Desenvolvimento Web II

Aula 01 - Introdução ao Javaserver Faces

Apresentação

Olá, seja bem-vindo(a) à nossa primeira aula sobre o JavaServer Faces (JSF). O JSF é um framework (Estrutura reutilizável que disponibiliza funcionalidades para facilitar o desenvolvimento de software) para o desenvolvimento de interfaces de usuário baseada em componentes para aplicações Web utilizando a linguagem Java. É um dos frameworks mais utilizados no mercado, pois é MVC (Model, View e Controller) e contém diversos serviços que facilitam a vida do desenvolvedor.

Nessa aula conheceremos o que é o JSF, suas características e como funciona o ciclo de vida de uma requisição para aplicação. Por fim, criaremos uma aplicação básica para sintetizar o conhecimento adquirido nesta aula.

Vídeo 01 - Apresentação

Objetivos

- Conhecer as características do JavaServer Faces;
- Entender o ciclo de vida das aplicações JSF;
- Conhecer qual é a estrutura de uma aplicação básica com JSF.

O que é o JSF?

O JavaServer Faces (JSF) é um framework MVC para desenvolvimento de aplicações web utilizando Java no lado servidor. Ele é composto por:

- Uma biblioteca para representação dos componentes HTML nas páginas web;
- Conversores de dados;
- Validadores e tratadores de erros.

Todos esses itens que compõem o JSF são importantes para a criação de aplicações web. Com isso, utilizando esses itens poderemos criar páginas web colocando componentes nas mesmas para que possamos manipulá-los, ligá-los a métodos de classes Java que realizarão determinados processamentos no lado servidor.

Uma aplicação web desenvolvida com JSF é similar a qualquer outra aplicação web desenvolvida com outro framework para Java. Dessa forma, ao analisarmos uma aplicação JSF típica encontraremos as seguintes partes:

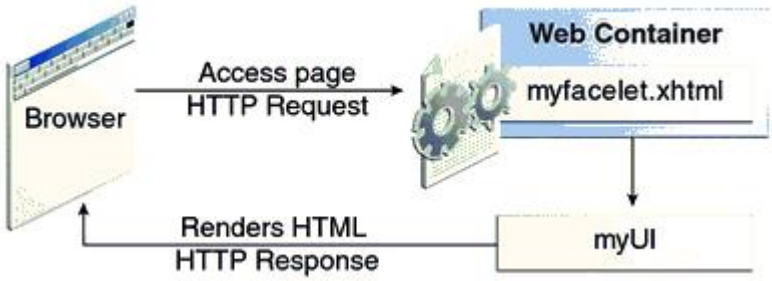
- Um conjunto de páginas XHTML denominadas **facelets**, representando a **view** do MVC. Cada página XHTML possui uma árvore de componentes JSF;
- Um conjunto de classes Java denominadas **managed beans**, que representam o **controller** do MVC;
- Um conjunto de classes de domínio referentes ao negócio da aplicação, representando o **model** do MVC;
- O descritor de implantação web (arquivo **web.xml**).

Poderemos, também, encontrar um ou mais arquivos de configuração da aplicação JSF denominados faces-config.xml. Esses arquivos, que a partir da versão do JSF 2.0 se tornaram opcionais, definem principalmente as regras de navegação da

aplicação. A Figura 1 apresenta como ocorre a resposta a uma requisição vinda de um navegador em uma aplicação JSF.

Vídeo 02 - MVC

Figura 01 - Requisição a uma aplicação JSF.



Fonte: http://docs.oracle.com/javaee/6/tutorial/doc/bnapk.html

A principal característica do JSF é que o mesmo possibilita a separação clara entre o comportamento e a apresentação nas aplicações web. Essa separação permite que o desenvolvimento e a manutenção da aplicação web sejam realizados de maneira mais fácil, o que corresponde a menos retrabalho.

