivo de Modelo de Facelets
- Criando Arquivos Clientes de Modelo

’ Criando o Arquivo de Modelo de Facelets

1. Crie um arquivo de modelo de Facelets. Pressione Ctrl-N (& # 8984-N no Mac) para abrir o assistente Arquivo. Selecione a categoria JavaServer Faces e, em seguida, o Modelo de Facelets. Clique em Próximo.
2. Digite `template` como nome do arquivo.
3. Escolha um dos oito estilos de layout e clique em Finalizar. (Você utiliza uma folha de estilo existente, portanto, não importa qual estilo de layout você escolherá.)

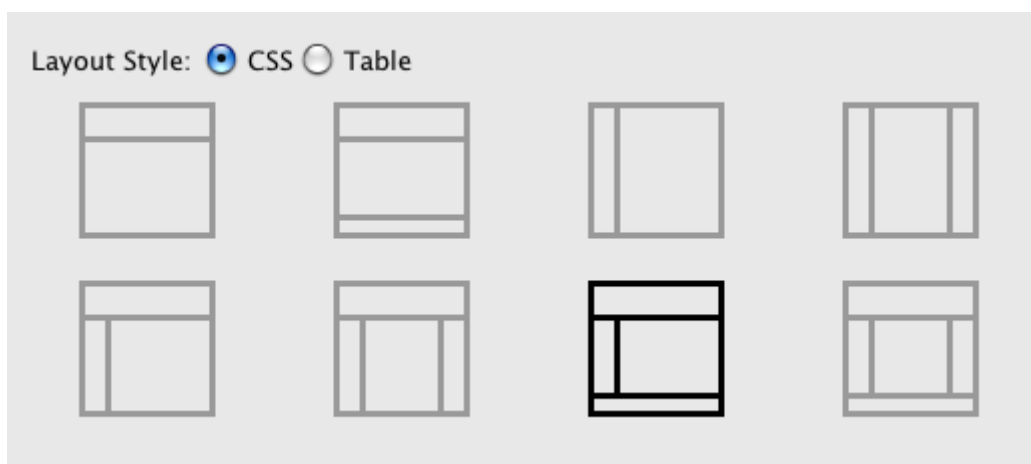


Figura 24. O assistente de Modelo de Facelets permite selecionar entre estilos de layout comuns

O assistente gera o arquivo `template.xhtml` e as folhas de estilo relacionadas com a seleção feita, a seguir, coloca-as em uma pasta `resources > css` dentro da raiz da Web do projeto.

Depois de concluir o assistente, o arquivo de modelo é aberto no editor. Para exibir o modelo em um navegador, clique com o botão direito do mouse no Editor e selecione Exibir.

1. Examine a marcação do arquivo de modelo. Observe os pontos a seguir:

- o Uma biblioteca de tags `facelets` é declarada na tag `<html>` da página. A biblioteca de tags possui o prefixo `ui`.

```
< html xmlns = " http://www.w3.org/1999/xhtml "
    * xmlns : ui = " http://xmlns.jcp.org/jsf/facelets " *
    xmlns : h = " http: // xmlns .jcp.org / jsf / html " >
```

- A página de Facelets utiliza como tags `<h:head>` e `<h:body>` em vez das tags `<head>` e `<body>`. Quando você utiliza essas tags, o Facelets pode construir uma árvore de componentes que abrange toda a página.
- A página faz referência às folhas de estilo que também foram confirmados ao concluir o assistente.

```
< h : head >
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    *<link href="./resources/css/default.css" rel="stylesheet" type="text/css" />*
    *<link href="./resources/css/cssLayout.css" rel="stylesheet" type="text/css" />*
    <title>Facelets Template</title>
</h:head>
```

- As tags `<ui:insert>` são utilizadas no corpo da página em todos os compartimentos associados ao estilo de layout que você escolheu. Cada tag `<ui:insert>` possui um atributo `name` que identifica o compartimento. Por exemplo:

```
<div id="top">
    *<ui:insert name="top">Top</ui:insert>*
</div>
```

1. Examine novamente as páginas de boas-vindas e de resposta. O único conteúdo que é alterado nas duas páginas é o título e o texto contido no quadro cinza. O modelo, portanto, pode fornecer todo o conteúdo restante.

