Pasar de *App Inventor* a *Android Studio*

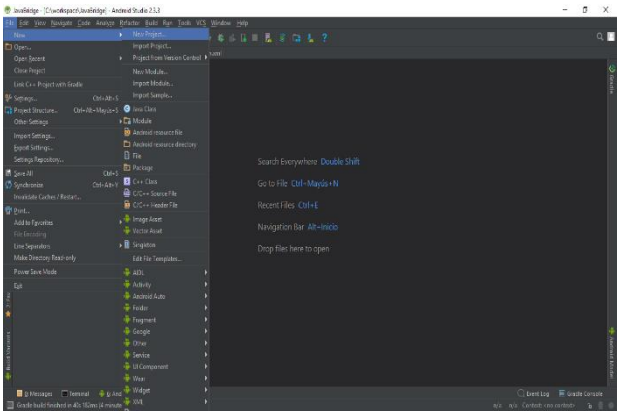
JAVA BRIDGE

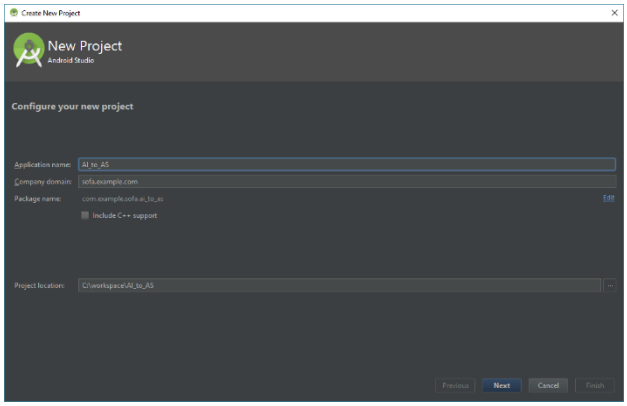
¿Qué es **Java Bridge**? Se trata de una **librería** de Java para App Inventor para crear aplicaciones en Android. Es más sencillo de aprender que el SDK de Android y utiliza la misma terminología de App Inventor – hay una clase Java para cada componente.

Para empezar debemos tener instalado el **Java SE Development Kit** (Java SDK) y **Android Studio** y debemos configurarlo, para ello seguimos los pasos de la **página oficial**.

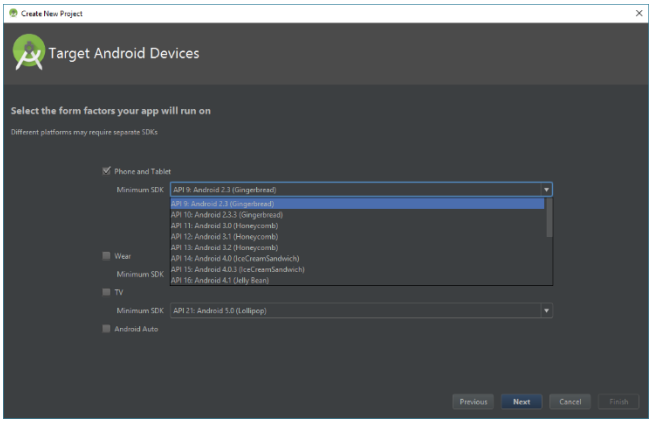
Una vez tenemos todo descargado y preparado comenzaremos con Java Bridge, para ello vamos a crear una aplicación sencilla: cuando pulsemos un botón cambie de color. Para ello:

- Abrir Android Studio y crear nuevo proyecto.

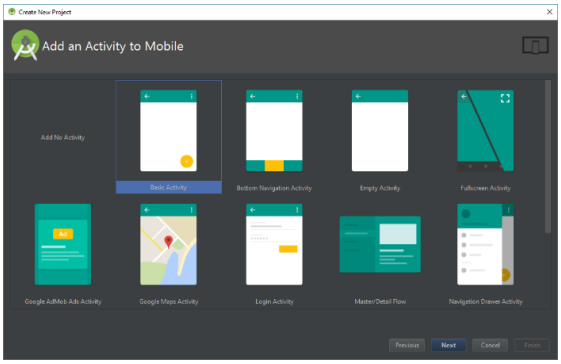




- Establecemos la versión mínima y máxima del SDK, mientras más baja en el *minimun SDK* mayor será el alcance de las apps, puesto que aún hay muchos móviles con versiones antiguas.



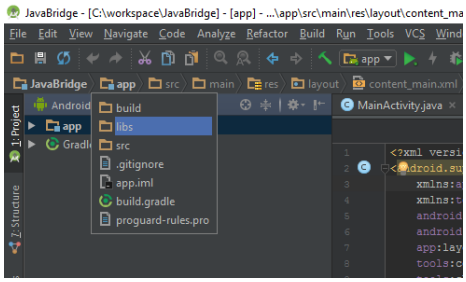
- En la siguiente pantalla elegimos Basic Activity y le damos a siguiente hasta finalizar.



