Minha primeira aplicação utilizando Java Server Faces + Glassfish

Este tutorial apresenta de forma simples, como iniciar no desenvolvimento com o framework Java Server faces 1.2, utilizando-se o servidor glassfish como servidor de aplicação e a IDE eclipse.

Requisitos deste tutorial:

Conhecer o básico sobre projetos Java EE (Projetos Web, Servlets, JSP).

Introdução

Nós desenvolvedores de software, estamos assistindo a rápida evolução da internet e seus sites. Quem se lembra dos CGI e logo depois os Servlets /JSPs, Struts e agora o Java Server Faces que tem disponibilizado ao desenvolvedor uma forma de desenvolver uma aplicação web 2 que provê interação do usuário às aplicações. O fato é que cada vez mais as aplicações web fornecem interação ao usuário, e, para prover padrões de mercado e de desenvolvimento surgiu o framework Java Server Faces.

O que é o Java Server Faces?

O Java Server Faces é um framework MVC que provê componentes UI, mecanismos de ações, validadores e toda a estrutura necessária para que seja criada uma aplicação Web complexa.

Por que utilizar o Java Server Faces?

O framework Java Server Faces foi desenvolvido na tecnologia Java e utiliza-se dos padrões de projetos atuais, além de possuir uma poderosa arquitetura para o desenvolvimento de aplicações de grande porte com qualidade, e fornece recursos e componentes web 2 tornando assim possível a criação de produtos interativos.

Evolução do framework

Como sabemos, uma mudança radical não surge da noite para o dia, e com o Java Server Faces não seria diferente. As primeiras versões das especificações do Java Server Faces possuíam muitos bugs, mas já dava para ter uma visão do potencial que o framework disponibilizaria para o desenvolvimento de aplicações JEE.

Foram tantas as marteladas no dedo, horas na frente do computador buscando uma solução para problemas que pareciam tão simples. Mas Hoje, podemos dizer que temos uma boa versão do framework, e que podemos sim desenvolver com produtividade uma aplicação que rode 24x7 com compatibilidade entre navegadores e servidores de aplicação.

Objetivo

Para mostrar que é possível e introduzir esta tecnologia aos desenvolvedores que estão curiosos para conhecê-la, iremos desenvolver uma aplicação de login.

Requisitos

Para desenvolvermos a aplicação deste tutorial, devemos realizar o download da Api JBoss RichFaces, o Glassfish o Eclipse Ganymede.

Richfaces:

http://www.jboss.org/jbossrichfaces/downloads/

Eclipse:

http://www.eclipse.org/downloads/
Commons Beanutils, Collections, Logging e Digester
http://commons.apache.org
Glassfish:
https://glassfish.dev.java.net/

Obs: Para realizar a instalação do glassfish é necessário que o apache ant esteja instalado na máquina. Segue abaixo o link do tutorial de como instalar o servidor de aplicação do glassfish(Em Inglês):
http://blogs.sun.com/bharath/entry/glassfish_installation_process_the_beginner

Mãos na Massa

Nossa aplicação utilizará a API richfaces, que provê componentes UI ao framework JSF e neste projeto, será realizado o armazenamento dos dados, porém na memória volátil e não em um banco de dados.

Início

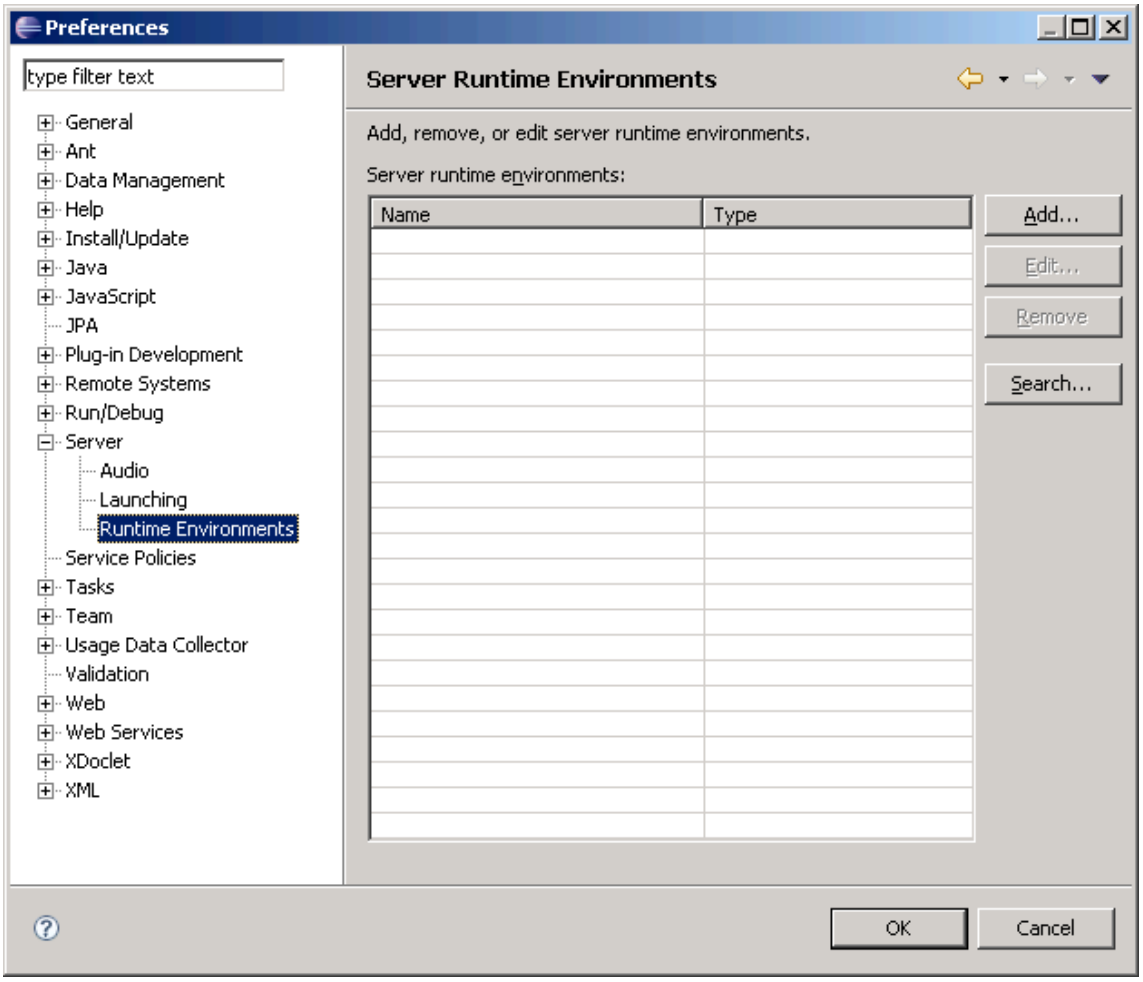
Após configurar o seu eclipse com o glassfish e o Java 5, iremos iniciar o nosso Projeto.

