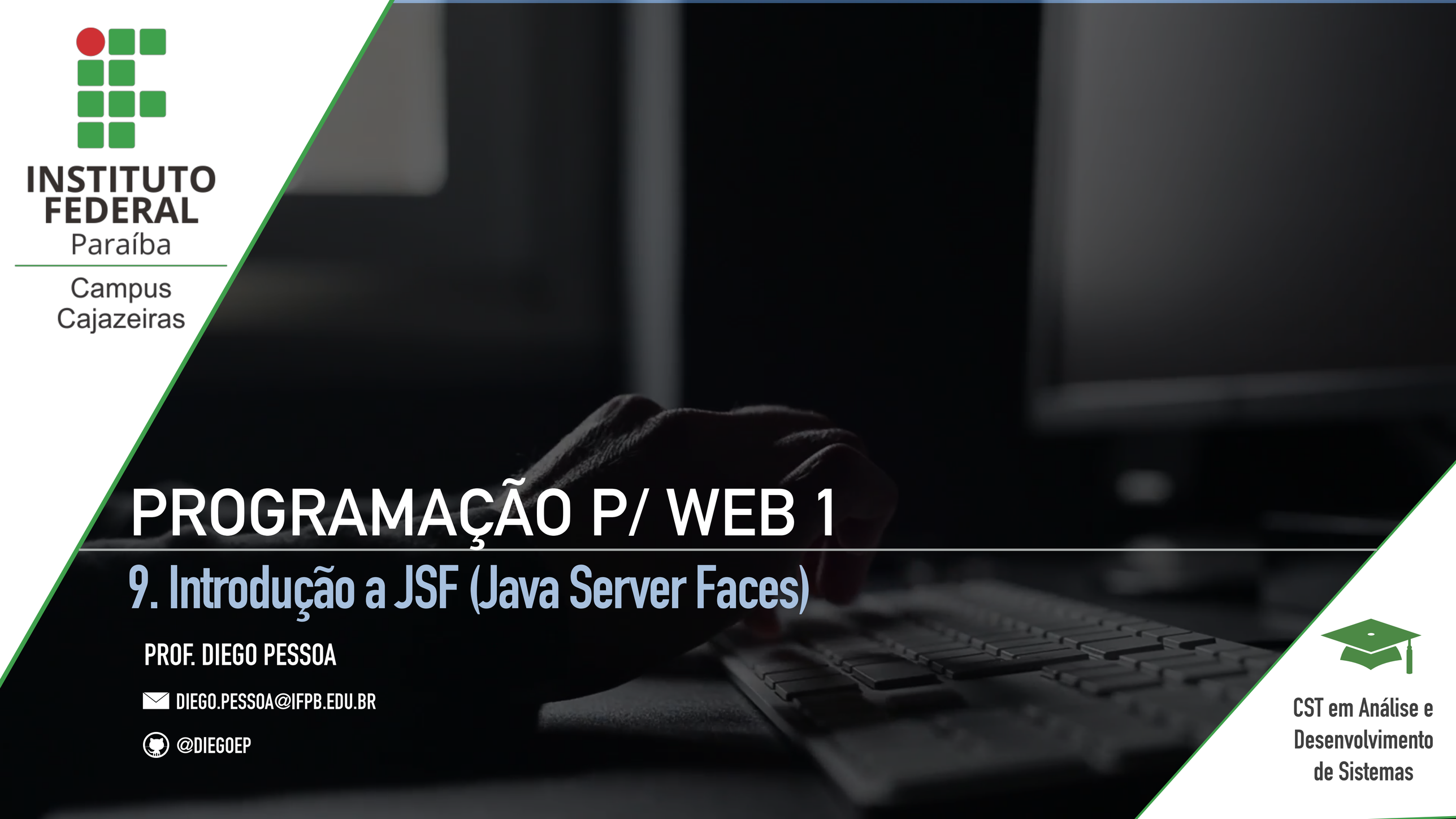


INSTITUTO FEDERAL
Paraíba

Campus Cajazeiras

# PROGRAMAÇÃO P/ WEB 1

## 9. Introdução a JSF (Java Server Faces)


PROF. DIEGO PESSOA

DIEGO.PESSOA@IFPB.EDU.BR

@DIEGOEP


CST em Análise e Desenvolvimento de Sistemas

# Tópicos

- Java Server Faces
  - Definições
  - Componentes
  - Por que JSF?
  - Concorrentes
  - Ciclo de Vida
  - Biblioteca de tags padrão

