Resumen llamadas cliente-servidor (RPC):

1. En el cliente:

Se definen los servicios,

Primero la interfaz del servicio:

import com.google.gwt.user.client.rpc.RemoteService;

import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

@RemoteServiceRelativePath("**sampleservice**")

public interface SampleService extends RemoteService{

String sayHello(String name);

}

También la interfaz asincrónica del pedido de servicio:

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface SampleServiceAsync {

void sayHello(String name, AsyncCallback callback);

}

Y la clase asincrónica de la devolución del pedido. Esto define que hace el cliente con la respuesta del servidor

import com.google.gwt.user.client.Window;

import com.google.gwt.user.client.rpc.AsyncCallback;

/\*\*

\* Class which handles the asynchronous callback from the server

\*

\* Need to react on server communication failure and success

\*/

public class SampleCallback implements AsyncCallback {

@Override

public void onFailure(Throwable caught) {

// handle failure from server.

Window.alert("Not able to process client reuest. Exception occured at server: " + caught);

}

@Override

public void onSuccess(String result) {

// handle the successful scenario.

Window.alert("Client request processed sucessfully. Result from server: " + result);

}

}

1. En el servidor:

Definir la implementación de la llamada del cliente, y que es lo que hara el servidor

import com.google.gwt.user.server.rpc.RemoteServiceServlet;

import com.javacodegeeks.helloworld.client.service.SampleService;

public class SampleServiceImpl extends RemoteServiceServlet implements SampleService {

@Override

public String sayHello(String name) {

return "Hello " + name;

}

}

1. En el web.xml

Vincular los servlets del cliente con el servidor

<!-- Servlets -->

<servlet>

<servlet-name>sampleServlet</servlet-name>

<servlet-class<com.javacodegeeks.helloworld.server.SampleServiceImpl</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>sampleServlet>/servlet-name>

<url-pattern>/samplewebapplication/sampleservice</url-pattern>

</servlet-mapping>

</pre>

Ejemplo Completo:

/\*\*

\* Entry point classes define onModuleLoad().

\*/

public class SampleWebApplication implements EntryPoint, ClickHandler{

/\*\*

\* Instantiates service.

\*/

SampleServiceAsync sampleServiceAsync = GWT.create(SampleService.class);

/\*\*

\* Label & Text Box.

\*/

Label lbl; TextBox textBox;

/\*\*

\* This is the entry point method.

\*/

public void onModuleLoad() {

VerticalPanel verticalPanel = new VerticalPanel();

verticalPanel.setSize("100%", "100%");

verticalPanel.setVerticalAlignment(HasVerticalAlignment.ALIGN\_MIDDLE);

verticalPanel.setHorizontalAlignment(HasHorizontalAlignment.ALIGN\_CENTER);

textBox = new TextBox();

Button btn = new Button("Get Update from Server"); btn.addClickHandler(this);

lbl = new Label("The text will be updated here.");

Image image = new Image();

image.setUrl("https://www.javacodegeeks.com/wp-content/uploads/2012/12/JavaCodeGeeks-logo.png");

verticalPanel.add(textBox); verticalPanel.add(btn); verticalPanel.add(lbl);

verticalPanel.add(image);

RootLayoutPanel.get().add(verticalPanel);

}

@Override

public void onClick(ClickEvent event) {

sampleServiceAsync.sayHello(textBox.getText(), new AsyncCallback() {

@Override

public void onFailure(Throwable caught) {

// handle failure from server.

Window.alert("Exception Received from server.");

}

@Override

public void onSuccess(String result) {

lbl.setText(result);

}

});

}

}