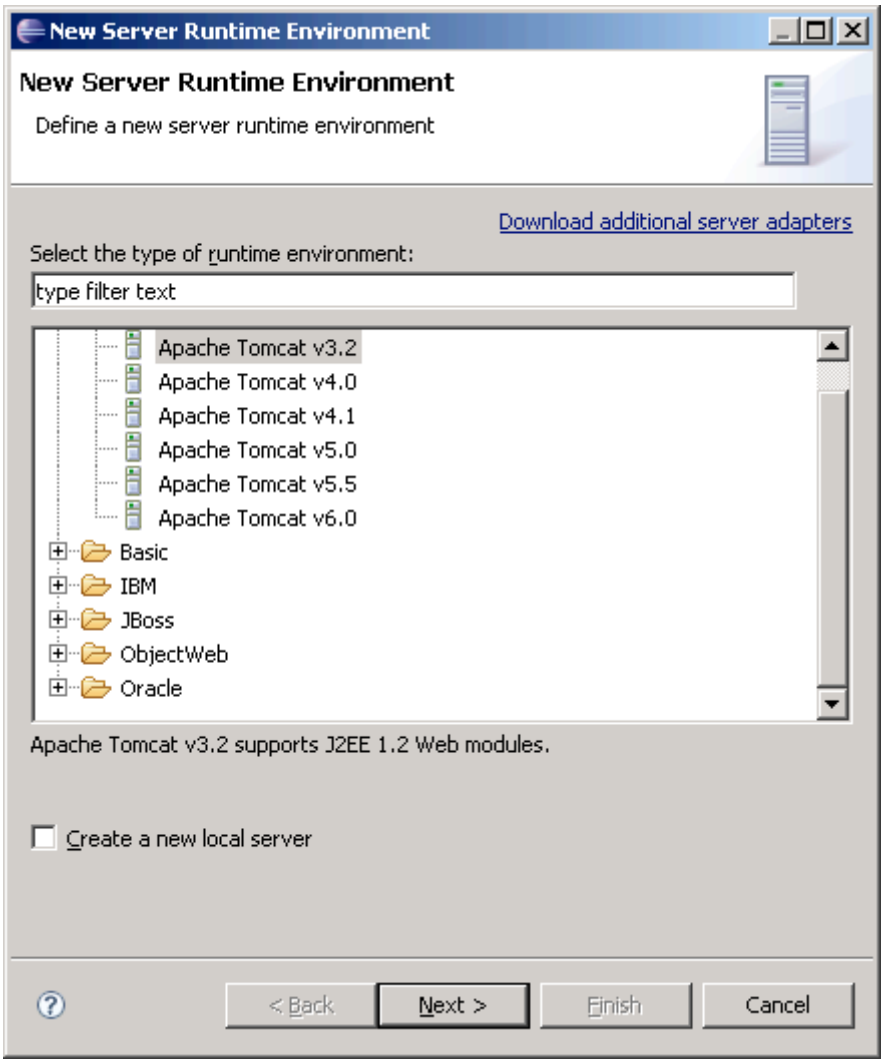
Obs: O glassfish já acompanha a implementação da Sun do Java Server Faces, portanto não será necessário o download do pacote no site da Sun.

Instalando o suporte ao glassfish ao Eclipse

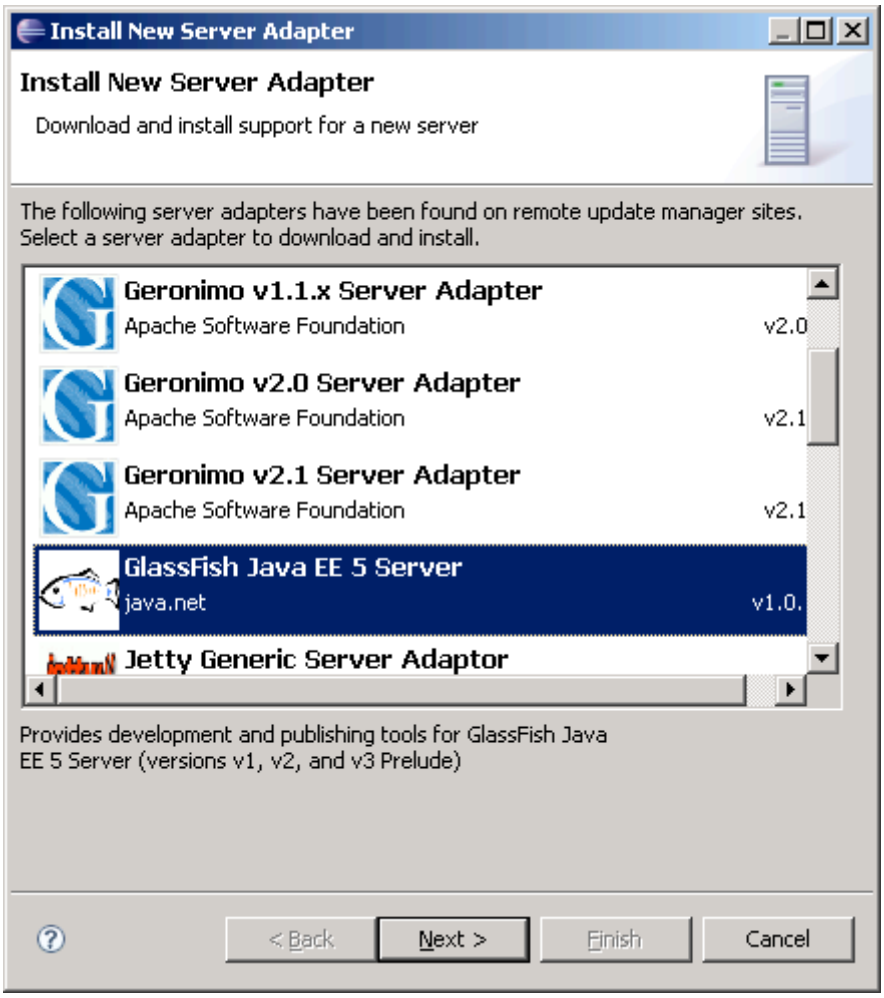
Primeiro será preciso instalar o suporte ao servidor Glassfish ao eclipse, pois por default na distribuição do eclipse, ele não vem com o suporte.