2. Substitua todo o conteúdo de seu arquivo de modelo pelo conteúdo abaixo.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:h="http://xmlns.jcp.org/jsf/html">
```



```

<h:head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <link href="css/stylesheet.css" rel="stylesheet" type="text/css" />

  <title><ui:insert name="title">Facelets Template</ui:insert></title>
</h:head>

<h:body>

  <div id="left">
    <ui:insert name="box">Box Content Here</ui:insert>
  </div>

</h:body>

</html>

```

O código acima implementa as seguintes alterações: * O arquivo `stylesheet.css` do projeto substitui as referências da folha de estilo do modelo criadas pelo assistente. * Todas as tags `<ui:insert>` (e suas tags `<div>`) foram removidas, exceto uma denominada `box`. * Um par de tags `<ui:insert>` foi colocado no título da página e foi denominado `title`.

1. Copie o código relevante do arquivo `index.xhtml` ou `response.xhtml` para o modelo. Adicione o conteúdo mostrado em **negrito** abaixo das tags `<h:body>` do arquivo de modelo.

```

<h:body>
  *<div id="mainContainer">*
    <div id="left" *class="subContainer greyBox"*>
      <ui:insert name="box">Box Content Here</ui:insert>
    </div>
    *<div id="right" class="subContainer">
      