# O Que E JSF?

- JavaServer Faces (JSF) é um framework para criação de interfaces Web baseadas em componentes
- Java Server Faces (JSF):
  - Conjunto de APIs (Frameworks) que:
    - Possuem componentes de UI prefabricados.
    - Permitem que desenvolvedores adicionem novos componentes
    - Podem prover todo o código necessário para tartar eventos e realizer a organização de componentes
  - Utiliza Java Server Pages e bibliotecas de tags customizadas.

# Partes do JSF

- UI Components
- Eventos e Listeners
- Validadores e Conversores
- Navegação
- Integração com Backend

# Partes do JSF

- **UI Components**
  - A API provê um conjunto de classes e interfaces que especificam o comportamento dos componentes gráficos e suas funcionalidades
  - UIComponent/UIComponentBase
    - Classe base para todos os componentes de interface do usuário
  - Subclasses padrões de UIComponent:
    - UICommand : representa um controle que dispara ações quando requisitado
    - UIForm: agrupa um conjunto de controles que podem submeter dados do usuário para a aplicação

# Partes do JSF

- UIOutput:  exige dados de saída numa página (mostrados pelo navegador)
- UIInput: recebe dados de entrada do usuário
- UISelectItem: representa a seleção de um único item
- UISelectMany: permite que um usuário selecione vários items a partir de um conjunto
- … entre outros não listados aqui

- É possível desenvolver novos componentes

# Partes do JSF

- **Eventos e Listeners**
  - Segue os padrões de especificação de JavaBeans
  - Eventos e Listeners padrões:
    - ActionEvent – UICommand ativado pelo usuário.
    - ValueChangedEvent –UIInput cujo valor foi alterado.

# Partes do JSF

- Tratamento de eventos é similar ao que o ocorre em uma aplicação Java regular:
  - Um objeto Event identifica o componente que gera o evento e guarda informações sobre ele
  - Uma aplicação provê a implementação de uma classe Listener que deve registro no componente que gera o evento
  - Quando usuário ativa o componente, através por exemplo do click em um botão, um evento é disparado
  - Isto faz com que a implementação JSF invoque o método do listener que processa o evento

# Partes do JSF

- JSF suporta três tipos de eventos:
  - Eventos *value-change*
    - Quando o usuário muda o valor de um componente representado por UIInput ou uma de suas subclasses.
    - Um exemplo é selecionar um checkbox, uma ação que resulta na mudança do volor do componente para true.
  - Eventos *action*
    - Um **action event** ocorre quando o usuário ativa um componente que implementa ActionSource.
    - Esses componentes incluem botões e hyperlinks.
  - Eventos *Data model*
    - Ocorre quando uma nova linha de um componente UIData é selecionada.

# Partes do JSF

- **Validators**—Realizam checagem em **UI Input values**
  - Definem um conjunto de classes padrões para realizar validação comum de dados. Crie seus próprios validadores.
  - Biblioteca de tags que define um conjunto de tags correspondente às implementações de validação padrões
  - Registro de um ou mais por componente
  - Enfileiramento uma ou mais mensagens de erros
  - Implementações padrões para casos comuns

# Partes do JSF

- **Conversores**
  - Permitem que um componente de uma aplicação JSF seja associado a um conjunto como um Java Bean.
  - Uma aplicação recupera e seta os dados do objeto para o componente, chamando as propriedades adequadas para aquele componente.
  - Dados podem ser convertidos de Objeto para String e passado para o componente ou
  - Dados podem ser passados para o componente em algum tipo primitivo