ACESSAR O MENU WINDOW -> PREFERENCES DO ECLIPSE
NO ITEM SERVER > RUNTIME ENVIORNMENTS SELECIONAR O BOTÃO ADD





SELECIONAR A OPÇÃO DOWNLOAD ADDITIONAL SERVER ADAPTERS



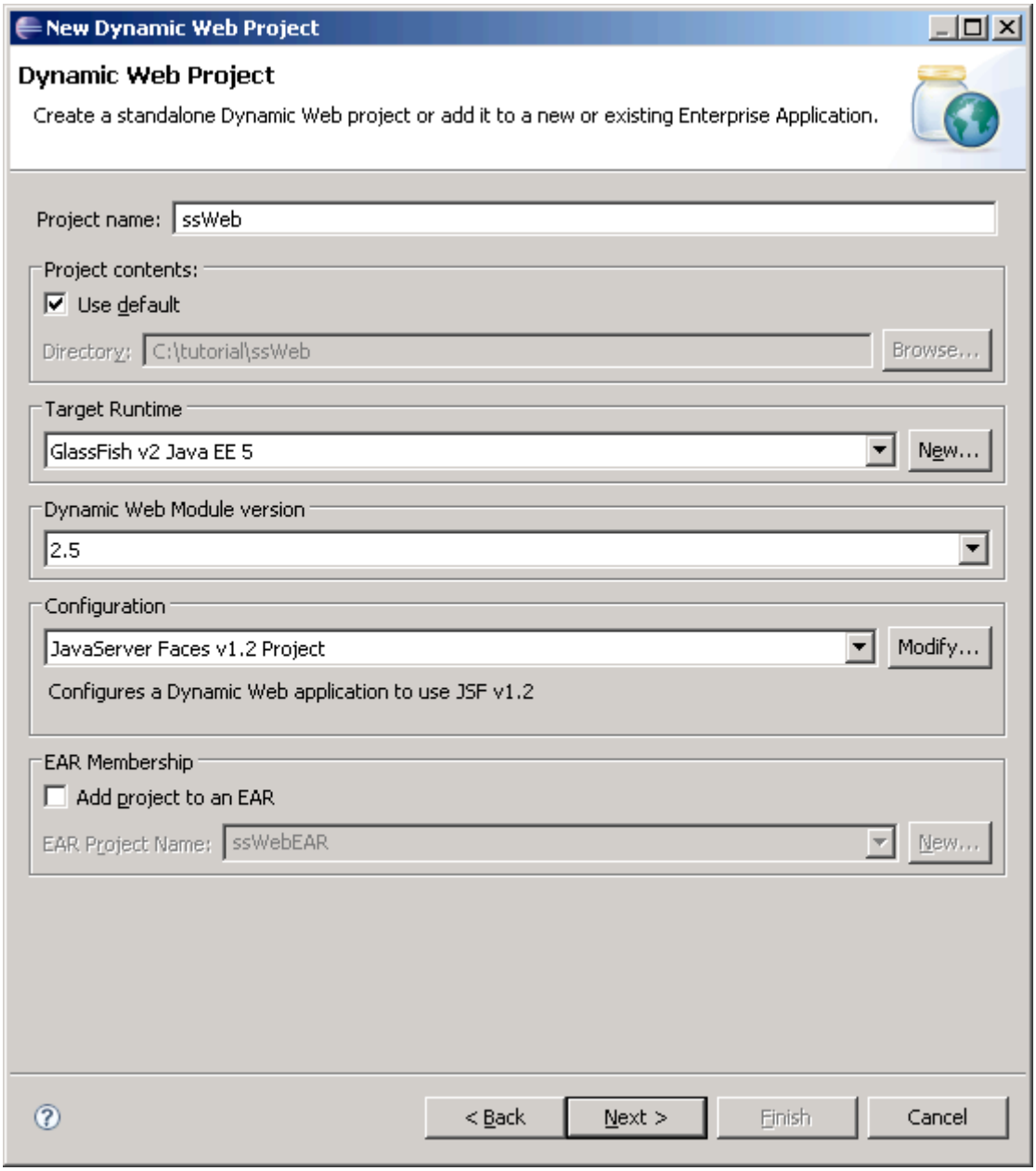
O SISTEMA IRÁ CARREGAR A CONFIGURAÇÃO DO SERVIDOR GLASSFISH. SELECIONÁLA E SEGUIR OS PASSOS DO BOTÃO NEXT.

SERÁ NECESSÁRIA A REINICIALIZAÇÃO DO ECLIPSE.

