Pincipios de programación orientada a objetos

Introducción

La programación orientada a objetos entiende los programas como un conjunto organizado de objetos cooperativos. Cada objeto representa una instancia de una clase. Muchas de estas clases se relacionan mediante herencia o composición. Podemos decir que se focaliza la programación en el tratamiento de la información más que el la algorítmica y la información es parte privada de cada objeto. Solo un objeto sabe como operar con su propia información, lo cual protege la información de cambios indeseados.

Los principios que vamos a tratar son:

- Abstraction
- Encapsulation
- Inheritance
- Composition
- Modularity
- Polymophism

Abstraction

La abstracción denota las características esenciales de un objeto, que lo distinguen de otros objetos. Se centra en las características y operaciones esenciales, haciendo una interpretación codificada del problema que queremos resolver.

Por ejemplo, un coche puede ser visto como un conjunto que funciona. Su abstracción nos llevara a sus características (niveles de gasolina, neumáticos, velocidad, frenos) y a un conjunto de operaciones (acelerar, frenar, repostar). Las operaciones pueden afectar a características (acelerar

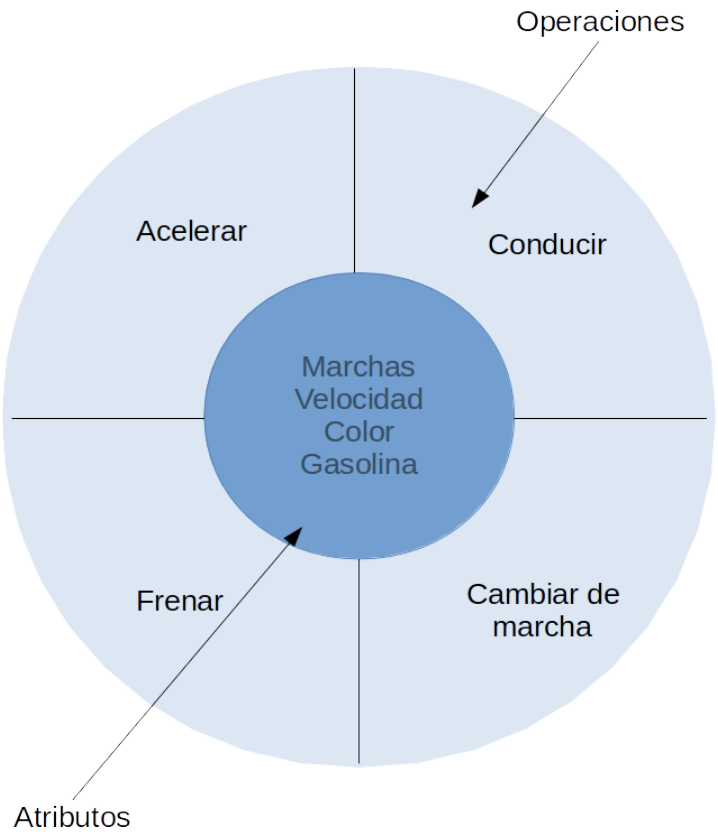
afectara al nivel de gasolina, a los neumáticos, la velocidad…).

Encapsulation

Consiste en separar la parte contractual de una abstracción y su implementación. Es decir, el proceso de compartimentar los elementos de una abstracción que constituye su estructura y comportamiento.

En la encapsulación, los elementos extraídos de la abstracción son compartimentados de tal manera que no todos serán accesibles des de fuera. Estos elementos pueden ser operaciones o atributos y estos son enlazados de tal manera que ciertos elementos no son accesibles des de fuera (normalmente atributos). De esta manera en vez de dar acceso a atributos directamente se hace des de operaciones donde se controla el acceso.

Cada objeto tiene su propio funcionamiento. Pero este no es visible a los usuarios. Tampoco es necesario y así evitamos problemas (posibles accesos a información sensible). Esto produce que los usuarios no entren en contacto con la implementación, esto permite modificarla sin ningún problema siempre y cuando se mantengan las operaciones intactas.

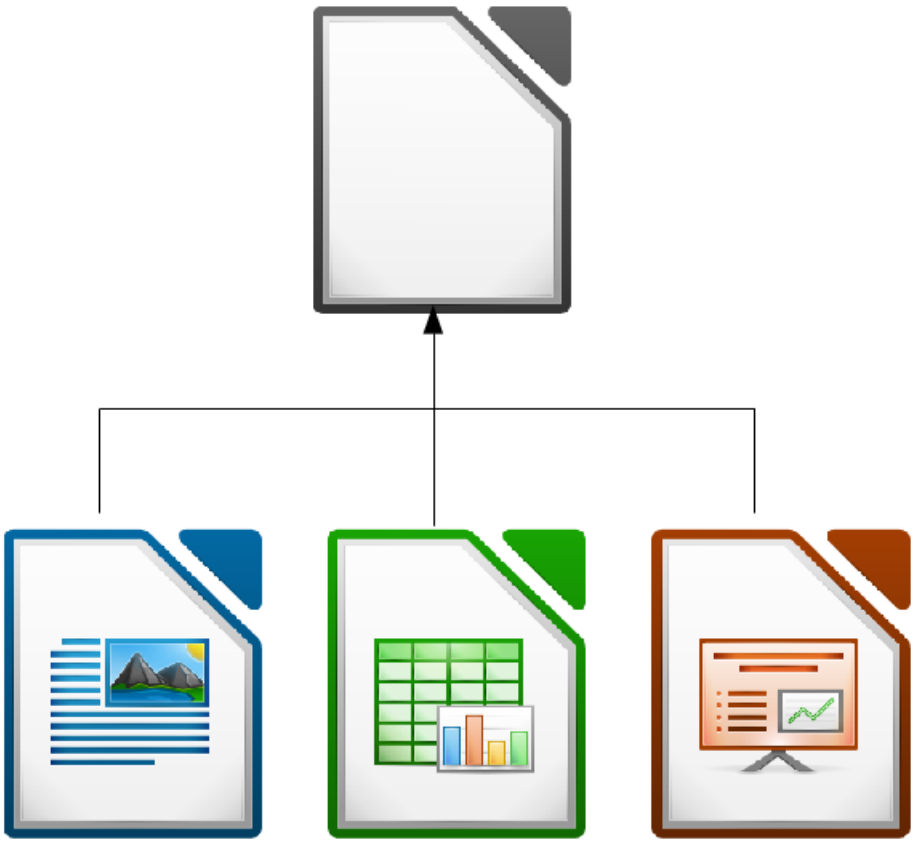


Inheritance

La herencia es la propiedad a través de la cual un objeto «hereda» las funcionalidades y estructura de otro.