# Partes do JSF

- **Renderers**- Adapta componentes a uma marcação específica
  - idioma
  - Decodificação
  - Codificação
  - Uma classe de componente que pode estar restrita a definir uma funcionalidade do componente.
  - Renderização pode ser feita por classes diferentes
    - Cada uma define uma forma de renderizar um component para diferentes clientes
- **RenderKits**—Biblioteca de Renderers
  - Mapeia classes de componente a componentes
  - É uma biblioteca de tags personalizada
  - Ex.: RenderKit HTML básico

# Partes do JSF

- **Modelo de Navegação**
  - Define a sequência em que as páginas são carregadas
  - É de responsabilidade do desenvolvedor da aplicação
  - Definido no arquivo de configuração da aplicação

    (faces-config.xml)

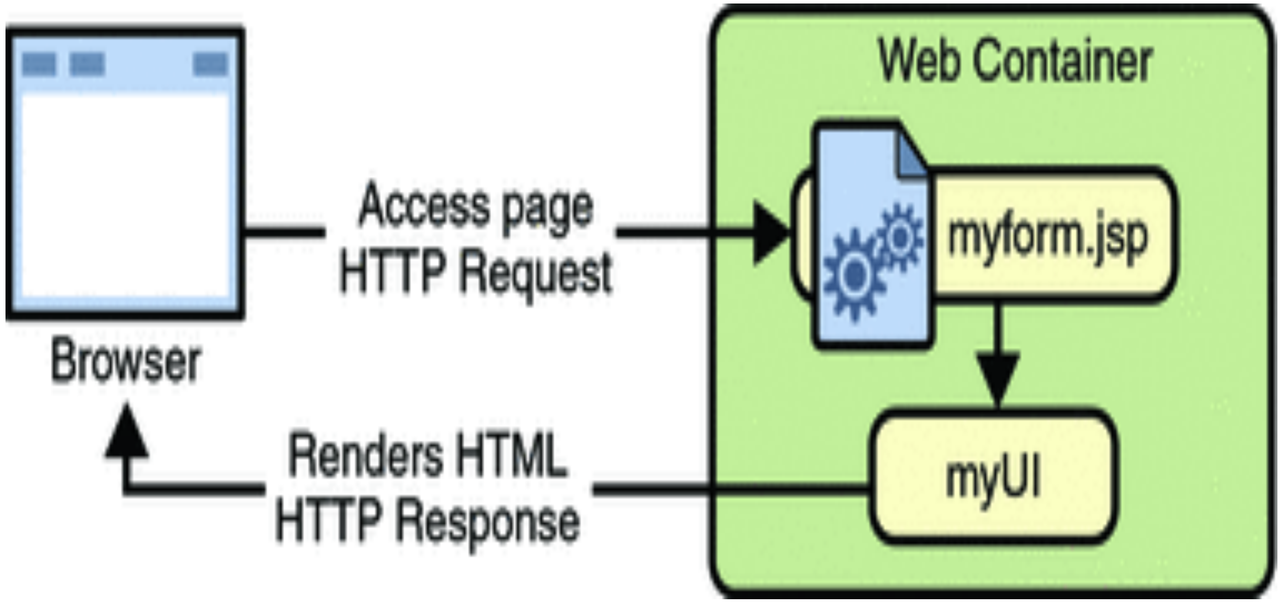
# Partes do JSF

- Exemplos de Regras de navegação:
  - <navigation-rule> <from-view-id>/greeting.jsp</from-view-id> <navigation-case> <from-outcome>success</from-outcome> <to-view-id>/response.jsp</to-view-id> </navigation-case> </navigation-rule>

# JSF–O Que Faz

- Dispõe componentes em uma página através da adição de tags
- Conecta eventos gerados por componentes ao código do backend da aplicação
- Vincula componentes de interface de uma página a dados do lado servidor
- Permite a construção de componentes de UI reusáveis e extensíveis
- Salva e restaura o estado da interface além do ciclo de vida das requisições do servidor

# JSF-O Que Faz:


Web Container
myform.jsp
myUI
Access page
HTTP Request
Renders HTML
HTTP Response
Browser

# JSF–O Que Faz:

- O cliente carrega uma página JSP que inclui tags JSF
  - A interface para a aplicação (representada por "myUI" no slide anterior) gerencia os objetos referenciados pela página JSP. Esses objetos incluem:
    - Os objetos UI component que mapeiam as tags na página JSP
    - Quaisquer listeners de eventos, validadores, e conversores que estão registrados nos componentes
    - Os componentes JavaBeans que encapsulam os dados e funcionalidades específicas da aplicação dos componentes