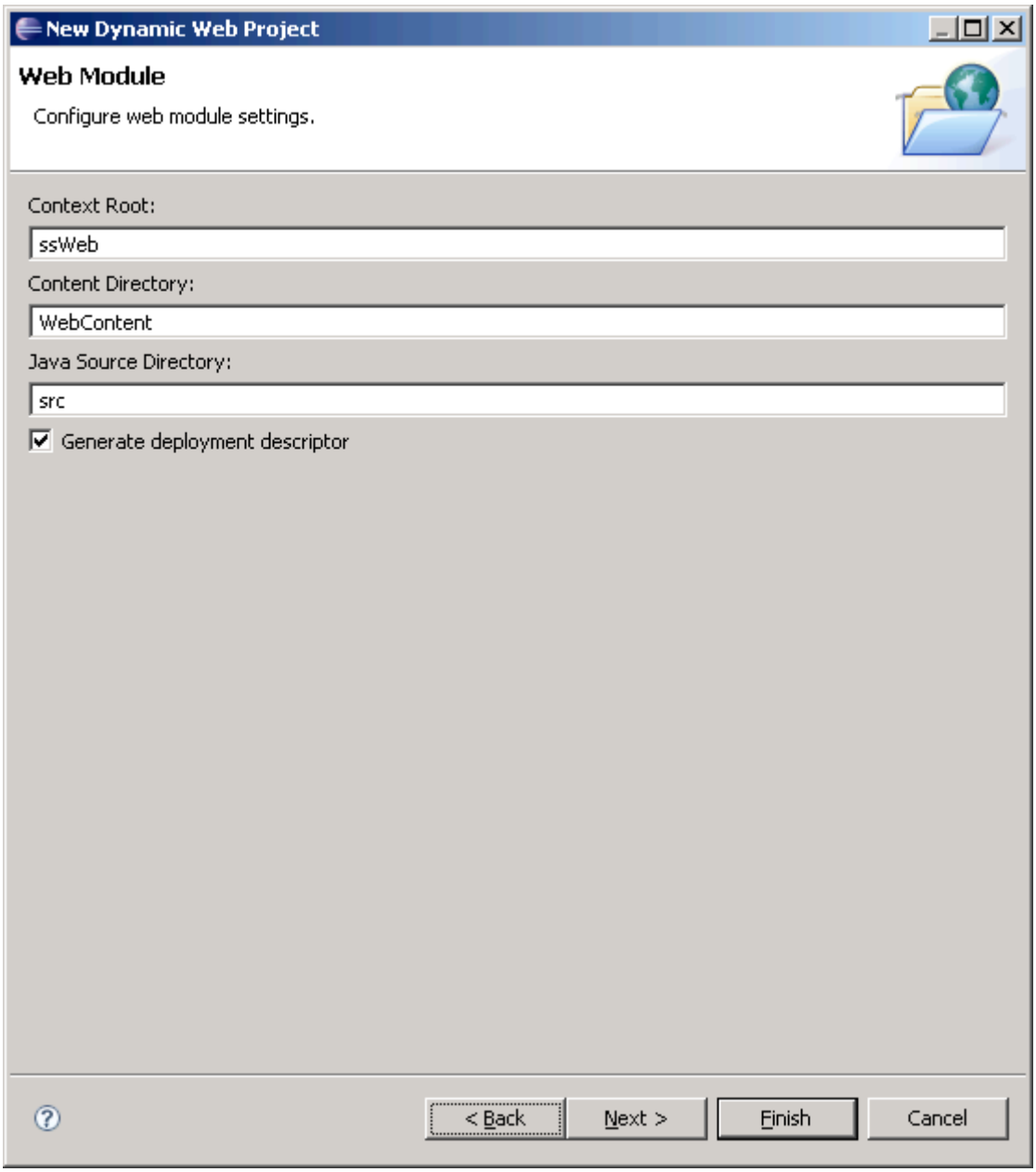
INICIANDO O PROJETO

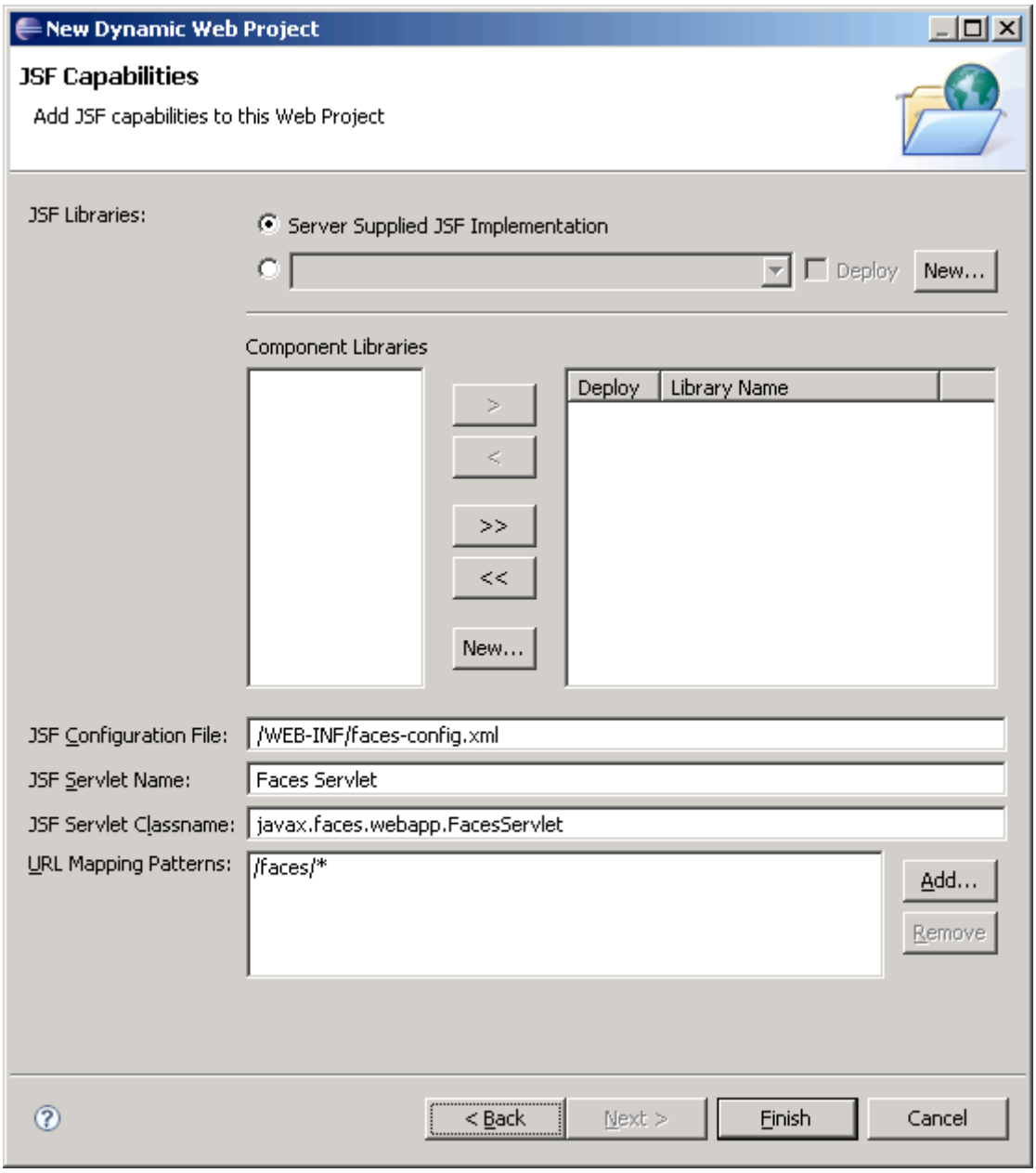
Agora iremos criar o nosso projeto web onde utilizaremos o framework do java server faces. Como dito anteriormente o servidor glassfish já possui uma bilbioteca com a implementação do Java Server faces, e para que estas bibliotecas sejam inclusas na aplicação, precisaremos configurar a aplicação com a faceta do projeto, basta seguir os passos a seguir:

SELECIONAR O MENU FILE > NEW > OTHER > WEB > DYNAMIC WEB PROJECT



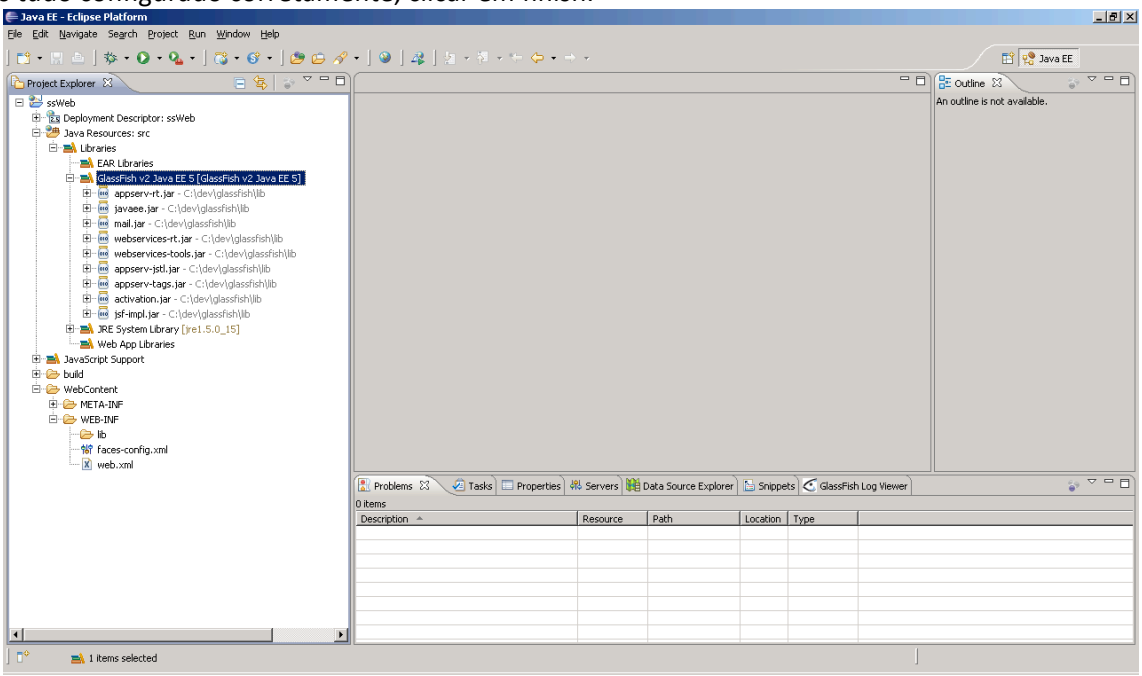
Configurar o seu projeto com o nome de ssWeb, configurar um runtime do Glassfish v2 (versão que você fez o download), módulo da web 2.5 e configuração de faceta do Java Server faces 1.2.


New Dynamic Web Project
Web Module
Configure web module settings.
Context Root:
ssWeb
Content Directory:
WebContent
Java Source Directory:
src
Generate deployment descriptor
< Back
Next >
Finish
Cancel


New Dynamic Web Project
JSF Capabilities
Add JSF capabilities to this Web Project
JSF Libraries:
Server Supplied JSF Implementation
Deploy
New...
Component Libraries
Deploy
Library Name
>
<
>>
<<
New...
JSF Configuration File:
/WEB-INF/faces-config.xml
JSF Servlet Name:
Faces Servlet
JSF Servlet Classname:
javax.faces.webapp.FacesServlet
URL Mapping Patterns:
/faces/*
Add...
Remove
< Back
Next >
Finish
Cancel

Selecionar o radio “Server Supplied Java Server Faces implementation”, o que fará com que o projeto utilize a biblioteca padrão do servidor glassfish.