Ciertos objetos tiene una misma estructura. En estos casos en vez de implementar esta misma estructura repetidamente, se puede representar en un solo objetos y los demás objetos puede «heredarlo» . Esto es posible cuando se ha creado una relación entre clases. Cuando se crean objetos de la clase el objeto hijo hereda las característica y funcionalidades del objeto padre.

En el siguiente esquema tenemos una clase documento. La clase documento contendrá la cabecera de los documentos y diferentes propiedades comunes de texto. Los documentos de texto, presentación y de hoja de calculo heredarán estas propiedades. De esta manera evitaremos repetir código inútilmente y lo reutilizaremos. Este tipo de relación de herencia se conoce como «is a».



Composition

La composición es una propiedad que permite a un objeto ser compuesto dentro de otro.

La composición facilita la reutilización de código. Se logra mediante la composición de un objeto dentro de otro. Por ejemplo, un objeto que representa un coche va a se compuesto por un objeto motor. Esta relación da lugar a objetos complejos y es vital para aplicaciones de gran tamaño.

Este tipo de relación se conoce como «has a» o «part of». Y existen dos tipos:

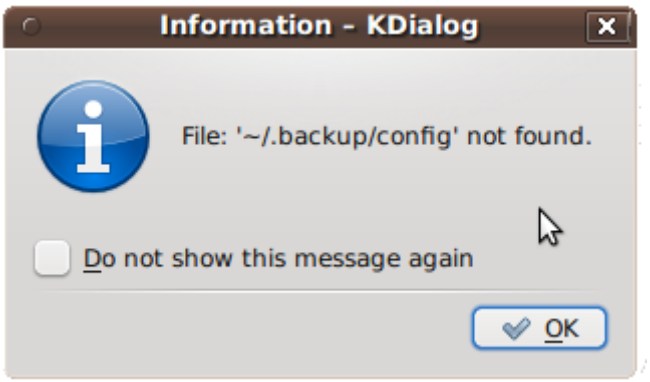
1. Aggregation

Se refiere a una relación débil entre el objeto exterior y el objeto interior. Donde este no depende de la vida útil del objeto exterior. En cuanto a código se refiere esto implica que el objeto exterior debe tener una referencia o puntero al

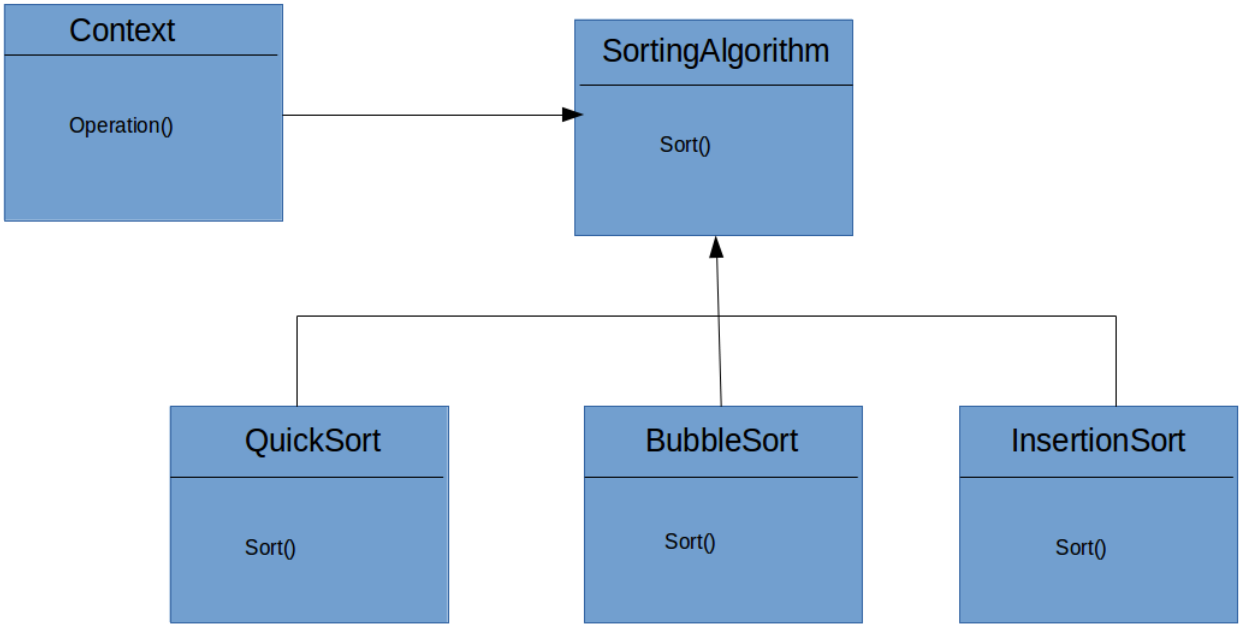
objeto interior. Por ejemplo, una ventana puede utilizar el cursor, pero cuando la ventana se destruye el cursor no.

1. Containment

Se refiere a una relación fuerte entre el objeto exterior y el objeto interior. En este tipo de relación (por ejemplo, en C++ una clase contiene el objeto de otra), la destrucción del objeto exterior implica la destrucción del objeto interior. Por ejemplo, cuando se destruye la ventana todos los controles de que dispone son destruidos (botones, label, edit boxes…)



A diferencia de la herencia, la composición es una relación dinámica porque se crea en tiempo de ejecución. Esto se consigue mediante punteros y referencias. En la figura siguiente, es posible para la clase *Context* utilizar cualquier tipo de algoritmo de ordenación en tiempo de ejecución.



Dada la naturaleza dinámica de la composición es mas flexible que la herencia. Las relaciones entre objetos se pueden crear o romper en tiempo de ejecución.