# JSF–Por que Usar?

- Necessidade de separação do entre comportamento e apresentação.
  - Isto não pode ser alcançado apenas com JSP puro.
    - Uma aplicação JSP não conseguem mapear requisições HTTP para o tratamento de um evento específico de componente, como também não consegue gerenciar elementos da interface como objetos stateful no servidor.
    - JavaServer Faces entrega UI-component and conceitos no nível de Web sem limitar o desenvolvedor a uma particular tecnologia de scripting ou linguagem de marcação.

# JSF–Por que Usar?

- Relativamente fácil de utilizar
- Provê uma arquitetura de componentes extensíveis
- Suporta independência por dispositivo do cliente
- Tem sido amplamente utilizado no mercado

# Conteúdos de uma Aplicação JSF

- Um conjunto de **páginas JSP** (apesar de que não limita-se ao uso de páginas JSP como tecnologia de apresentação)
- Um conjunto de **backing beans**, que são componentes JavaBeans que definem propriedades e funções para UI components numa página
- Um arquivo de configuração da aplicação (faces-config.xml), que define as regras para navegação entre páginas e configura beans e outros componentes personalizados
- Um descritor de implantação (arquivo web.xml)
- Possivelmente um conjunto de objetos personalizados criados pelo desenvolvedor da aplicação. Esses objetos podem incluir componentes personalizados, validadores, conversores ou listeners.
- Um conjunto de tags para representação de objetos em uma página

# Concorrentes

- **Struts / Spring MVC**
  - Struts é um framework MVC popular da Apache Software Foundation.
    - Provê um mecanismo action-based para MVC e não possui um modelo de componentes no estilo do JSF
    - Páginas são mapeadas para modelos que despacham a um servlet de acordo com ações
    - Aplicações Struts usam JSP para renderizar visões e se integram às jsp taglib (vide Apache Beehive)

# Concorrentes

- Struts não suporta
  - Modelo nativo de UI component
  - Modelo de Eventos para UI components
  - Gerenciamento de estados para UI components
  - Suporte de múltiplos renderizadores