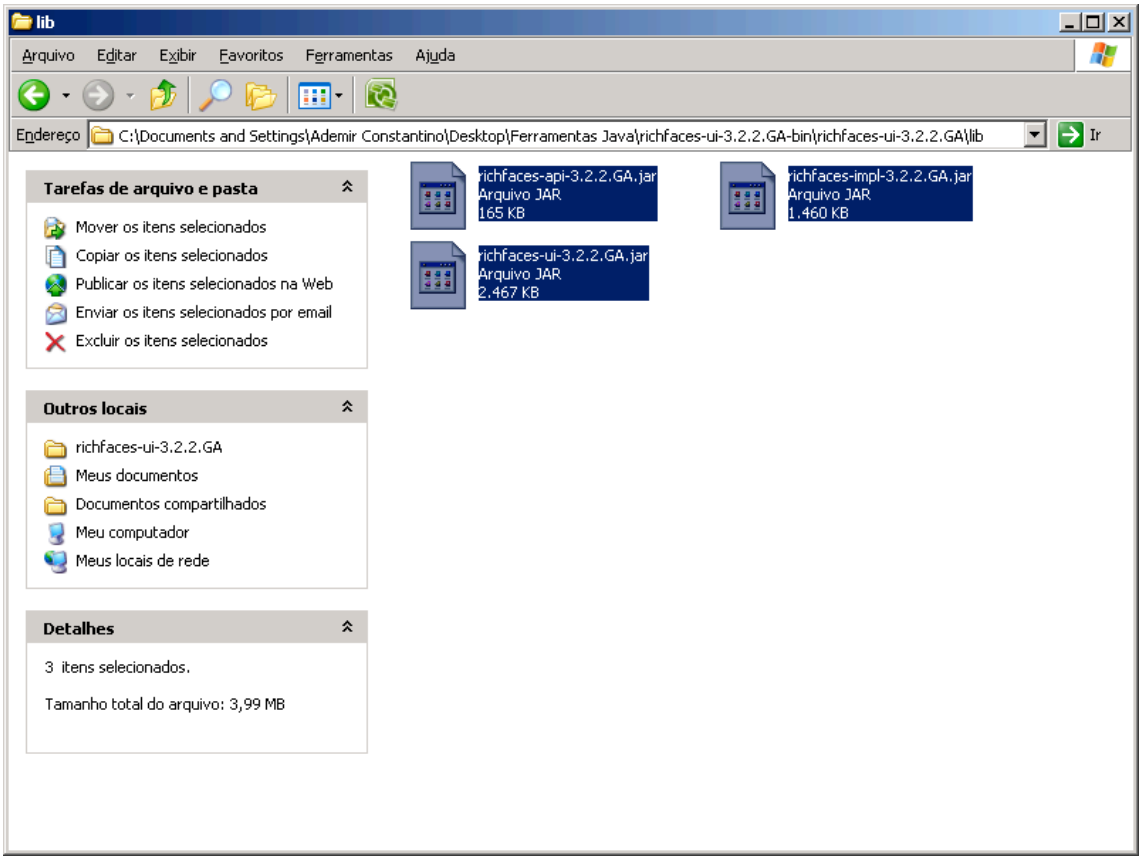
Após tudo configurado corretamente, clicar em finish.

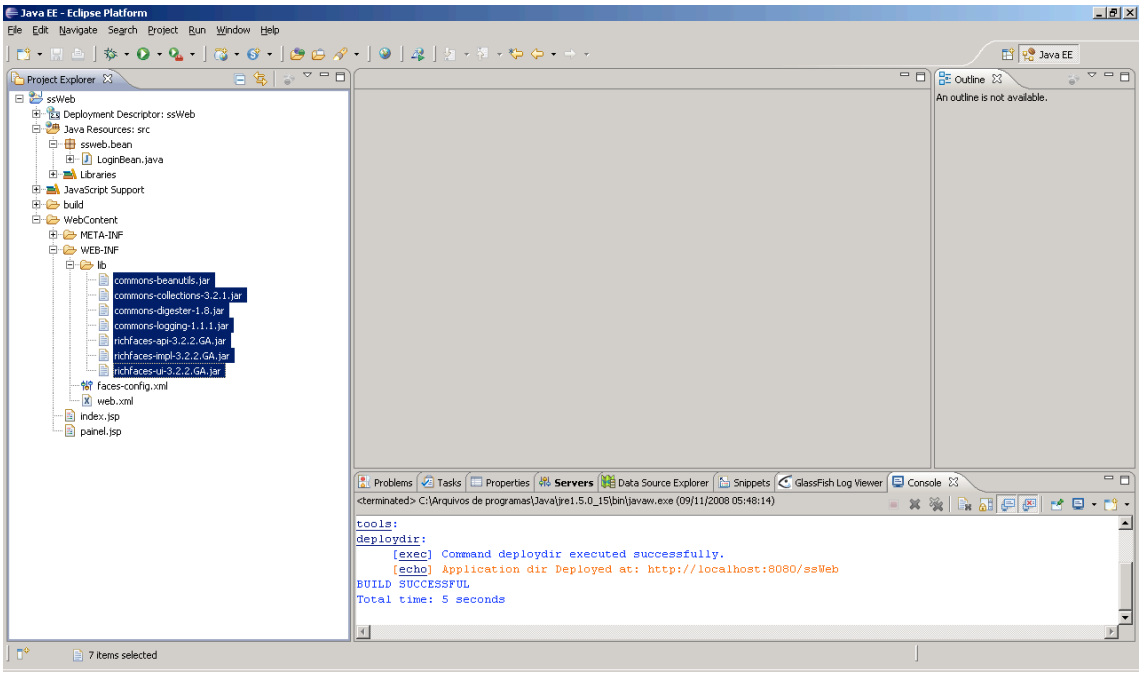


O segundo passo é incluir as bilbiotecas do richfaces e configurar os arquivos xml de configuração do Java Server Faces para fazer a utilização de suas bibliotecas.

Após descompactar o zip de distribuição do richfaces, copiar os arquivos jar richfaces-api-3.2.2.GA.jar, richfaces-impl-3.2.2.GA.jare e richfaces-ui-3.2.2.GA.jar contidos na pasta lib para a pasta lib do seu projeto.

Inclua também os jars do apache commons (logging, collections, beanutils e digester)

