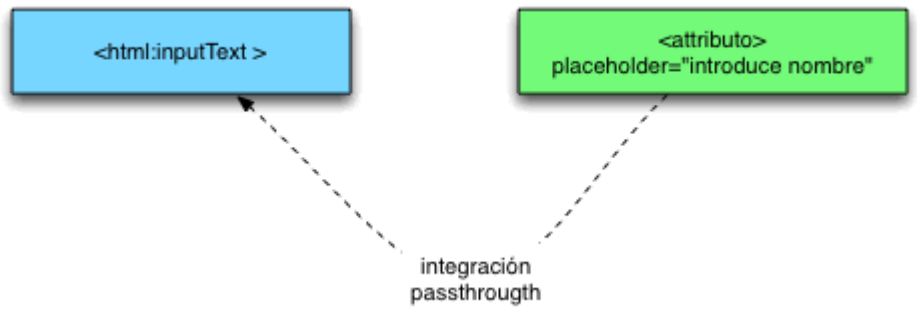
JSF sigue teniendo mucho hueco a la hora de abordar desarrollos web ya que se trata del standard. Quizás una de las cosas que mas se ha echado en falta es la capacidad de integrar las capacidades de HTML5 dentro de él . JSF 2.2 resuelve este problema y nos permite de una forma sencilla que nuestras librerías de etiquetas de JSF puedan usar atributos de HTML5 integrándolos de una forma natural.



Vamos a configurar JSF 2.2 para usarlo en Tomcat 8. El primer paso es añadir a un proyecto Maven las dependencias de JSF 2.2.

```
<dependencies>
 <dependency>
 <groupId>javax.faces</groupId>
 <artifactId>javax.faces-api</artifactId>
 <version>2.2</version>
 </dependency>
 <dependency>
 <groupId>com.sun.faces</groupId>
 <artifactId>jsf-impl</artifactId>
 <version>2.2.9</version>
 </dependency>
 </dependencies>
```

Asignadas las dependencias que necesitamos vamos a configurar el web.xml para dar de alta el controlador de JSF.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app 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

http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 version="3.0">
 <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>

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

Vamos a construir un ManagedBean con un texto sencillo que irá a una etiqueta HTML5 de placeholder.

```
package com.arquitecturajava;

import javax.faces.bean.ManagedBean;
@ManagedBean
public class MensajeBean {

private String placeholder;
public MensajeBean() {
this.placeholder = "introduce tu nombre";
}
public String getPlaceholder() {
return placeholder;
}
public void setPlaceholder(String placeholder) {
this.placeholder = placeholder;
}

}

 
```

Una vez que tenemos configurado el managedBean el siguiente paso es usar JSF HTML5 y las directivas adecuadas para asignar el valor vía HTML5. En este caso vamos a usar el namespace de “passthrough” para que JSF nos permita añadir nuevos atributos a la etiqueta de “inputText”.

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

<h:head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>Hola PlaceHolder</title>
</h:head>
<h:body>
<h:form>
Nombre <h:inputText p:placeholder="hola"/>
</h:form>

</h:body>
</html>
```

Como podemos ver el código es muy sencillo y la etiqueta inputText recibirá el atributo de “placeHolder” que recordemos nos permite asignar un texto por defecto a la etiqueta antes de rellenarla. Realizados estos pasos el resultado será el siguiente:

Si revisamos el código html generado tendremos como atributo la etiqueta placeholder:

```
<html>
.......
<input type="text" name="j_idt5:j_idt7" placeholder="hola" />
...
</html>
```

Otros artículos relacionados : JSF Rendered Attributes , JSF vs Spring MVC