Modularity

Normalmente, en especial en grandes aplicaciones, tenemos que trabajar con un gran nombre de clases y otros útiles como

parte del código fuente. Desarrollar el código fuente como una única unidad es una tarea complicada i difícil.

por lo tanto, para reducir la complejidad, el código fuente es dividido en módulos pequeños e independientes. Normalmente estos son agrupados de forma lógica (conjuntos de procedimiento o bien la información con la que trabajan), así mantenemos grupos de código ordenados y estructurados.

El objetivo de la modularidad es descomponer el código en pequeñas unidades que faciliten el diseño, desarrollo y test de módulos individuales. Así podemos modificar un módulo sin tener conocimiento de los demás o bien que estos se vean afectados.

En cuanto a mantenimiento se refiere la modularidad es una gran herramienta. Sabes donde buscar exactamente cuando surge un error puesto que todo esta debidamente ordenado y compartimentado. Y una vez solucionado el problema no es necesario realizar tests de todo el código, solo de la parte afectada.

Polymophism

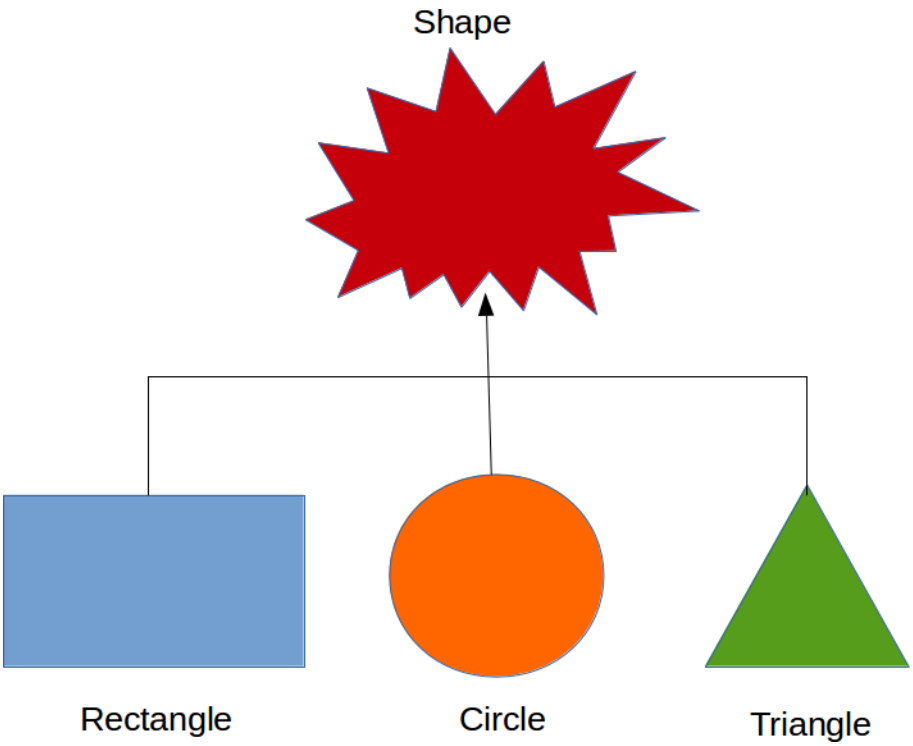
Es una característica de la programación orientada a objetos que denota la posibilidad de asignar un significado diferente para un símbolo particular o «operador» en diferentes contextos.

Un nombre puede referenciar a objetos de diferentes tipos que estn relacionados por características similares. Este nombre debe responder a un conjunto de operaciones y cada operación debe ejecutarse para el tipo adecuado.

En resumen el polimorfismo es la habilidad que tiene diferentes objetos para responder (a su manera) al mismo mensaje

En los lenguajes orientados a objetos el polimorfismo se consigue referenciando (o mediante punteros) al objeto de una subclase por el objeto padre. Las operaciones serán llamadas en la clase base del objeto, pero sern ejecutadas en el objeto correcto automaticamente

En este ejemplo el cliente llama a la función draw() en el objeto padre. La figura correcta se dibuja automáticamente en base a lo que el objeto principal se referencia (podría ser cualquier subclase de Shape)



Hay dos tipos de polimorfismo:

- Static (tiempo de compilación)
- Dynamic (run time)

El **polimorfismo estático** denota que la información requerida esta disponible en tiempo de ejecución. Por lo tanto, las llamadas funciones se pueden resolver en tiempo de compilación. La función exacta a llamar es determinada por el número de parámetros.

El polimorfismo estático se realiza mediante la sobre carga de funciones, operadores de sobrecarga o incluso templates (en c++). Siempre es mas rápido que el dinámico porqué el coste en tiempo de ejecución para resolver que función debe ser llamada se evita.

El **polimorfismo dinámico** denota que la información requerida para llamar la función no se conoce hasta que nos encontramos en tiempo de ejecución. esto se consigue mediante herencia o funciones virtuales.

Si una clase necesita redefinir una función en concreto definido en la clase base, el método es definido como virtual en la clase base y redefinido, para ajustarse a sus necesidades, en la clase derivada. La función virtual en la clase base básicamente define la interficie de la función. Cada clase derivada de ola clase base con las funciones virtuales puede redefinir las funciones con su propia implementación.

Una referencia a una clase base (puntero en C++) se puede usar para apuntar a un objeto de cualquier clase, el método es declarado como virtual en la clase base y redefinido según necesidad. En la clase base se define la interficie de la función. Cada clase derivada de la clase base con métodos virtuales puede redefinir los métodos con su propia implementación.

Dado que la llamada se resuelve en tiempo de ejecución, resultados polimorfismo dinámicos en poco más lenta ejecución del programa.