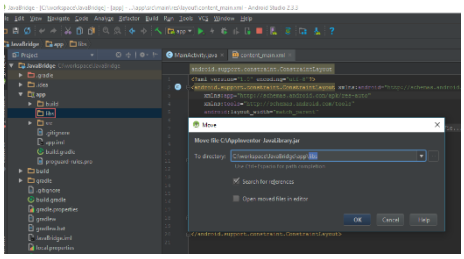
- Si todo va bien nos aparecerá la siguiente pantalla con el ya conocido “Hola Mundo”en la pestaña del *Design* dentro del fichero **activity_main.xml** y ejecutamos la app en nuestro dispositivo móvil o emulador (creando una nueva máquina virtual) pulsando sobre la flecha verde ‘Run app’.

Una vez tenemos esto vamos a instalar la librería Java Bridge. Nos dirigimos de nuevo a nuestro proyecto creado en Android Studio:

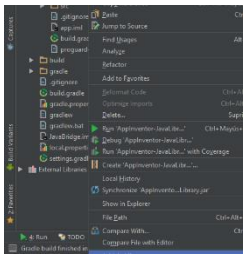
- Pulsar sobre **app/libs**.



- Localizar la librería descargada y arrastrar hasta la carpeta /lib del proyecto, si no nos dejase desde el explorador de carpetas localizar el proyecto y copiar ahí la librería descargada. C:\workspace\JavaBridge\app\libs



- Añadirla como librería pulsando sobre el botón derecho sobre esta y ya la tendremos cargada.



Pasemos a crear la aplicación, nos dirigimos a **MainActivity.java**:

- Importamos lo siguiente de app inventor necesario para todas las apps con Java Bridge:

import com.google.appinventor.components.runtime.HandlesEventDispatching;
import com.google.appinventor.components.runtime.EventDispatcher;
import com.google.appinventor.components.runtime.Form;
import com.google.appinventor.components.runtime.Component;

- En la clase extendemos a:
public class MainActivity extends Form implements HandlesEventDispatching {

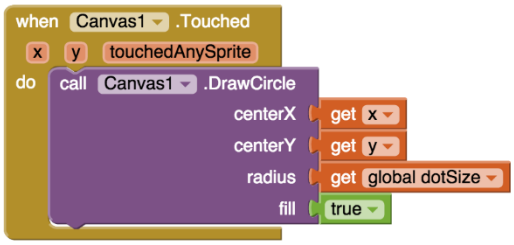
- **Crear la interfaz de usuario**

```
public class MainActivity extends Form implements HandlesEventDispatching
{
    //Se declaran las variables como variables de instancia de la clase
    private float dotSize;

    private Canvas canvas1;
    private HorizontalArrangement horizontalArrangement1;
    private Button redButton;
    private Button blueButton;
    private Button greenButton;
    private Button bigButton;
    private Button smallButton;
    private Button takePictureButton;
    private Camera camera1;

    //en este método es donde crearemos los componentes, inicializados como en
las aplicaciones de AppInventor

    protected void $define() {
        this.Title("Paint Pot");
        this.Icon("kitty.png");
        dotSize = 2;
        canvas1 = new Canvas( this );
        canvas1.Height( 300 );
        canvas1.Width(LENGTH_FILL_PARENT);
        canvas1.BackgroundImage( "kitty.png" );
        canvas1.PaintColor(COLOR_RED);

        horizontalArrangement1 = new HorizontalArrangement( this );
        horizontalArrangement1.Width(LENGTH_FILL_PARENT);

        takePictureButton = new Button( this );
        takePictureButton.Text( "Take Picture" );

        camera1 = new Camera( this );

        redButton = new Button( horizontalArrangement1 );
        redButton.Text( "RED" );
        redButton.TextColor(COLOR_RED );

        blueButton = new Button( horizontalArrangement1 );
        blueButton.Text( "BLUE" );

        greenButton = new Button( horizontalArrangement1 );
        greenButton.Text( "GREEN" );

        bigButton = new Button( horizontalArrangement1 );
        bigButton.Text( "BIG" );

        smallButton = new Button( horizontalArrangement1 );
        smallButton.Text( "SMALL" );
```

- **Definir los event-response**

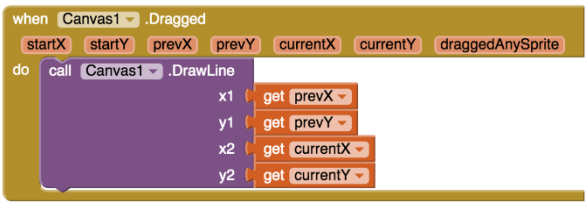
```
//Se pueden inicializar en $define o in los eventor de inicialización

        EventDispatcher.registerEventForDelegation( this, "canvas1", "Initialize"
);
        EventDispatcher.registerEventForDelegation( this, "canvas1", "Touched" );
        EventDispatcher.registerEventForDelegation( this, "canvas1", "Dragged" );
        EventDispatcher.registerEventForDelegation( this, "buttonClick", "Click"
);
        EventDispatcher.registerEventForDelegation( this, "camera1",
"AfterPicture" );
    }
```