Ciclo de Vida

Agora que você já conhece o que é o JSF, vamos conhecer e entender o ciclo de vida de uma requisição para uma aplicação desenvolvida com esse framework e como sua resposta é gerada. O ciclo de vida do JSF, apresentado na Figura 2, é composto pelas seguintes fases:

1. restaurar a visualização;
2. aplicar valores da requisição;
3. processar as validações;
4. atualizar os valores no modelo;

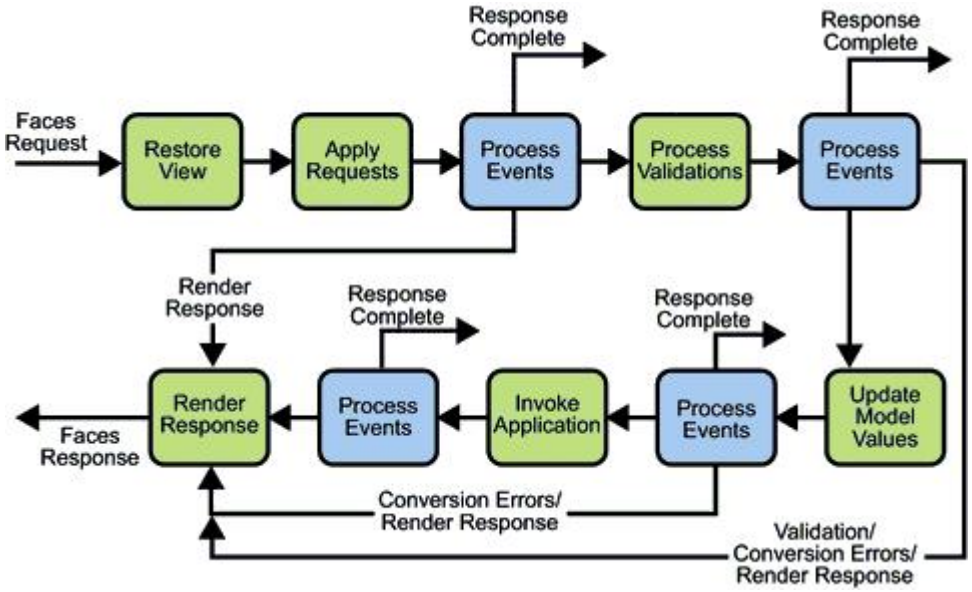
5. invocar a aplicação;

6. renderizar a resposta.

Observe que temos a presença do MVC nesse ciclo de vida, representadas pelas fases de **restaurar a visualização** e **atualizar os valores no modelo**, demonstrando que o JSF realmente é um framework MVC.

Vídeo 03 - Ciclo de Vida

Figura 02 - Ciclo de vida das aplicações JSF.



Então, a primeira fase desse ciclo de vida é a restauração da visualização iniciada a partir de um clique em botão ou link da aplicação JSF. Nessa fase o JSF cria a visualização ligando os eventos e validadores aos componentes definidos na página (árvore de componentes). Essas informações sobre a visualização são incluídas na instância do FacesContext, que é criada ao iniciar uma requisição e destruída no fim da mesma. Após a primeira fase, essa instância possuirá as informações necessárias para processar a requisição.

A segunda fase é a aplicação dos valores da requisição. Todos os valores presentes nos componentes são extraídos e armazenados em seus respectivos tipos. Por exemplo, se o valor presente em um componente não for uma String,

então ele é convertido para seu determinado tipo (boolean, int, long, double, etc). Caso essa conversão falhe, uma mensagem de erro é gerada e associada ao componente e será exibida na fase de renderização da resposta junto com possíveis erros de validação detectados na fase de processamento de validações (terceira fase do ciclo).

A terceira fase é o processamento das validações. Nessa fase todas as validações definidas para os componentes presentes na página são realizadas. Caso alguma validação falhe, uma mensagem de erro é gerada e associada ao componente. Em seguida, a fase de renderização é invocada apresentando as mensagens de validação.

Passamos então para a quarta fase: atualização dos valores do modelo. Observe que as fases de aplicação dos valores da requisição e processamento das validações garantirá que tudo ocorra bem nessa quarta fase, pois todos os componentes já foram convertidos e validados. Com isso, nesse momento os dados do modelo da aplicação são atualizados.

Agora já estamos prontos para a quinta fase: invocação da aplicação. Nessa fase o JSF manipula os eventos da aplicação. É nesse momento que a ação associada a um determinado botão ou link será realizada. Por fim, chegamos à fase de renderização da resposta. Essa é a fase final do ciclo de vida, que tem o objetivo de renderizar a página para o usuário com o resultado de toda a execução do mesmo.