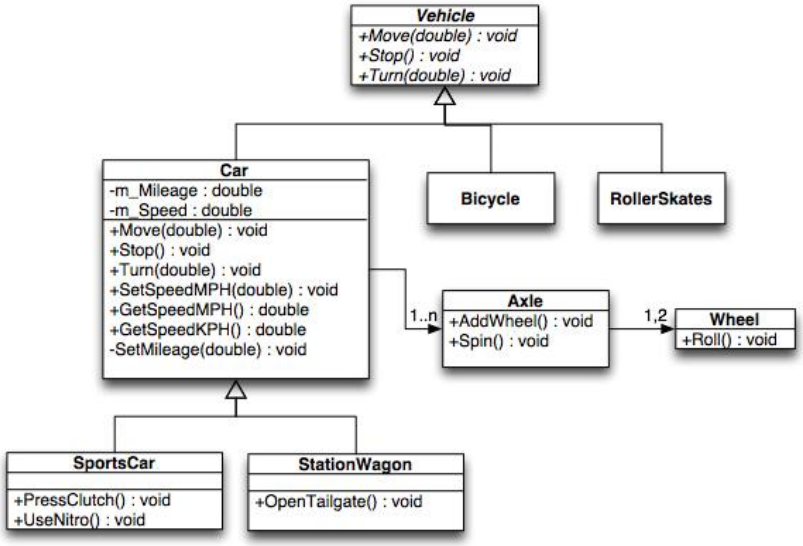
Observaciones

Espero que este post resulte útil. Es una primera aproximación

a la programación orientada a objetos. En los próximos días introduciremos los principios de diseño de clases. Cabe tener en cuenta que todo esto no es un dogma, mas bien una guía muy útil a la hora de programar.

Ruben.

Principios del diseño de clases



Introducción

Los principios de diseño nos ayudan a exprimir todo el potencial de la programación orientada a objetos. Permitiéndonos programar software flexible, extensible y escalable. Estos principios suelen conocerse con el acrónimo de SOLID, que se refiere a cada una de las iniciales de estos

principios:

- Single Responsabilty
- Open-Closed
- Liskov Substitution
- Inteface Segregation
- Dependency Inversión

Single Responsability

Nunca debe haber más de una razón para un cambiar una clase.

La responsabilidad de una clase es definida como una razón para cambiarla, porque una responsabilidad implica implementación. Y las implementaciones pueden cambiar en el futuro. Si una clase tiene más de una razón para cambiar entonces tiene más de una responsabilidad.

Vamos a ver la siguiente clase que representa el jugador de un equipo

```
public class Player{

        private String name;
        private int weight;
        private int height;
        private int position;

        public Player(String name, int weight, int height, int position){
      this.name = name;
      this.weight = weight;
      this.height = height;
      this.position = position;
        }

        public int calculateBMI(){
                //Implementation
        }
```

```
        public void SaveToFile(){
                //Implementation
        }

    public void LoadFromFile(){
                //Implementation
        }
}
```

Como podemos observar la clase tiene la responsabilidad de trabajar con la información de un jugador. Pero ademas, también de guardar y cargar dicha información.

Como podemos observar las funciones de guardado y carga pretenden hacerlo des de un fichero. Si mas adelante tuviéramos que hacerlo des de una base de datos nos veriamos obligados a modificar el código, teniendo que retestear todo por la posible introducción de errores. Estos errores afectarían a la clase Player y a las que dependan de esta.

Resulta mucho mas util separar las responsabilidades en dos clases diferentes.

```
public class Player{

        private String name;
        private int weight;
        private int height;
        private int position;

        public Player(String name, int weight, int height, int
position){
      this.name = name;
      this.weight = weight;
      this.height = height;
      this.position = position;
        }

        public int calculateBMI(shapeType){
                //Implementation
        }
```

```
}

public class Db{

        private String playerInfo;

         public Db(String playerInfo, int weight, int height,
int position){
      this.playerInfo = playerInfo;
        }

        public void Save(){
                //Implementation
        }

    public void Load(){
                //Implementation
        }
}
```

Open-Closed Principle

Las entidades de software (clases, módulos, funciones…) deben estar preparadas para ser extensibles pero cerradas a modificaciones.

En el siguiente código podemos ver una clase que dibuja figuras y calcula su área:

```
public enum ShapeType {
    CIRCLE, SQUARE, TRIANGLE
}

public class ManageShape{

        private ShapeType shapeType;
    private int factorResize;

        public ManageShape(ShapeType shapeType){
      this.shapeType = shapeType;
```

```
        }

        void calculateArea(shapeType){
                switch(shapeType){
                        case shapeType.CIRCLE:
                                //implementation
                                break;
                        case shapeType.TRIANGLE:
                                //implementation
                                break;
                        case shapeType.SQUARE:
                                //implementation
                                break;
                        default:
                                //implementation
                                break;
                }
        }

        void drawShape(shapeType){
                switch(shapeType){
                        case shapeType.CIRCLE:
                                //implementation
                                break;
                        case shapeType.TRIANGLE:
                                //implementation
                                break;
                        case shapeType.SQUARE:
                                //implementation
                                break;
                        default:
                                //implementation
                                break;
                }
   }
}
```

Podemos encontrarnos con la necesidad de añadir mas figuras en el futuro. Estos cambios pueden introducir errores y causar varios fallos. Nuestro enum deber cambiar también y todos los módulos que depengan de la clase ManageShape con todo lo que

ello supone.

El principio «open-closed» nos sugiere programar de la siguiente manera:

- Los módulos están abiertos para su extensión. La funcionalidad puede cambiar añadiendo nuevo código.
- El código existente esta cerrado a modificación y rara vez se tiene que modificar el código existente para cambiar el comportamiento del módulo.

La programación orientada a objetos nos provee de herramientas para implementar el principio «open-close». Una opción is codificar nuestro programa utilizando herencia y polimorfismo. Llevado esto a nuestro código crearemos una estructura en la que cada figura heredara de ManageShape y cada subclase se encargara de mantener su funcionalidad y la enumeración no sera necesaria.

```
public abstract class ManageShape{

    private int factorResize;

        public abstract void calculateArea(){

        }

        public abstract void drawShape(){

   }
}

public class Triangle extends ManageShape{

        void calculateArea(shapeType){

        }

        void drawShape(shapeType){
```

```
   }
}

public class Circle extends ManageShape{

        void calculateArea(shapeType){

        }

        void drawShape(shapeType){

   }
}

public class Square extends ManageShape{

        void calculateArea(shapeType){

        }

        void drawShape(shapeType){

   }
}
```

De esta manera los módulos que necesiten utilizar una figura dependerán de la clase abstracta ManageShape y utilizaran las subclsases o punteros. Si se añaden mas clases de tipo figura el código existente no se tendrá que modificar, no habrá impacto en el resto del código. Otra ventaja es que el código común esta contenido en la clase base y así se evitan duplicidades.