- **Programar el evento Touched**



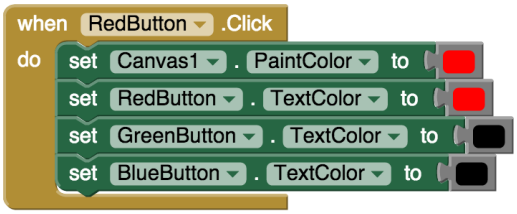

```
if( component.equals(canvas1) && eventName.equals("Touched") )
{
    //este evento tiene dos parámetros x,y, flotantes y un touchedSprite booleano
    //entoces el parametro[0] es x, el siguiente y y el último el touchedSprite

    canvas1Touched((Float) params[0], (Float) params[1], (Boolean) params[2]);
    return true;
}
public void canvas1Touched( Float x, Float y, Boolean touchedSprite )
{
    canvas1.DrawCircle( x.intValue(),y.intValue(), dotSize,true );

}
```

- **Programar el evento Dragged**




```
if( component.equals(canvas1) && eventName.equals("Dragged") )
{
    canvas1Dragged((Float) params[0], (Float) params[1], (Float) params[2], (Float)
params[3], (Float) params[4], (Float) params[5], (Boolean) params[6]);
    return true;
}
public void canvas1Dragged( Float startX, Float startY, Float prevX, Float prevY, Float
currentX, Float currentY, Boolean draggedSprite )
{
    canvas1.DrawLine( prevX.intValue(), prevY.intValue(), currentX.intValue(),
currentY.intValue() );
}
```

- **Programar el color de los botones**



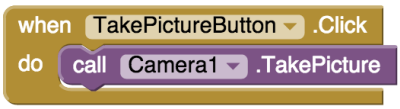

```
if( component.equals(redButton) && eventName.equals("Click") )
{
    redButtonClick();
    return true;
}
if( component.equals(blueButton) && eventName.equals("Click") )
{
    blueButtonClick();
    return true;
}
if( component.equals(greenButton) && eventName.equals("Click") )
{
    greenButtonClick();
    return true;
}
public void redButtonClick()
{

    canvas1.PaintColor(COLOR_RED);
    redButton.TextColor(COLOR_RED);
    greenButton.TextColor(COLOR_BLACK );
    blueButton.TextColor( COLOR_BLACK );
}
public void blueButtonClick()
{
    canvas1.PaintColor( COLOR_BLUE );
    blueButton.TextColor( COLOR_BLUE );
    greenButton.TextColor( COLOR_BLACK );
    redButton.TextColor( COLOR BLACK );
}
public void greenButtonClick()
{
    canvas1.PaintColor( COLOR_GREEN );
    greenButton.TextColor( COLOR_GREEN );
    blueButton.TextColor( COLOR_BLACK );
    redButton.TextColor( COLOR_BLACK );
}
```

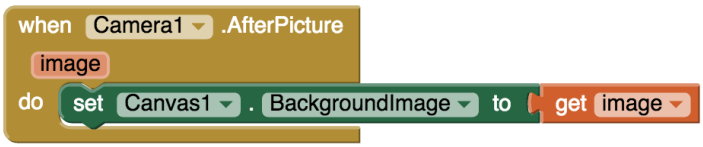
- **Programar el tamaño de los botones**


```
if( component.equals(bigButton) && eventName.equals("Click") )
{
    bigButtonClick();
    return true;
}
if( component.equals(smallButton) && eventName.equals("Click") )
{
    smallButtonClick();
    return true;
}
public void bigButtonClick()
{
    dotSize = 8;
}
```

```
public void smallButtonClick()
{
    dotSize = 2;
}
```

- **Programar el evento Tomar Foto**






```
if( component.equals(takePictureButton) && eventName.equals("Click") )
{
    takePictureButtonClick();
    return true;
}
if( component.equals(camera1) && eventName.equals("AfterPicture") )
{
    camera1AfterPicture((String) params[0]);
    return true;
}
return false;
public void takePictureButtonClick()
{
    camera1.TakePicture();
}
public void camera1AfterPicture( String image )
{
    canvas1.BackgroundImage( image );
}
```

¡Y ya tenemos nuestra app de App Inventor implementada en Android Studio!

Código Completo

Fuente: http://www.appinventor.org/content/java-bridge/introduction/paintpot