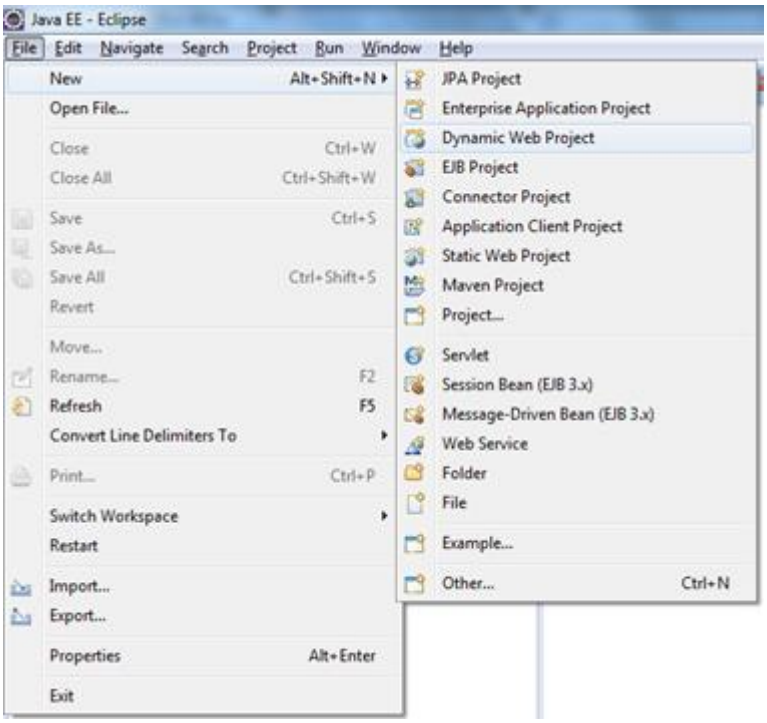
Interessante esse ciclo de vida, não é? Relembre que ele é realizado no lado servidor. Observe que validar as entradas antes de atualizar os valores do modelo da aplicação otimiza o processamento envolvido nesse ciclo, uma vez que a não validação dos dados acarreta na renderização da resposta, ou seja, nenhuma fase posterior é realizada. Agora estamos prontos para criar nossa primeira aplicação com o JSF!

Criando uma Aplicação Básica Web

Após ter conhecido o ciclo de vida do JSF, vamos agora criar uma aplicação básica, nosso Hello World JSF! Esta primeira aplicação lhe dará embasamento para o desenvolvimento de aplicações maiores e mais complexas com o JSF. Você precisará

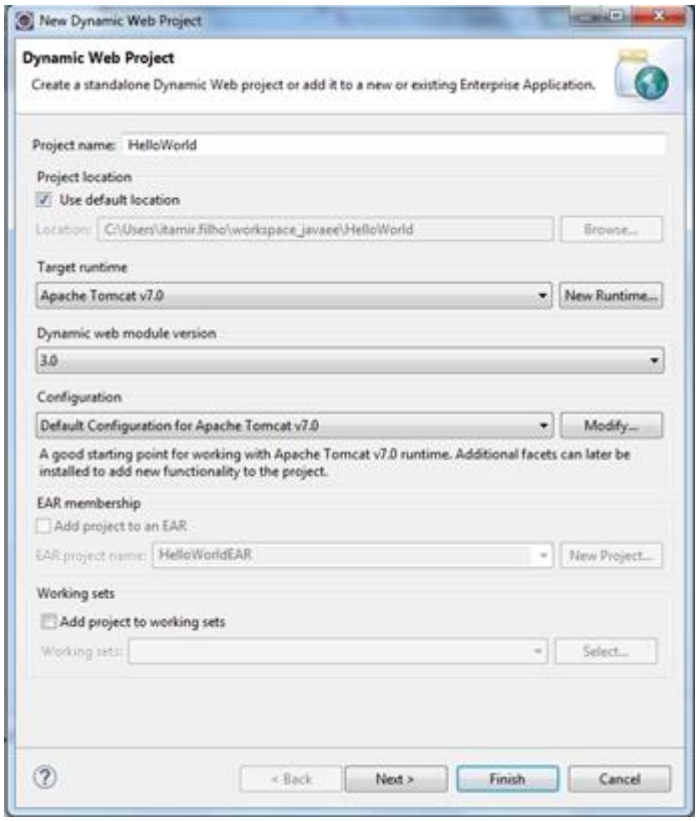
do ambiente eclipse e do servidor de aplicação Apache Tomcat. Segundo a Oracle, apesar da versão 2.3 do JSF está relativamente finalizada, o Java EE 8 ainda não está (na data de criação desse material) e para evitar problemas com o servidor Tomcat estável que estamos utilizando a versão recomendada do JSF é a 2.2.8. Teremos que fazer o download da biblioteca do JSF javax.faces-2.2.8.jar, que está no link: https://maven.java.net/content/repositories/releases/org/glassfish/javax.faces/2.2.8/. Em seguida, no Eclipse, vamos criar um "projeto web dinâmico". O caminho para criação desse tipo de projeto é apresentado na Figura 3.