Liskov Substitution

Las funciones que utilizan punteros o referencias a una clases base tienen que poder utilizar objetos de las clases derivadas sin tener conocimiento de ello. Es decir, una clase debe funcionar correctamente cuando se utiliza su sublcase en vez

de la clase padre.

Cojamos el siguiente código de ejemplo:

```
class Bird {
        public void fly(){}
        public void eat(){}
}

class Crow extends Bird {}

class Ostrich extends Bird{
}
```

Como podemos obervars la classe Birs tiene los metodos fly() y eat(). Nos encontramos que al crear la clase Ostrich nos sobra el método fly(). Y aquí nos encontramos con el problema al utilizar la clase Ostrich (subclase de Bird) nos encontramos con un método que debería volver una excepció del tipo UnsupportedOperationException().

Una solución sería la siguiente:

```
class Bird{
        void eat();
}

class Crow extends Bird{
        public void fly(){
        }
        @Override
        public void eat() {
                // TODO Auto-generated method stub

        }
}

class Ostrich extends Bird{

        @Override
        public void eat() {
                // TODO Auto-generated method stub
```

```
        }
}
```

Una la clase Bird contiene solo el método eat(), el cual es comun en todas la aves. El método fly() el cual se implementará en la subclase y a su vez Ostrich.

Inteface Segregation

Los clientes no se deben ver forzados a depender de métodos que no utilizan.

Este principio trata de lidiar con interfícies no cohesionadas y excesivamente generalistas.Esto produce que los clientes se vean forzados a depender de métodos que jamas van a utilizar.

Por ejemplo Vamos a poner el caso de una interficie IFile utilizada para operaciones de ficheros. Dos clases heredan de ella TextFile y SocketFile

```
interface IFile {
        void Open();
        void Close();
        void Read();
        void Write();
        void Seek();
        void Position();
}

class TextFile implements IFile {

        public void Open(){
                //Implementation
        }

        public void Close(){
                //Implementation
        }

        public void Read(){
```

```
                //Implementation
        }

        public void Write(){
                //Implementation
        }

        public void Seek(){
                //Implementation
        }

        public void Position(){
                //Implementation
        }

}

class SocketFile implements IFile {

        public void Open(){
                //Implementation
        }

        public void Close(){
                //Implementation
        }

        public void Read(){
                //Implementation
        }

        public void Write(){
                //Implementation
        }

        public void Seek(){
                throw new NotImplementedException();
        }

        public void Position(){
                throw new NotImplementedException();
```

```
        }
}
```

Los métodos seek() y position() no son útiles para SocketFile. The echo está clase no puede implementarlos y por ello se lanza una excepción de tipo NotImplementedException(). Una solución que para nada resulta pulida. La clase puede acceder a los métodos pero para asegurarnos que no va haber ningún problema tenemos que lanzar una excepción.

El principio de segregación de interficies sugiere que una interficie no cohesionadas se deben dividir en interficies pequeñas, para que cada una de estas de servicio al cliente correspondiente.

Aplicando esto a nuestro caso, la interficie IFile se verá dividida en dos interficies

```
interface IFile {
        void Open();
        void Close();
        void Read();
        void Write();
}

interface IDiskFile implements IFile{
        void Seek();
        void Position();
}
```

La interficie IDiskFile será implementada por la clase TextFile, la cual implementara los metodos de las dos interficies (vease que IDiskFile implementa IFile). I SocketFile implementara solo la interficie IFile y no tendrá que preocuparse de métodos que no tulilizará (seek() y position()).

Dependency Inversión

Los módulos de alto nivel no deben depender de los módulos de bajo nivel. Y ambos deben depender de la abstracción. La abstracción no debe depender de los detalles. Los detalles deberían depender de las abstracciones

En los diseños de análisis estructurado, el software que se crea tiende a tener una dependencia en la dirección equivocada. Las reglas de negocio dependen de los detalles de implementación de bajo nivel. Estos causa problemas cuando las reglas de negocio cambian y la implementación de bajo nivel se tiene que modificar. Sino que más bien debería ser al revés; la implementación de bajo nivel debe depender de las reglas de negocio. Por lo tanto, la dependencia tiene que ser invertida; de ahí el nombre de principio de dependencia invertida.

Consideremos el ejemplo de un botón que enciende y apaga una lampara

```
class Lamp {

        public void turnOn(){
                //Implementation
        }

        public void turnOff(){
                //Implementation
        }

}

class Button {
        private Lamp lamp = new Lamp();
        private Boolean pressed = false;

        public void pressButton(){
                if(pressed){
                        lamp.turnOff();
                }else{
```

```
                        lamp.turnOn();
                        pressed = !pressed;
                }
        }
}
```

la clase Lamp tiene la implementación de las funcionalidades de la lampara. Està controlada por la clase Button que llama a los métodos turnOn() y turnOff(). Esto es un ejemplo de un diseño estructurado. La logica de control esta en Button y la implementación de bajo nivel en Lamp. De esta manera la clase Button contiene la implementación de alto nivel (logica) y Lamp contiene la implementación de bajo nivel.

Es bastante obvio que la dirección de la dependencia es de Button a Lamp. Si la implementación de Lamp se tiene que modificar, también afectará a la clase Button. Esto significa que la lógica de negocio se ve afectada por la implementación de bajo nivel. Este ejemplo muestra un código muy pequeño pero en una aplicación grande puede generar una modificación en cascada de las clases de alto nivel.

Idealmente, las lógica de negocio define cómo se debe hacer la implementación de bajo nivel. Así, los módulos de bajo nivel idealmente dependen de la lógica de negocio y operan de acuerdo a ellos.Dado esto debemos invertir la dependencia de la clase Button en la clase Lamp.