    </div>
  </div>*
</h:body>

```

1. Execute o projeto. Quando a página de boas-vindas abrir no browser, modifique o URL para o seguinte:

`http://localhost:8080/jsfDemo/faces/template.xhtml`

O arquivo de modelo é exibido da seguinte forma:

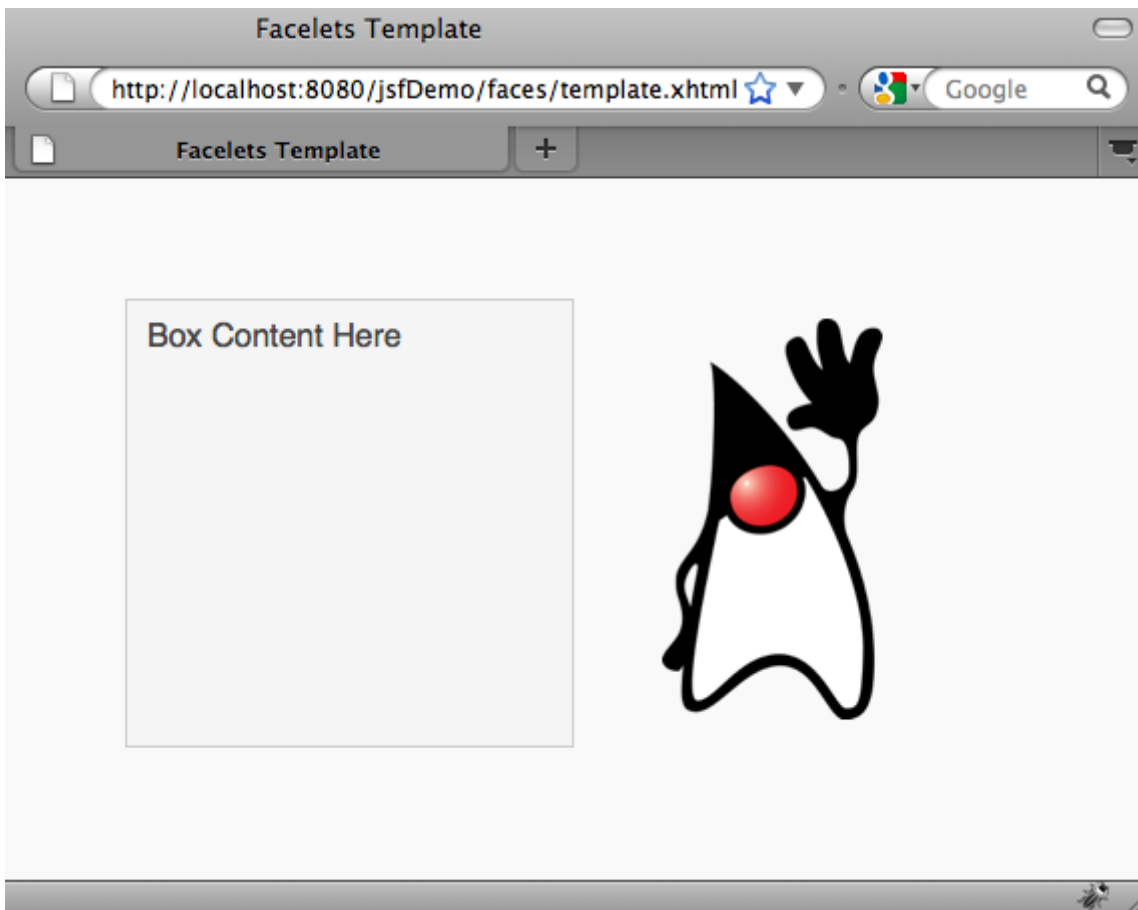


Figure 25. Exibir o modelo de Facelets em um browser

O projeto agora contém um arquivo de modelo que fornece a aparência e a estrutura de todas as views. Agora é possível criar arquivos de clientes que chamem o modelo.

’ Criando Arquivos de Clientes de Modelo

Crie os arquivos de clientes de modelo das páginas de boas-vindas e resposta. Nomeie o arquivo de cliente de modelo da página de boas-vindas `greeting.xhtml`. Para a página de resposta, o arquivo será `response.xhtml`.

’ `greeting.xhtml`

1. Pressione Ctrl-N (⌘-N no Mac) para abrir o assistente de Novo Arquivo. Selecione a categoria JavaServer Faces e, em seguida, Cliente de Modelo de Facelets. Clique em Próximo.
2. Digite `greeting` como nome do arquivo.
3. Clique no botão Procurar ao lado do campo Modelo e, em seguida, utilize a caixa de diálogo exibida para navegar até o arquivo `template.xhtml` criado na seção anterior.

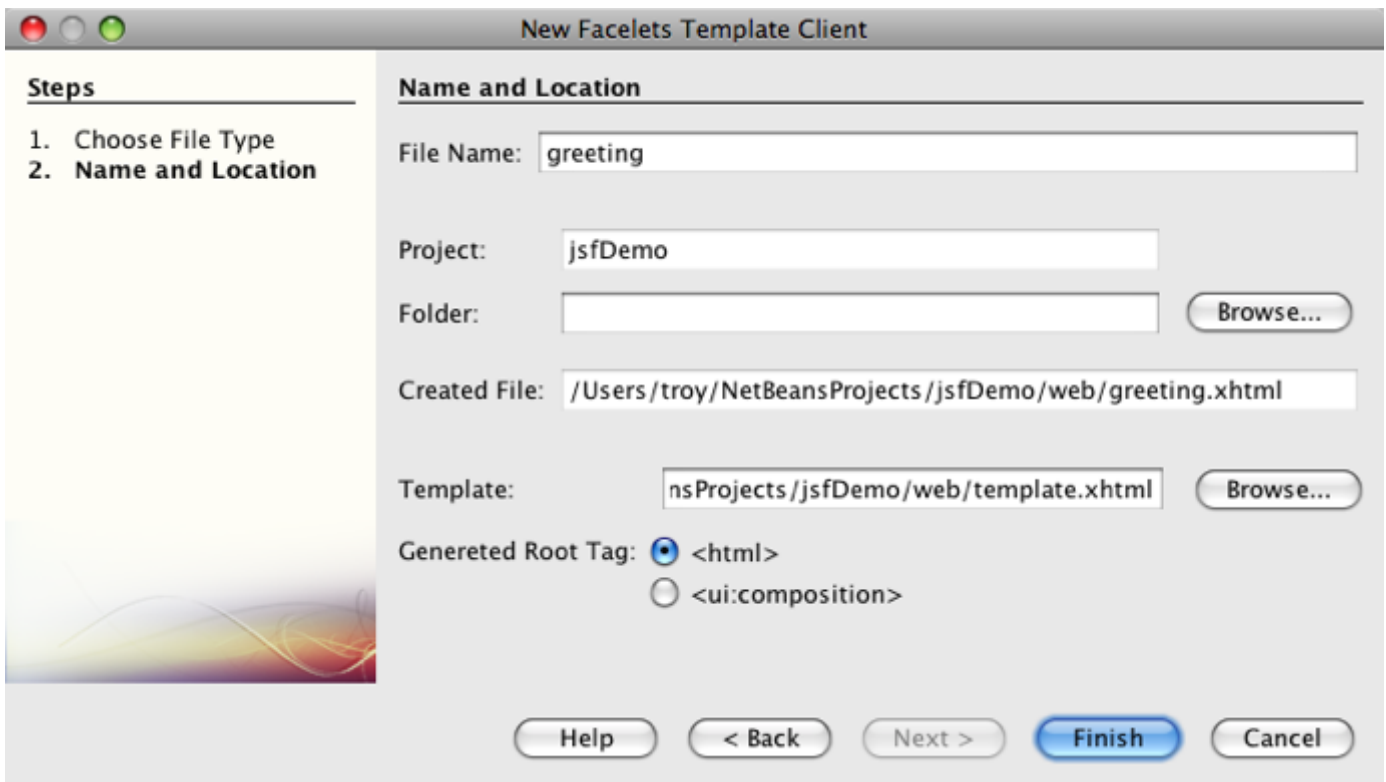


Figure 26. Assistente de Cliente de Modelo de Facelets

1. Clique em Finalizar. O novo arquivo de cliente de modelo `greeting.xhtml` é gerado e exibido no editor.
2. Examine a marcação. Observe o conteúdo realçado em **negrito**.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

  <body>

    <ui:composition *template="./template.xhtml"*>

      <ui:define *name="title"*>
        title
      </ui:define>

      <ui:define *name="box"*>
        box
      </ui:define>

    </ui:composition>

  </body>
</html>
```