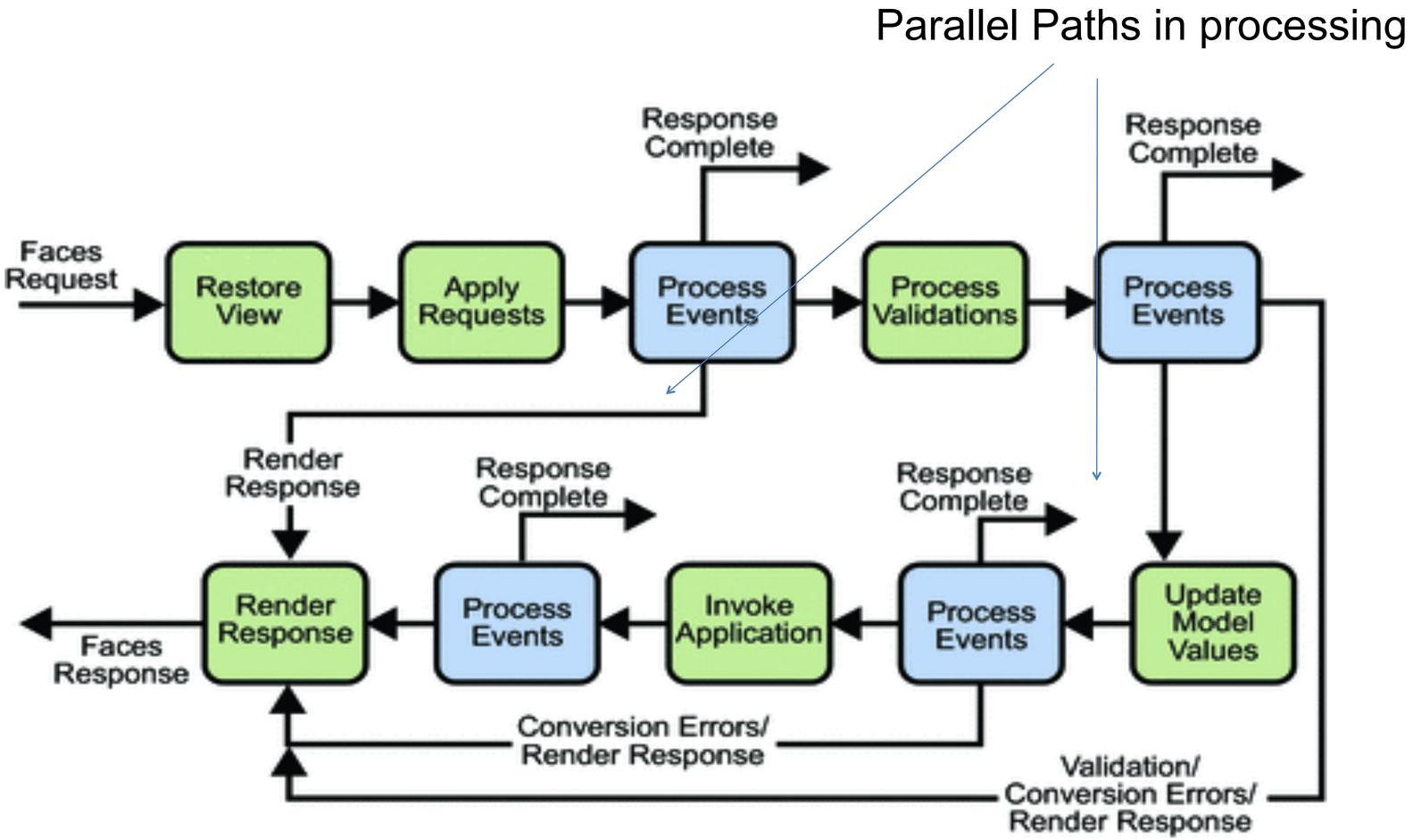
# Competitors

- **Microsoft ASP.NET**
  - As funcionalidades component-based do JSF são similares às oferecidas pelo ASP.NET da Microsoft.
  - Em ASP.NET v1, o código de renderização do componente é empacotado junto com o UI component
    - JSF separa a renderização e interfaces de componente, permitindo a personalização do renderizador.
  - ASP.NET v2 introduziu o suporte a um framework que permite a sobreposição de um renderizador padrão a partir de códigos-fontes de terceiros.

# JSF – Processo de Desenvolvimento

- **Etapas do processo de desenvolvimento**
  - Desenvolver objetos de modelo que guardam os dados
  - Adicionar objetos de modelo (backing beans) e declarações no arquivo faces-config.xml ou anotações
  - Criar páginas usando UI component e tags padrões
  - Definir navegação da página no faces-config.xml
  - Configurar web.xml

# JSF – Ciclo de Vida de uma Requisição



# JSF – Ciclo de Vida de uma Requisição

- **Suponha que um componente JSF submete uma requisição. O ciclo de vida dela (e de seus dados) é:**
  - 1) "Restore View"- Fase de reconstrução da árvore de componentes (chamada View)
    - A implementação JSF deve construir a View considerando o estado salvo de submissões anteriores da página.
    - Tratadores de evento e validadores de componentes salvam o estado da View na instância de FacesContext, que contém todas as informações necessárias para o processamento de uma requisição.

# JSF – Ciclo de Vida

- 2) "Apply request values"
  - Cada componente na árvore extrai seu novo valor a partir dos parâmetros da requisição, utilizando seu método de decodificação
  - O valor é então salvo localmente no componente
- 3) "Process validations"
  - Processamento de todos os validadores registrados nos componentes da árvore
- 4) "Update model values"
  - Após a verificação se o dado é válido, navega-se na árvore de componentes para atribuir propriedades de objeto específicas do lado servidor para valores locais dos componentes
  - Esses objetos são Java Beans (chamados backing beans ou managed beans)

# JSF – Ciclo de Vida

- 5) "Invoke application"
  - Tratamento de eventos do nível da aplicação, tais como submissão de um formulário para redirecionar o usuário para outro recurso.
- 6) Render response phase
  - Delega a função de renderização da página JSP para o container (se a aplicação usa páginas JSP).
  - Os componentes irão se renderizar por conta própria, visto que JSF contém as tags de componentes da página.