La solución es la interficie Switchable que contiene el protocolo que todos los objetos deben seguir para controlar la clase Button. La clase Button no interactuará con Switchable en vez de con la clase Lamp.

```
interface Switchable {
        void turnOn();
        void turnOff();
}
```

```
class Lamp implements Switchable{

        public void turnOn(){
                //Implementation
        }

        public void turnOff(){
                //Implementation
        }
}

class Television implements Switchable{

        public void turnOn(){
                //Implementation
        }

        public void turnOff(){
                //Implementation
        }
}

class Button {
        private Switchable switchable = new Lamp();
        private Boolean pressed = false;

        public Button(Switchable s){
                this.switchable = s;
        }

        public void pressButton(){
                if(pressed){
                        this.switchable.turnOff();
                }else{
                        this.switchable.turnOn();
                        pressed = !pressed;
                }
        }
}
```

Ahora, es posible añadir mas objetos que se puedan encender y apagar a traves de Button sin que este tenga que ser

modificado. La dependencia se ha invertido, los módulos de bajo nivel dependen de los módulos de alto nivel. Esto crea una estructura flexible facil de mantener y a la cual se pueden añadir nuevas funcionalidades

Observaciones

En esta segunda aproximación hemos definido los principios del diseño de clases. Este conjunto de reglas nos ayudará a mantener un código limpio, ordenado, fácil de mantener y robusto. Todo ello enfocado a facilitar la nueva entrada de funcionalidades a nuestra aplicación y la corrección de errores.

Ruben.

Tutorial Threejs – Parte I , Introducción

Hola a todos.

Vamos a comenzar una serie de posts dedicados a Threejs , que espero que sea de vuestro agrado.

¿Que es Threejs?

Threejs es motor 3D ligero para javascript .

Funciona muy bien bajo webGL , aprovechando la aceleración de nuestra tarjeta gráfica , dando unos resultados

espectaculares.

Podemos encontrar unos ejemplos de lo que se puede hacer en la página de ejemplos oficial de Threejs.

Ésta serie de artículos pretende explicar el funcionamiento de Threejs , que hacer , como empezar , como crear escenas , importar objetos , etc , etc.

Empezaremos explicando un poco que es necesario para empezar:

Un navegador con soporte webGL , actualmente los mejores resultados los da Google Chrome , aunque se puede utilizar Mozilla Firefox , aunque es más pesado y por lo tanto un poco más lento , no olvidemos que Chrome utiliza el motor webkit que utilizan también muchos smartphones , mientras que Mozilla utiliza el motor Gecko , basado en java .

Conocimientos de programación , javascript , y html5 .

Nociones de 3D , para no perderse , o mucha paciencia para absorver conceptos nuevos.

Descargar la librería Threejs con los ejemplos y herramientas que iremos utilizando y explicando.

Que podemos hacer !

Con Threejs podemos hacer casi cualquier cosa que podemos hacer con un motor 3D , vamos a explicarlo para que sirva también como tutorial de iniciación al 3D.

Básicamente consiste en crear una escena , con objetos dentro que visualizaremos en la pantalla , en nuestro caso en un canvas de html5 , a través de cámaras e iluminados por luces .

La escena

Es la representación del mundo virtual tridimensional , donde se irán colocando los demás elementos , es algo así como un contenedor del espacio en 3D , que empieza estando vacío .

Este espacio está vacío , no tiene nada , iremos colocando por turno los objetos que deseamos visualizar , basándonos en que la escena se pude medir , con eje de coordenadas virtual a partir del cual iremos posicionando los objetos en las 3 dimensiones básicas .

Gracias a esta referencia dentro de la escena podemos rotar los objetos , desplazarlos , etc , etc.

La escena tiene un fondo que se puede personalizar , que sería un poco así lo que sería el horizonte , se pude dejar un color plano , o una imagen , depende de lo que se quiera hacer.

La cámara

La cámara nos permite visualizar la escena desde un punto de vista concreto .
Se comporta como un objeto más , se puede rotar , desplazar , pero además tiene algunas propiedades más , propias de una cámara .

Podemos tener varias cámaras y utilizar una u otra según convenga.

Las luces

Las luces son objetos que irradian luz y nos permiten visualizar los objetos gracias a la reflexión de la luz en ellos.

Hay varios tipos de luz , de punto (Point), solar (sun), foco (Spot) , en otras aplicaciones podemos encontrar más tipos de luz.

La intensidad y el color de las luces son configurables y nos permiten una multitud de posibilidades y pueden provocar sombras sobre los objetos de la escena.

Cada objeto reacciona diferente a la luz según sus valores para la luz de difusión (color reflejado) , especular (color

refractado) y emisión (color emitido por ejemplo una bombilla) , variando según el color e intensidad.

Para una iluminación correcta es necesario el uso de normales , que son el cálculo de la perpendicular de cada cara visible del objeto .

Para empezar , la mayoría de aplicaciones comienzan con un escena que contiene una cámara y una luz , en muchos casos incluye un objeto en modo de ejemplo que suele ser un cubo o una esfera .

Los objetos

Los objetos son una representación de arrays de vértices que forman líneas y caras.
Jeje que fácil no ,

Los vértices son puntos que ocupan unas coordenadas .
La unión de 2 vértices forma líneas.
La unión de varias líneas forman planos que se suelen llamar caras.

Las caras se suelen formar por 3 o 4 líneas que le dan forma , y obviamente una cara/plano de 4 líneas se puede representar como 2 caras/planos de 3 líneas , así que lo más normal es encontrarnos con modelos de objetos formados por arrays de vértices que representan estos triángulos .

En la mayoría de aplicaciones para modelar se especifica al exportar si se quieren crear triángulos (3 líneas) en aquellas caras que son romboides (4 líneas) , para su uso posterior.

Este array de vértices que comentamos no es más que la forma del objeto , hay que posicionar el objeto en la escena .

Los objetos están hechos con uno o varios materiales a los que hay que asignarle valores según tenga que reaccionar a la luz un color de difusión , otro de especular y el color de emisión si emite luz.

Cada material puede tener una textura , que puede ser una imagen , hasta en ocasiones un vídeo.

Las texturas dan el toque de calidad a los objetos , siendo unos de los temas más importantes del desarrollo en 3d ,por ejemplo podemos aplicar transparencias gracias a las texturas , o de un plano básico aplicarle una textura para que parezca una casa.