O arquivo de cliente de modelo faz referência a um modelo utilizando o atributo `template` da tag `<ui:composition>`. Como o modelo contém tags `<ui:insert>` de `title` e `box`, esse cliente de modelo contém as tags `<ui:define>` desses dois nomes. O conteúdo especificado entre as tags `<ui:define>` será o conteúdo inserido no modelo entre as tags `<ui:insert>` do nome correspondente.

1. Especifique `greeting` como o título do arquivo. Faça a seguinte alteração em **negrito** .

```
< ui : define  name = " title " >
    *Saudações*
</ ui : define >
```

1. Alterne para o arquivo `index.xhtml` (pressione Ctrl-Tab) e cópia do conteúdo que normalmente aparece no quadro cinza aplicável na página renderizada. Em seguida, volte ao `greeting.xhtml` e cole-o no arquivo de cliente de modelo. (Alterações em **negrito** .)

```
< ui : define  name = " box " >
    * < h4 > Olá, meu nome é Duke! </ h4 >

    < h5 > Estou pensando em um número

        < Br />
        entre
        < span  class = " highlight " > 0 </ span > e
        < span  class = " highlight " > 10 </ span >. </ h5 >

    < h5 > Você consegue adivinhar? </ h5 >

    < h : form >
        < h : inputText  size = " 2 "  maxlength = " 2 "  value = " # {UserNumberBean.userNu
        < h : commandButton  id = " submit "  value = " submit "  action = " response " />
    </ h : form > *
</ ui : define >
```

1. Declara uma biblioteca de tags HTML JSF do arquivo. Coloque o cursor em qualquer um das tags sinalizadas com um erro (qualquer tag com o prefixo `h'`) e pressione Ctrl-Espaço. Em seguida, selecione a tag na lista de sugestões da funcionalidade autocompletar código. O namespace da biblioteca de tags será adicionado à tag `<html>` do arquivo (número em **negrito** abaixo) e os indicadores de erro desaparecerão.

```
< html xmlns = " http://www.w3.org/1999/xhtml "
    xmlns : ui = " http://xmlns.jcp.org/jsf/facelets "
    * xmlns : h = " http:// xmlns. jcp.org/jsf/html " * >
```

Se você colocar o cursor depois de 'm' no `<h:form>` e pressionar Ctrl-Espaço, o namespace será automaticamente adicionado ao arquivo. Se somente uma opção ativa estiver disponível ao pressionar Ctrl-Espaço, ela será imediatamente aplicada ao arquivo. As bibliotecas de tags JSF são automaticamente declaradas ao chamar uma funcionalidade autocompletar código nas tags.