# JSF – Bibliotecas Padrões de Tags

- Uma página asp podem incluir componentes JSF
- Para incluí-los, é necessário ter acesso às bibliotecas padrões de tags do JSF.
- Há duas bibliotecas padrões:
  - JavaServer Faces HTML render kit tag library
    - Representa componentes de interface comuns HTML.
  - JavaServer Faces core tag library
    - Tags que realizam ações centrais e são independentes de um kit de renderização particular

# JSF – Bibliotecas Padrões de Tags

- Um exemplo de tags HTML padrões é a **form:**

```
<h:form id="jsftags">
  ...
</h:form>
```

Examplo:

```
<form id="jsftags" method="post" action="/jsftags/faces/pages/tags.jsp" enctype="application/x-www-form-urlencoded">
  ...
        <input type="hidden" name="jsftags" value="jsftags" />        <input type="hidden" name="jsftags:link" />
</form>
```

# JSF – Bibliotecas Padrões de Tags

- Core tag library inclui tags (entre outras) para:
  - Tratamento de eventos
  - Conversão de dados
  - Validação

# Usando as Tags Padrões do JSF

- Uma página JSF típica inclui os seguintes elementos no arquivo jsp:
  - Um conjunto de declarações de biblioteca, que indicam as bibliotecas de tags a serem utilizadas
  - Uma tag view
  - Uma tag form

# Usando as Tags Padrões do JSF

- Para usar qualquer tag do JSF, é preciso incluir as seguintes diretivas taglib no topo de cada página jsp que contém tags definidas por essas bibliotecas:

```
<%@ taglib uri="http://java.sun.com/jsf/html"
  prefix="h" %>

  <%@ taglib uri="http://java.sun.com/jsf/core"
  prefix="f" %>
```

# Usando as Tags Padrões do JSF – Exemplo

- <h:commandButton id="submit" value="#{msg.buttonHeader}" action="nextPage">

  </h:commandButton>

- <h:inputText id="address" value="#{jsfexample.address}" />

# Usando as Tags Padrões do JSF

- As tags “core”:
  - Validation Tags
  - Converter Tags
  - Listener Tags
  - View Tags
  - Select Tags
  - Facet Tag
  - Miscellaneous Tags

# Using the JSF Standard Tags– Validation Tags

- Validation Tags
  - Vincula UI Components e ***Lógica de Validação***. Quatro tags são disponibilizadas por padrão para validação de UI Components:
    - Validate Length Tag
    - Validate Long Range Tag
    - Validate Double Range Tag
    - Validator Tag

# Tags de Validação

- Por exemplo, vamos usar: Validate Length Tag

```
<f:validateLength minimum = "minRange" maximum = "maxRange">

</f:validateLength>
```

# Tags de Validação

- **Exemplo de uso:**

```
<h:inputText value = "#{UserBean.userName}" id =
   "userNameTextField" required="true"> <f:validateLength minimum
   = "10" maximum = "15"/> </h:inputText>
```

- UserBean se refere a um ***Managed Bean***

- userName é uma propriedade da classe UserBean.

  - O atributo 'id' do ***<h:inputText>*** especifica a identificação do UI Component Text Field e deve ser único para os componentes do formulário.

# Tags de Conversão

- **Converter Tags**
  - Convert Number Tag
  - Convert Date Time Tag
  - Converter Tag

# Usando as Tags Padrões do JSF

- **Tags Listener**

- ***JSF UI Components*** emitem ***Eventos*** para ***Listeners*** se registrados.
  - Action Listener Tag
  - Value Change Listener Tag
  - Phase Listener Tag

# Usando as Tags Padrões do JSF

- **Tags View**
  - São tags de ***Container*** usadas para manter u grupo de UIComponent.
    - View Tag
    - Sub View Tag

# EXEMPLO DE APLICAÇÃO EM JSF