Los objetos pueden ser de formas básicas como un cubo o una esfera , o puede ser una figura compleja , pero en el caso de ser una figura compleja mejor crearla con un editor externo (no con código me refiero) , e importarlo como un modelo.

Los modelos

Los modelos 3D son objetos pre-diseñados que se utilizan para incorporar en la escena. Pueden incluir sus propias texturas , y estar formados por uno o varios objetos , a su vez con sus propias texturas.

Los modelos se comportan como una unidad , y se ubican a partir de un punto que es su centro de coordenadas , que se suele ubicar abajo , pero puede variar según el creador.

Podemos diseñar objetos en aplicaciones como Blender , con la comodidad que ello aporta , y exportarlos para el uso en nuestras aplicaciones , Threejs nos aporta herramientas para convertir objetos en formato obj en objetos en formato compatible para Threejs en json.

Los modelos pueden tener animaciones programadas preparadas para nuestro uso , por ejemplo ficheros md2 con acciones como saltar , correr , cargar , etc .

Threejs nos aporta una herramienta para crear modelos en formato Threejs , muy útil y práctica.

Resumiendo un poco , ya sabemos que tenemos que crear una escena , añadirle una cámara , una luz , y los objetos y

modelos que queramos.

Bueno hasta aquí la introducción , en el próximo artículo pasaremos a la acción creando la primera escena .

Saludos ,
Scuraki

Tutorial Arduino parte 4

Hola a todos .

Este es el cuarto tutorial sobre Arduino , en este tutorial vamos a continuar el tutorial 3 , creando una aplicación 3d para Android , que encienda la bombilla de forma remota .

Creamos un proyecto Android nuevo en eclipse con Android SDK.

Continuando como teníamos en el tutorial 3 , en la parte del servidor php , el fichero que recibía una variable GET y encendía o apagaba la bombilla si la variable es high o low.

En este primer código podemos ver como desde el proyecto Android podemos realizar la conexión con el servidor PHP para que envíe la señal a la placa Arduino.

El fichero PHP ha de estar en modo servicio , con las lineas //die… descomentadas , para que nos devuelva una respuesta.

```
private void lighting()
{
ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
if (cm != null) {
boolean connected=false;
```

```
NetworkInfo[] info = cm.getAllNetworkInfo();
if (info != null) {
for (int i = 0; i < info.length; i++) { if (info[i].getState()
== NetworkInfo.State.CONNECTED) { connected = true; } } }
if(connected) { String
destiny="http://192.168.0.131/arduino/service.php?check=on";
String missatxe="Signal CHECK sended. Click again to switch
ON"; cambiarTextoBoton("Switch ON"); switch(estado) { case 0:
destiny ="http://192.168.0.131/arduino/service.php?hight=on";
missatxe="Signal ON sended. Click again to switch OFF";
cambiarTextoBoton("Switch OFF"); break; case 1: destiny
="http://192.168.0.131/arduino/service.php?low=on";
missatxe="Signal OFF sended. Click again to switch ON";
cambiarTextoBoton("Switch ON"); break; } escribir(missatxe);
InputStream is = null; //initialize String result = ""; try{
HttpClient httpclient = new DefaultHttpClient(); HttpPost
httppost = new HttpPost(destiny); HttpResponse response =
httpclient.execute(httppost); HttpEntity entity =
response.getEntity(); is = entity.getContent(); //convert
response to string try{ BufferedReader reader = new
BufferedReader(new InputStreamReader(is,"UTF-8"),8);
StringBuilder sb = new StringBuilder(); String line = null;
while ((line = reader.readLine()) != null) { sb.append(line +
"\n"); } is.close(); result=sb.toString(); ArrayList
message=this.parse( result);
String status=message.get(0).toString();
String signal=message.get(1).toString();
String mensage=message.get(2).toString();
if(signal.compareTo("1")==0)
{
checkOK=1;
estado=0;
}
else
{
if(signal.compareTo("2")==0)
{
```

```
checkOK=2;
estado=1;
}
else
{
if(signal.compareTo("-1")==0)
{
checkOK=-1;
estado=-1;
}
}

}
}catch(Exception e){
Log.e("log_tag", "Error converting result "+e.toString());
escribir("Error converting result "+e.toString());
}
}catch(Exception e){
Log.e("log_tag", "Error in http connection "+e.toString());
escribir("Error in http connection "+e.toString());
}
}
}

}
```

Bien , ya podemos comunicarnos con el servidor , pero bueno un poco rancia la aplicación , no?

¿Falta un botón , un poco de color o algo no?

Puesto a investigar un poco pués encontré una librería que permitía crear aplicaciones OPENGL de manera un poco más sencilla de lo normal , con un lenguage más para un principiante como yo que ir directamente a OPENGL de Android , que es un poco más rudo para alguien que no lo usa muy a menudo.

Esa librería es Min3D.

Min3D nos permite crear escenas 3D e importar objetos 3DS o OBJ , así como texturas y objetos básicos como esferas y cubos.

Tras descargar min3d de su repositorio , incluimos la carpeta al proyecto y podemos ver y probar los ejemplos.

Partiendo del ejemplo ExampleLoadObjFileMultiple.java , donde podemos ver como cargar un objeto desde un fichero .OBJ .

En este fichero podemos ver que hereda de la clase extends RendererActivity , y lo aplicaremos a la clase que estemos utilizando .

El método initScene() es el encargado de crear la escena , en este método es donde colocaremos todos los objetos de la escena , cargaremos todos los ficheros necesarios , .OBJ , .PNG , y crearemos las esferas y rectángulos que necesitemos.

Este método es llamado al iniciar la aplicación y cada vez que el dispositivo entra en modo PAUSE , o se bloquea la pantalla.

Como en toda aplicación 3D , además de una escena necesitamos una cámara y luces para iluminar nuestros objetos.

En este método asignaremos un color de fondo , o una textura , situaremos las luces , los objetos y situaremos y enfocaremos la cámara , para obtener la perspectiva adecuada.

El método updateScene() se ejecuta en cada frame , sería el equivalente al típico evento draw() de repintado de la pantalla en aplicaciones 2D , osea que se ejecuta cada vez que se pinta la pantalla.