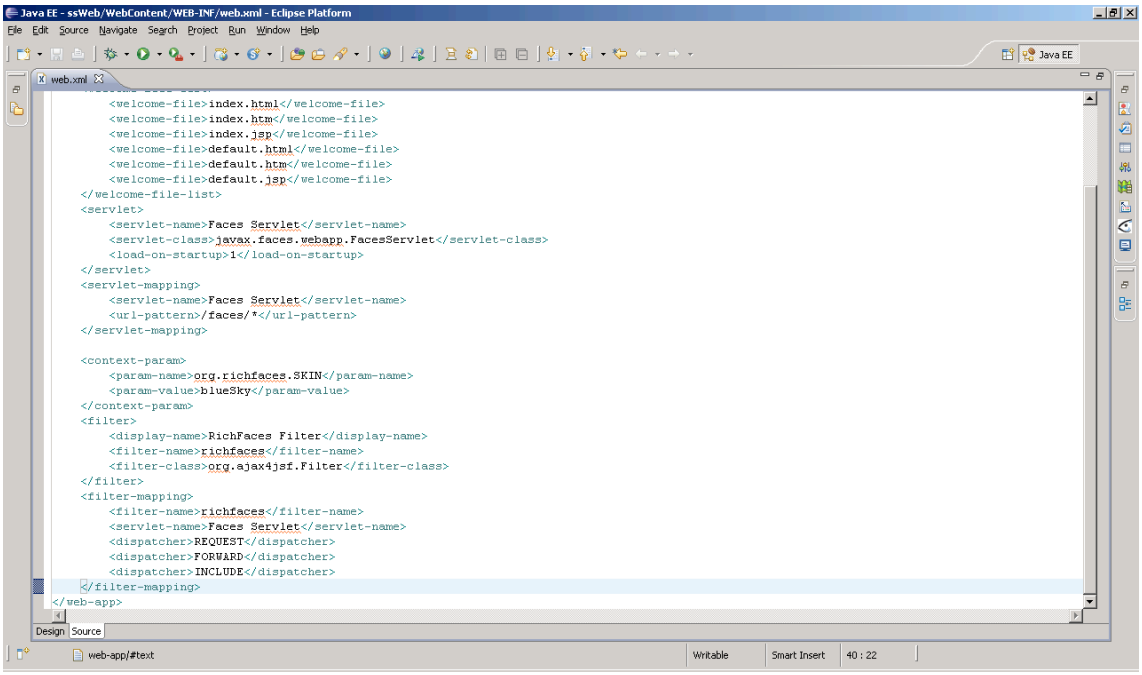

lib
Arquivo Editar Exibir Favoritos Ferramentas Ajuda
Endereço C:\Documents and Settings\Ademir Constantino\Desktop\Ferramentas Java\richfaces-ui-3.2.2.GA-bin\richfaces-ui-3.2.2.GA\lib
Ir
Tarefas de arquivo e pasta
Mover os itens selecionados
Copiar os itens selecionados
Publicar os itens selecionados na Web
Enviar os itens selecionados por email
Excluir os itens selecionados
Outros locais
richfaces-ui-3.2.2.GA
Meus documentos
Documentos compartilhados
Meu computador
Meus locais de rede
Detalhes
3 itens selecionados.
Tamanho total do arquivo: 3,99 MB
richfaces-api-3.2.2.GA.jar
Arquivo JAR
165 KB
richfaces-impl-3.2.2.GA.jar
Arquivo JAR
1.460 KB
richfaces-ui-3.2.2.GA.jar
Arquivo JAR
2.467 KB



Colar na pasta lib de seu projeto.

Em seguida você deverá alterar o arquivo web.xml , incluindo as seguintes linhas:

```
    <context-param>
    <param-name>org.richfaces.SKIN</param-name>
    <param-value>blueSky</param-value>
    </context-param>
    <filter>
    <display-name>RichFaces Filter</display-name>
    <filter-name>richfaces</filter-name>
    <filter-class>org.ajax4jsf.Filter</filter-class>
    </filter>
    <filter-mapping>
    <filter-name>richfaces</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
```



Criando a Página de Login

O próximo passo é criar nossa primeira página utilizando o richfaces, para isso, devo introduzir a estrutura básica dos documentos jsp do Java Server faces.

No seu projeto criar o documento índex.jsp e colocar o seguinte conteúdo dentro dos arquivo:

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@ taglib uri="http://richfaces.org/a4j" prefix="a4j"%>
<%@ taglib uri="http://richfaces.org/rich" prefix="rich"%>
<html>
<head>
<title>Solicitação de Serviço Web</title>
</head>
<body>
<f:view>
</f:view>
</body>
```

```
</html>
```

1. Taglibs:
    a. As duas primeiras linhas, são as taglibs obrigatórias do framework Java Server faces, elas contém as estruturas básicas para se trabalhar com os componentes do JSF.
        i. A taglib core é referente as funcionalidades do framework JSF como validação, conversão e manipulação de eventos.
        ii. A taglib HTML é referente aos componentes HTML (botões, tabelas, campos de texto, etc).
    b. A terceira e quarta linha são as taglibs do richfaces, que contém os componentes e estruturas básicas para se trabalhar com os componentes e ajax.
        i. A taglib rich é referente aos componentes UI ricos, como abas, painéis, menus, etc..
        ii. A taglib Ajax é referente às funcionalidades do Ajax4JSf para se trabalhar com chamadas Ajax.
2. A tag f:view – Esta tag é obrigatória no documento JSF ela indica o início e o fim do documento JSF. È o nível mais alto de tags do JSF.

Criar também o arquivo painel.jsp e colocar o seguinte conteúdo dentro do arquivo:

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@ taglib uri="http://richfaces.org/a4j" prefix="a4j"%>
<%@ taglib uri="http://richfaces.org/rich" prefix="rich"%>
<html>
<head>
<title>Solicitação de Serviço Web</title>
</head>
<body>
<f:view>
<div align="center">
<h:outputText value="Solicitações de Serviço"></h:outputText>

<rich:tabPanel style="width: 500px;">
        <rich:tab label="Em Andamento">
             Não existem itens a serem exibidos.
        </rich:tab>
        <rich:tab label="Encerradas">
            Não existem itens a serem exibidos.
        </rich:tab>
```

```
    </rich:tabPanel>
    </div>
    </f:view>
    </body>
    </html>
```

O objetivo deste tutorial não é abordar todos os componentes do richfaces, mas introduzir os conceitos básicos para que o desenvolvedor através das documentações consiga desenvolver aplicações utilizando todos os componentes disponíveis.

Utilizando o primeiro componente

Após criarmos nossa estrutura básica iremos adicionar um painel do richfaces à nossa aplicação. Este painel será referente ao agrupamento dos campos de login:

```
    <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
    <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
    <%@ taglib uri="http://richfaces.org/a4j" prefix="a4j"%>
    <%@ taglib uri="http://richfaces.org/rich" prefix="rich"%>
    <html>
    <head>
    <title>Solicitação de Serviço Web</title>
    </head>
    <body>
    <f:view>
      <h:form>
      <rich:panel id="painelLogin" style="width: 250px;">

        <f:facet name="header">
          <h:outputText value="Login"></h:outputText>
        </f:facet>

        <h:panelGrid columns="2">
          <h:column><h:outputText value="Login:"/></h:column>
          <h:column><h:inputText/></h:column>
          <h:column><h:outputText value="Senha:"/></h:column>
          <h:column><h:inputText/></h:column>
        </h:panelGrid>

        <h:commandButton value="Logar"></h:commandButton>
      </rich:panel>
      </h:form>
    </f:view>
    </body>
</html>
```

Explicação das tags:

H:form - esta tag produz um formulário do JSF

Rich:panel – A tag rich:panel inclui um painel do richfaces na tela. Todo o seu conteúdo deve ficar aninhado dentro de sua tag.