Figura 03 - Caminho para criação de um projeto web dinâmico (*Dynamic Web Project*)



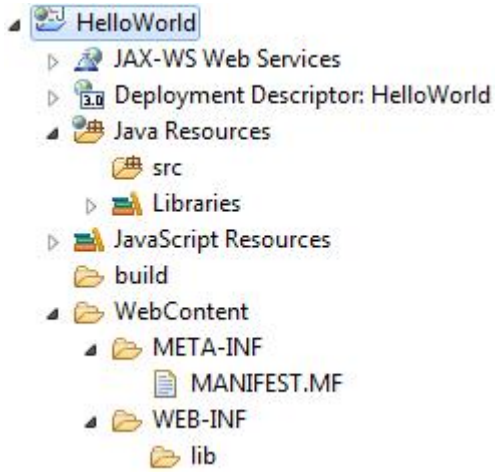
Ao clicar nesse tipo de projeto, uma janela é exibida para que você informe o nome do projeto e o servidor de aplicação de destino. Vamos colocar o nome do nosso projeto de "HelloWorld", nosso servidor de aplicação de destino, o Apache Tomcat, e clicar em finalizar. Essa janela com as configurações corretas é exibida na Figura 4.

Figura 04 - Configuração do nome do projeto e servidor de aplicação.



Ficaremos com a estrutura do nosso projeto semelhante ao que é apresentado na Figura 5.

Figura 05 - Estrutura do projeto JSF HelloWorld.



Coloque dentro da pasta lib do projeto HelloWorld o jar do JSF (javax.faces-2.2.8.jar). Agora criaremos um arquivo chamado web.xml, que deverá ser adicionado na pasta WEB-INF. Esse arquivo conterá a definição da servlet do JSF, que chamaremos de Faces Servlet, e o seu mapeamento. O conteúdo do arquivo web.xml é apresentado na Listagem 1.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">

<display-name>HelloWorld</display-name>

<welcome-file-list>
   <welcome-file>index.jsf</welcome-file>
</welcome-file-list>

<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>

</web-app>
```

Listagem 1 - Código do web.xml

Com o web.xml configurado, vamos criar o pacote *br.ufrn.imd.helloworld.controllers* onde ficará o nosso ManagedBean, denominado HelloWorldMBean. Dentro desse pacote criaremos esse ManagedBean, que tem o conteúdo apresentado na Listagem 2.

```java
package br.ufrn.imd.helloworld.controllers;

import javax.faces.bean.ManagedBean;

