Escuela Colombiana de Ingeniería

Procesos de desarrollo de Software – PDSW

Frameworks Web MVC – Java Server Faces / Prime Faces

En este ejercicio, usted va a desarrollar una aplicación Web basada en el marco JSF, y en una de sus implementaciones más usadas: PrimeFaces. Se trata de un juego en línea para adivinar un número, en el que el ganador, si atina en la primera oportunidad, recibe `$100.000`. Luego, por cada intento fallido, el premio se reduce en `$10.000`.

1. Construya un proyecto Maven, usando el arquetipo de aplicación Web estándar maven-archetype-webapp:
2. Al proyecto Maven, debe agregarle las dependencias mas recientes de `javaee-api`, `jsf-api`, `jsf-impl`, `jstl` y Primefaces (en el archivo pom.xml).
3. Agregar a la sección build:

```
<plugins>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
        <configuration>
            <source>1.8</source>
            <target>1.8</target>
            <compilerArguments>
                <endorseddirs>${endorsed.dir}</endorseddirs>
            </compilerArguments>
        </configuration>
    </plugin>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.3</version>
        <configuration>
            <failOnMissingWebXml>false</failOnMissingWebXml>
        </configuration>
    </plugin>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-dependency-plugin</artifactId>
        <version>2.6</version>
        <executions>
            <execution>
```

```
                <phase>validate</phase>
                <goals>
                    <goal>copy</goal>
                </goals>
                <configuration>
                    <outputDirectory>${endorsed.dir}</outputDirectory>
                    <silent>true</silent>
                    <artifactItems>
                        <artifactItem>
                            <groupId>javax</groupId>
                            <artifactId>javaee-endorsed-api</artifactId>
                            <version>7.0</version>
                            <type>jar</type>
                        </artifactItem>
                    </artifactItems>
                </configuration>
            </execution>
        </executions>
    </plugin>

    <!-- Tomcat embedded plugin. -->
    <plugin>
        <groupId>org.apache.tomcat.maven</groupId>
        <artifactId>tomcat7-maven-plugin</artifactId>
        <version>2.2</version>
        <configuration>
            <port>8080</port>
            <path>/</path>
        </configuration>
    </plugin>
</plugins>
```

4. Empaquete la aplicacion con maven: `mvn package` y ejecutela utilizando `mvn tomcat7:run`. Revise que la aplicacion este corriendo en el puerto asignado utilizando el browser.

5. Para que configure automáticamente el descriptor de despliegue de la aplicación (archivo web.xml), de manera que el *framework* JSF se active al inicio de la aplicación, en el archivo `web.xml` agregue la siguiente configuración:

```
<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.jsp</welcome-file>
</welcome-file-list>
```

6. Revise cada una de las configuraciones agregadas anteriormente para saber qué hacen y por qué se necesitan. Elimine las que no se necesiten.

7. Ahora, va a crear un `Backing-Bean` de sesión, el cual, para cada usuario, mantendrá de lado del servidor las siguientes propiedades:

   a. El número que actualmente debe adivinar.

   b. El número de intentos realizados.

   c. El premio acumulado hasta el momento.

   d. El estado del juego, que sería una cadena de texto que indica si ya ganó o no, y si ganó de cuanto es el premio.

   Para hacer esto, cree una clase que tenga:

   - el constructor por defecto (sin parámetros);
   - los métodos `set/get` necesarios dependiendo si las propiedades son de escritura o lectura;
   - coloque las anotaciones:
     - `@ManagedBean`, incluyendo el nombre: `@ManagedBean(name = "guessBean")` y
     - `@ApplicationScoped`.

   A la implementación de esta clase, agregue un método llamado `restart`, el cual sirva para volver a iniciar el juego (inicializar de nuevo el número a adivinar, y restaurar el premio a su valor original).

8. Cree una página XHTML, de nombre `guess.xhtml` (debe quedar en la ruta `src/main/webapp`). Revise en la página 13 del manual de PrimeFaces, qué espacios de nombres XML requiere una página de PrimeFaces, y cual es la estructura básica de la misma.

9. Con base en lo anterior, agregue un formulario con identificador `guess_form` con el siguiente contenido básico:

```
<h:body>
 <h:form id="guess_form">

 </h:form>
</h:body>
```

10. Al formulario, agregue:

    a. Un elemento `<p:inputText>` para que el usuario ingrese su número.

b. Un elemento de tipo `<p:outputLabel>` para mostrar el número de intentos realizados.

c. Un elemento de tipo `<p:outputLabel>` para mostrar el estado del juego.

d. Un elemento de tipo `<p:outputLabel>` para mostrar en cuanto va el premio.

Y asocie dichos elementos al BackingBean de sesión a través de su propiedad `value`, y usando como referencia el nombre asignado:

```
value="#{guessBean.nombrePropiedad}"
```

11. Al formulario, agregue dos botones de tipo `<p:commandButton>`, uno para enviar el número ingresado y ver si se *atinó*, y otro para reiniciar el juego.

    a. El botón de *envío de adivinanza* debe tener asociado a su propiedad `update` el nombre del formulario en el que se agregaron los campos antes descritos, de manera que al hacer clic, se ejecute un ciclo de JSF y se *refresque* la vista:

    ```
    <p:commandButton update="guess_form">...
    ```

    b. El botón de reiniciar juego tendrá la misma propiedad de `update` del otro botón, más la propiedad `actionListener`, con la cual se le indicará que, al hacer clic, se ejecutará el método `restart`, creado en el backing-bean de sesión:

    ```
    <p:commandButton update="…" actionListener="#{guessBean.restart}">
    ```

12. Para verificar el funcionamiento de la aplicación, agregue el plugin tomcat-runner dentro de los plugins de la fase de construcción (build). Tenga en cuenta que en la configuración del plugin se indica bajo que ruta quedará la aplicación:

```
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
        <path>/</path>
    </configuration>
</plugin>
```

13. Una vez hecho esto, desde la terminal, en el directorio del proyecto, ejecute:

    a. mvn package

    b. mvn tomcat7:run

    Si no hay errores, la aplicación debería quedar accesible en la URL: http://localhost:8080/faces/guess.xhtml

14. Si todo funcionó correctamente, realice la siguiente prueba:

    a. Abra la aplicación en un explorador. Realice algunas pruebas con el juego e intente adivinar el número.

    b. Abra la aplicación en dos computadores diferentes. Si no dispone de uno, hágalo en dos navegadores diferentes (por ejemplo Chrome y Firefox; incluso se puede en un único navegador usando una ventana normal y una ventana de incógnito / privada). Haga cinco intentos en uno, y luego un intento en el otro. ¿Qué valor tiene cada uno?

    c. Aborte el proceso de Tomcat-runner haciendo Ctrl+C en la consola, y modifique el código del backing-bean de manera que use la anotación @SessionScoped en lugar de @ApplicationScoped. Reinicie la aplicación y repita el ejercicio anterior.

        - ¿Coinciden los valores del premio?.
        - Dado la anterior, ¿Cuál es la diferencia entre los backing-beans de sesión y los de aplicación?