F:facet name=”header”: - Esta tag indica que estamos configurando o header de nosso painel

H:outputtext – Esta tag produz um label nos padrões do skin utilizado

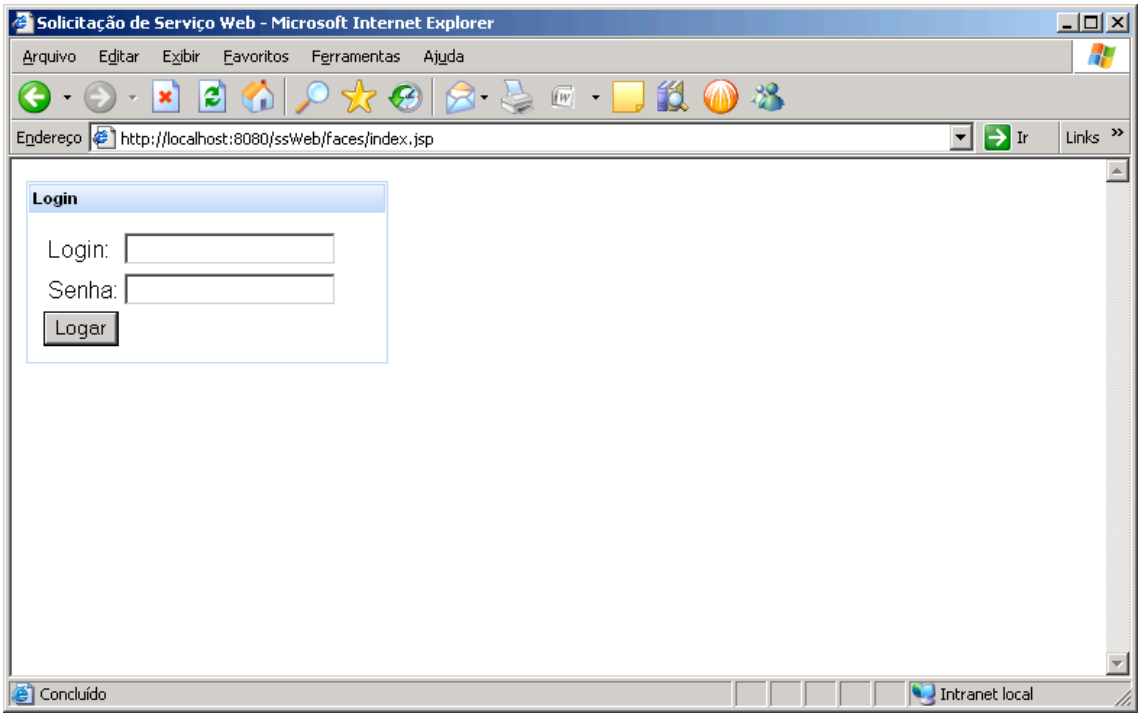
H: panelgrid – Esta tag produz uma tabela orientada a colunnas. O atributo columns indica a quantidade de colunas que a tabela possuirá, a cada 2 tags column uma nova linha é criada.

H: column: A tag column produz uma coluna dentro do panel grid

h:inputText – A tag h:input text produz um campo de entrada de texto, a seguir, veremos como associar este campo com os atributos do back-bean.

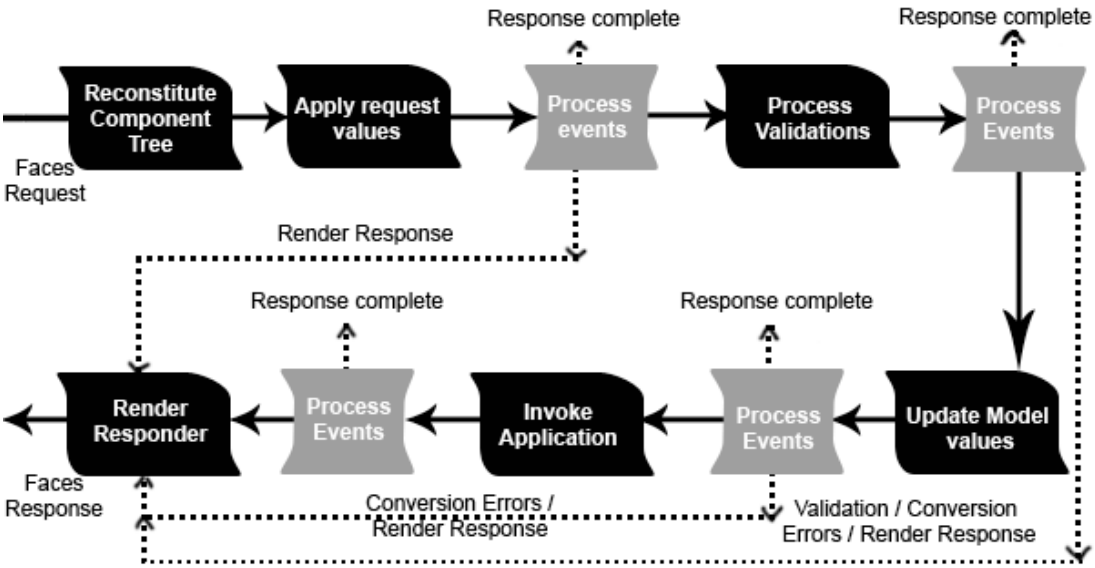
H: commandButton: esta tag produz um botão de ação para enviar o formulário

Resultado:



Associando um Bean à Tela

O próximo passo é criar a camada controller da nossa aplicação de login. O JSF utiliza um modelo diferente dos servlets, struts. Para entendermos melhor estes mecanismos vamos falar sobre o ciclo de vida do JSF. Segue abaixo a imagem do ciclo de vida do JSF.



O ciclo inicia-se quando o usuário faz uma requisição.

- Restaurar a view – Esta fase é iniciada quando o usuário faz uma requisição.
  - Nesta fase, a página é iniciada, os validadores e manipuladores de eventos são invocados e a view é salva no FacesContext.

*****O FacesContext contém toda informação do estado da página, necessária para manipular os componentes GUI para a requisição atual.***

Fase: Atualizar valores do modelo

- Após confirmar que os dados são válidos na fase anterior, os valores dos backing beans são atualizados/armazenados. Caso os dados não sejam compatíveis, o fluxo do ciclo de vida segue para a etapa de exibir a resposta.

- Fase: Invocar a fase da aplicação

- Antes desta fase, os componentes foram convertidos, validados, e armazenados nos beans então agora, você pode utilizar estes dados para executar a regra de negócio da aplicação. Após executar suas regras de negócio, a aplicação pode direcionar para uma página contida na regra de navegação do faces config.

- Faser: Renderizar a resposta
  - Nesta fase a resposta é renderizada ao cliente e os componentes são exibidos em seus estados atuais.

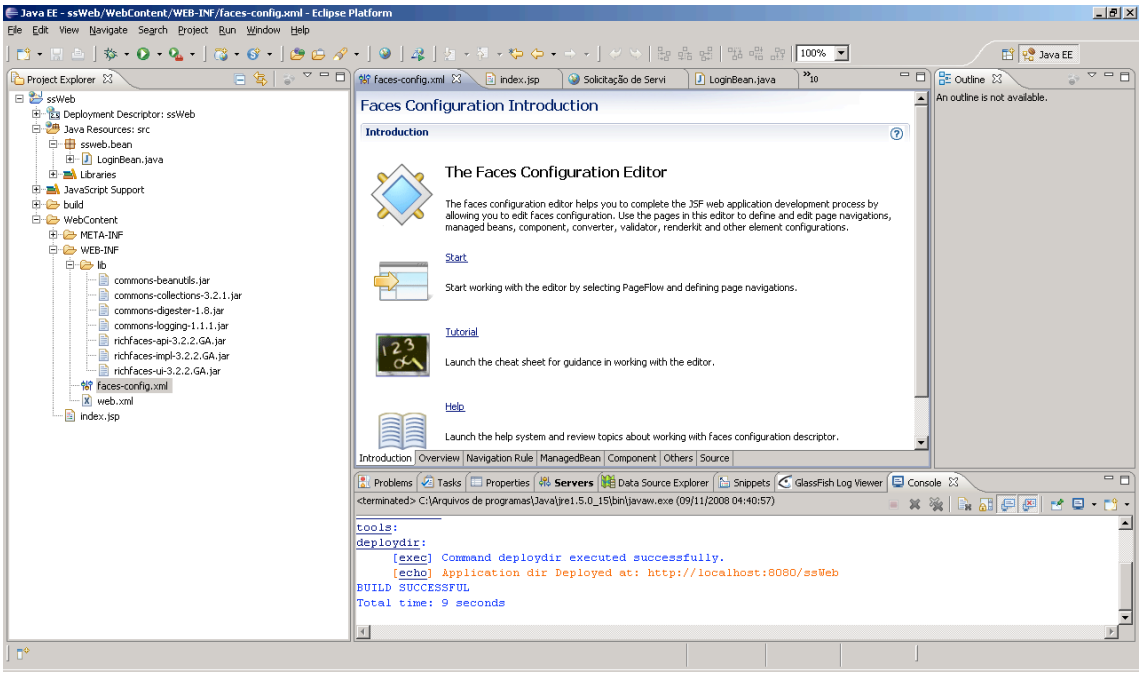
As classes referentes à camada controller no Java Server Faces são comumente chamadas de back-beans(Actions no Struts).