@ManagedBean
public class HelloWorldMBean {
   public String getWorld() {
      return "Hello World JSF!";
   }
}
```

Listagem 2 - Código do HelloWorldMBean.

Observe no código da listagem 2 que os ManagedBeans precisam estar configurados com a anotação @ManagedBean. Essa anotação é necessária para que a servlet do JSF (Faces Servlet) saiba que uma determinada classe Java é um

ManagedBean. Veja ainda que nosso ManagedBean possui apenas o método getWorld() que retorna a String “Hello World JSF!”. Está pertinho de rodarmos nosso exemplo, tudo certo até agora?

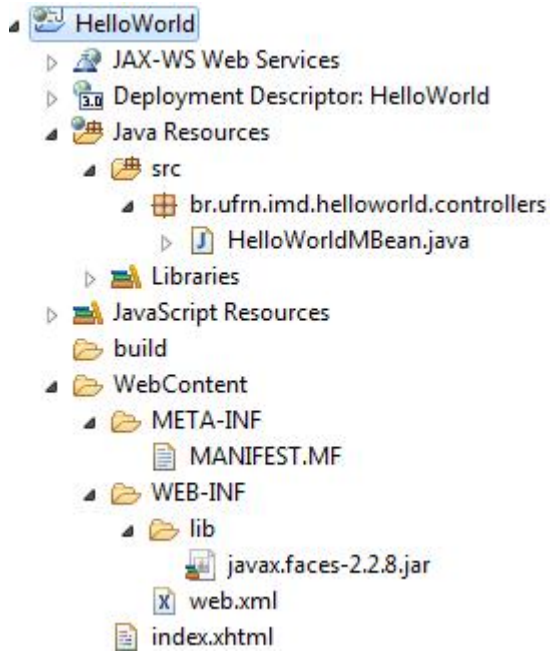
Então, vamos criar agora um arquivo denominado index.xhtml, que representará nossa página web. Essa página, cujo conteúdo é apresentado na Listagem 3, deverá ser criada dentro da pasta WebContent. Observe na linha 7 da listagem 3 que nós temos uma chamada ao método getWorld() do ManagedBean HelloWorldMBean.

```
1  <html xmlns="http://www.w3.org/1999/xhtml"
2  xmlns:h="http://java.sun.com/jsf/html">
3
4  <h:head>
5     <title>Hello World</title>
6  </h:head>
7
8  <h:body>
9     #{helloWorldMBean.world}
10 </h:body>
11
12 </html>
```

Listagem 3 - Código do arquivo index.xhtml.

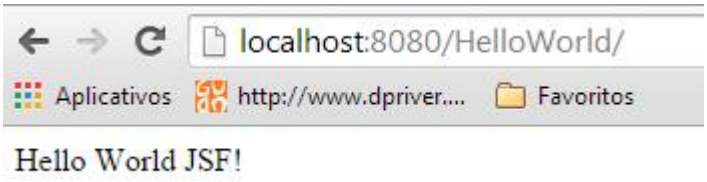
Estamos quase lá! Confira se seu projeto ficou estruturalmente da mesma forma do que é apresentado na Figura 6.

Figura 06 - Estrutura do projeto JSF HelloWorld depois da criação dos arquivos.



Pronto! Agora é só adicionar o projeto HelloWorld ao projetos configurados para o Tomcat, iniciar o mesmo e acessar o endereço http://localhost:8080/HelloWorld/. Fazendo isso você verá a tela apresentada na Figura 7.

Figura 07 - Projeto HelloWorld com JSF em execução.



E com isso chegamos ao fim da nossa aula. Na próxima aula continuaremos apresentando mais sobre o framework JSF para que possamos criar aplicações web mais completas. Estamos só começando!

Resumo

Nessa aula, você aprendeu quais as características do framework JSF e como o mesmo é utilizado para o desenvolvimento de aplicações web. Em seguida, vimos como se comporta o ciclo de vida das aplicações JSF. Por fim, criamos uma aplicação “HelloWorld” para você entender quais as bibliotecas e a estrutura de um projeto desenvolvido com esse framework. Na próxima aula conheceremos as principais TAGs JSF. Até lá e bons estudos!

Autoavaliação

1. Descreva como os managed beans podem ser usados para o desenvolvimento de uma aplicação.
2. Descreva, de forma geral, como podemos implementar uma aplicação web com JSF.
3. Identifique outros pontos em que você poderia usar JSF para melhorar o desenvolvimento de um sistema e implementar melhorias.

Referências

Disponível em: <http://docs.oracle.com/javaee/6/tutorial/doc/bnapk.html>. Acesso em: 19 maio. 2015.

Disponível em: <http://docs.oracle.com/javaee/5/tutorial/doc/bnaqq.html>. Acesso em: 19 maio. 2015.