' response.xhtml

Como o projeto já contém um arquivo com o nome `response.xhtml` , e visto que você já sabe qual é a aparência do arquivo de cliente de modelo, modifique o arquivo `response.xhtml` para que se torne o arquivo de cliente de modelo. (Neste tutorial, basta copiar e colar o código fornecido.)

1. Abra `response.xhtml` no editor. (Se já estiver aberto, pressione Ctrl-Tab e selecione-o.) Substitua o conteúdo de todo o arquivo pelo código abaixo.

```
<? xml version = ' 1.0 ' encoding = ' UTF-8 ' ?>
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR
< html xmlns = " http://www.w3.org/1999/xhtml "
    xmlns : ui = " http://xmlns.jcp.org/jsf/facelets "
    xmlns : h = " http://xmlns.jcp.org/jsf/html " >

< corpo >

    < ui : composition template = " ./template.xhtml " >

        < ui : define name = " title " >
            Resposta
        </ ui : define >

        < ui : define name = " box " >
            < h4 > < h : outputText escape = " false " value = " # {UserNumberBean.res

                < h : form prependId = " false " >

                    < h : commandButton id = " backButton " value = " Back " action = " g

                </ h : formulário >
            </ ui : define >

        </ ui : composição >

    </ body >
</ html >
```

Observe que o arquivo é idêntico a `greeting.xhtml`, exceto pelo conteúdo especificado entre as tags `<ui:define>` de `title` e `box`.

1. No descritor de implantação `web.xml` do projeto, modifique a entrada do arquivo de boas-vindas para que `greeting.xhtml` seja a página exibida quando a aplicação for executada.

Na janela Projetos, clique duas vezes em Arquivos de Configuração > `web.xml` para abri-lo no editor. Na guia Páginas, altere o campo Arquivos de Boas-Vindas para `faces/greeting.xhtml`.



Figure 27. Alterar a entrada de Arquivos de Boas-Vindas no descritor de implantação

1. Execute o projeto para ver como ele é em um browser. Pressione F6 (fn-F6 no Mac) ou clique no botão (▶) Executar Projeto na barra de ferramentas principal. O projeto será implantado no GlassFish Server e aberto em um browser.

Quando você utiliza o modelo de Facelets e os arquivos de clientes do modelo, uma aplicação se comportará exatamente da mesma forma que antes. Fatorando o código duplicado nas páginas de boas-vindas e de resposta da aplicação, você consegue reduzir o tamanho da aplicação e eliminar a possibilidade de escrever mais códigos duplicados, caso mais páginas sejam adicionadas posteriormente. Isso pode tornar o desenvolvimento mais fácil e eficiente ao trabalhar em grandes projetos.

Enviar Feedback neste Tutorial

’ Consulte Também

Para obter mais informações sobre o JSF 2.x, consulte os recursos a seguir:

’ Tutoriais e Artigos NetBeans

- Suporte a JSF 2.x no NetBeans IDE
- Gerando uma Aplicação CRUD JavaServer Faces 2.x Usando um Banco de Dados
- Scrum Toys: A Aplicação de Amostra Completa do JSF 2.0
- Conceitos Básicos sobre Aplicações do Java EE
- Trilha do Aprendizado do Java EE e Java Web

’ Recursos Externos

- Tecnologia JavaServer Faces (homepage Oficial)
- Especificação do JSR 314 para o JavaServer Faces 2.0
- O Tutorial do Java EE 7, Capítulo 12: Desenvolvendo a Tecnologia JavaServer Faces
- Projeto Mojarra GlassFish (Implementação oficial de referência do JSF 2.x)

- Fóruns de Discussão OTN: JavaServer Faces
- Central do JSF

’ Blogs

- Ed Burns
- [Jim Driscoll](#)