O próximo passo será criar um back bean e associá-lo ao nosso formulário de login, para isto criaremos o pacote ssweb.bean. Crie um bean Java simples com o nome de LoginBean.

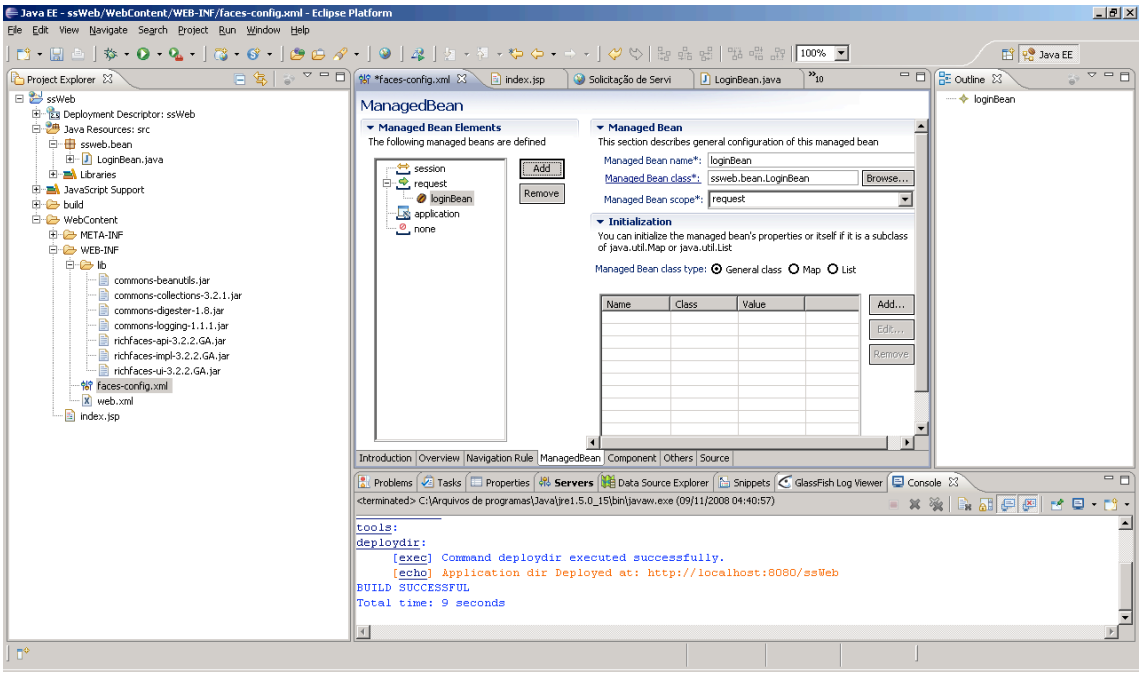
```
package ssweb.bean;

public class LoginBean {

  private String login;
  private String senha;

  /**
   * Método responsável por executar a ação de Login
   * @return String contendo o forward
   */
  public String executarLogin() {

     if(login.equals("admin") && senha.equals("123")) {
        return "sucesso";
     }

     return "falha";
  }

  public String getLogin() {
     return login;
  }
  public void setLogin(String login) {
     this.login = login;
```

```
    }
    public String getSenha() {
       return senha;
    }
    public void setSenha(String senha) {
       this.senha = senha;
    }

}
```

Após criarmos o nosso bean, será necessário configurar o arquivo faces-config.xml. O eclipse provê um editor do faces-config, para abri-lo basta dar um duplo clique no arquivo em sua árvore de navegação.



Após o editor do Arquivo de Configuração do JSF estiver aberto, selecione a aba ManagedBean. E adicione um novo managed bean utilizando-se o escopo de request. Utilize o nome loginBean.



Após criado o bean, será necessária a criação da regra de navegação que direcionará o usuárioo do formulário de login para a tela do painel, caso o usuário e senha estejam corretos.

Para criar esta configuração, selecione no editor do Arquivo de Configuração do JSF a aba source e inclua no arquivo a seguinte configuração:

```
    <navigation-rule>
      <from-view-id>/index.jsp</from-view-id>
      <navigation-case>
        <from-outcome>sucesso</from-outcome>
        <to-view-id>/painel.jsp</to-view-id>
      </navigation-case>
      <navigation-case>
        <from-outcome>falha</from-outcome>
        <to-view-id>/index.jsp</to-view-id>
      </navigation-case>
  </navigation-rule>
```

As regras de navegação definem as regras de mudanças de páginas que podem ocorrer no framework. Estas configurações são obrigatórias e devem ser configuradas corretamente. Para criação de uma regra é utilizada a tag navigation-rule. A tag from-view-id define a qual página a regra

pertence e cada navigation-case case é referente à um fluxo a ser direcionado... Estes fluxos são o retorno string do método referente às ações do commandButton que veremos a seguir.

Configurando o JSP para refletir a ação de Login.

Abra novamente o arquivo index.jsp e altere seu código fonte para:

```
    <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
    <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
    <%@ taglib uri="http://richfaces.org/a4j" prefix="a4j"%>
    <%@ taglib uri="http://richfaces.org/rich" prefix="rich"%>
    <html>
    <head>
    <title>Solicitação de Serviço Web</title>
    </head>
    <body>
    <f:view>
      <h:form>
      <rich:panel id="painelLogin" style="width: 250px;">

         <f:facet name="header">
            <h:outputText value="Login"></h:outputText>
         </f:facet>

         <h:panelGrid columns="2">
            <h:column><h:outputText value="Login:"/></h:column>
            <h:column><h:inputText value="#{loginBean.login}"/></h:column>
            <h:column><h:outputText                                    value="Senha:"
value="#{loginBean.senha}"/></h:column>
            <h:column><h:inputText/></h:column>
         </h:panelGrid>

         <h:commandButton                                              value="Logar"
action="#{loginBean.executarLogin}"></h:commandButton>
      </rich:panel>
      </h:form>
    </f:view>
    </body>
</html>
```

Agora execute a aplicação novamente, forneça o usuário admin e senha 123 e execute a ação de login.

Conclusão

Este foi um tutorial rápido de como iniciar no desenvolvimento de aplicações JSF, espero que possa ter aberto a primeira porta para que você deslanche no desenvolvimento com este framework e venha a criar novos tutoriais para a comunidade Java.

Você pode visualizar exemplos de outros componentes em:

http://livedemo.exadel.com/richfaces-demo

E incluir na página painel.jsp os códigos fontes.

Ademir Constantino