Éste método es el encargado de las instrucciones de las animaciones , por ejemplo si queremos rotar un objeto , en este método calculamos el valor nuevo de la rotación y se le asigna.

Para cargar los ficheros .OBJ en la aplicación Android es

necesario modificar los nombres de los archivos.

Los ficheros .OBJ suelen ir acompañados de un fichero con el mismo nombre con extensión .mtl con la definición de los materiales de los grupos de los objetos que hayan definidos en el fichero .OBJ , además de las imágenes de las texturas.

Osea que el fichero .OBJ tiene objetos , valores de vértices y objetos y el fichero .mtl los materiales.

Las imágenes de las texturas las colocaremos en la carpeta res/drawable… .

Los ficheros .OBJ y .MTL los renombramos sustituyendo el punto por un «_» osea un guión bajo , para evitar los conocidos conflictos de ficheros con el mismo nombre y diferente extensión que tiene la programación con Android.

Para este tutorial hemos creado una farola , con una esfera que hace de bombilla , y una esfera con una textura con transparencia .PNG , que hará el efecto de que la farola está encendida.

También se ha creado una especie de bullofa que emite unas esfera con transparencia , emulando el envío de ondas , dando a entender que se está comunicando con el servidor , una esfera nos indicará según la textura que tenga el texto ON de color verde , o el texto OFF de color roja o de color ambar si no hay conexión con el servidor ,a además hay otra bola que nos indicará según su textura , si estamos a más de cierta distancia de la bombilla , si hay cobertura GPS .

Bueno abrimos blender y creamos una lámpara y una bullofa (algo que de el efecto de ser un emisor de ondas) , y las exportamos en formato wavefront .OBJ .

Si no queremos utilizar blender o otro software para crear objetos 3D , los podemos descargar de alguna página , por ejemplo ésta , que ofrezcan objetos 3D en formato .OBJ .

Podemos ver como asignar el color de fondo:

```
scene.backgroundColor().setAll(0xfff2d533);
```

Asignar una textura a un rectángulo:

```
Bitmap b = Utils.makeBitmapFromResourceId(this, R.drawable.scuraki);
float w = 20f;
float h = w * (float)b.getHeight() / (float)b.getWidth();
Rectangle suelo = new Rectangle(w, h, 1,1, new Color4());
suelo.doubleSidedEnabled(true); // ... so that the back of the plane is visible
suelo.normalsEnabled(false);
scene.addChild(suelo);

Shared.textureManager().addTextureId(b, "scuraki", false);
suelo.textures().addById("scuraki");
```

Asignar una textura a una esfera:

```
b = Utils.makeBitmapFromResourceId(R.drawable.bolaroja);
Shared.textureManager().addTextureId(b, "bolaroja", false);
bolarojaTexture = new TextureVo("bolaroja");
esfera.textures().addReplace(bolarojaTexture);
```

Iluminar la escena:

```
luzobj = new Light();
luzobj.ambient.setAll(new Color4 (128,128,128,128));
luzobj.diffuse.setAll(new Color4 (64,64,164, 128));
luzobj.emissive.setAll(new Color4 (0,0,0,255));
luzobj.specular.setAll(new Color4 (0,0,0,255));
luzobj.type(LightType.POSITIONAL);
scene.lights().add(luzobj);
luzobj.position.setAll(0.65f, -0.85f, 3.5f);
```

Añadir un objeto .OBJ a la escena:

```
parser2 = Parser.createParser(Parser.Type.OBJ, getResources(), "adictosalainformatica.min3DAdictos:raw/fanal_obj", true);

parser2.parse();
```

```
fanal = parser2.getParsedObject();
fanal.scale().y = 0.25f;
fanal.scale().z = 0.25f;
fanal.scale().x = 0.25f;
fanal.shadeModel(ShadeModel.SMOOTH);
fanal.vertexColorsEnabled(true);
fanal.normalsEnabled(true);
fanal.colorMaterialEnabled(false);

scene.addChild(fanal);
fanal.position().x=0.0f;
fanal.position().y=1.6f;
fanal.position().z=-5f;
fanal.rotation().x=25f;
```

Controlar la rotación de los objetos en el método updateScene:

```
bombilla.rotation().y=yrot;
bombilla.rotation().x=xrot;
receptor.rotation().y=yrot;
receptor.rotation().x=xrot;

xrot += xspeed;
yrot += yspeed;
_count++;
```

Para dar un toque de color la bullofa cambia el valor de escalado cada cierto tiempo , y se crean unas esferas con transparencia , que viajan desde la bullofa (que está cerca de la cámara) hasta la lámpara (que está al fondo de la escena), emulando el envio de ondas.

Un array de esferas ameniza la pantalla , desplazandose .

La clase Burbuja es la encargada de controlar estas esferas:

```
package adictosalainformatica.min3DAdictos;

import android.graphics.Bitmap;
import min3d.Shared;
import min3d.Utils;
```

```
import min3d.core.Scene;
import min3d.objectPrimitives.Sphere;
import min3d.vos.Color4;
import min3d.vos.TextureVo;
public class Burbuja {
private float velocidad_x=.000f;
private float velocidad_y=0.02f;
private float velocidad_z=0.04f;
private float posicion_x=0.45f;
private float posicion_y=-0.25f;
private float posicion_z=0.3f;
private float variacion_x=0.45f;
private float variacion_y=-0.25f;
private float variacion_z=0.6f;
private float limite_x=4.01f;
private float limite_y=4.01f;
private float limite_z=4.8f;
private Color4 color=null;
private Sphere esfera =null;
long _index;
private float contador=0;
public Burbuja(long index)
{
this._index=index;
}
public void make(Scene scene )
{
if(esfera!=null)
{
esfera.clear();
}
esfera=null;
TextureVo textura=null;
esfera = new Sphere(1.5f, 20,20);
esfera.scale().x = esfera.scale().y = esfera.scale().z = .1f;
esfera.position().setAll(posicion_x, posicion_y, posicion_z);
Bitmap                          b                                =
```

```
Utils.makeBitmapFromResourceId(R.drawable.tscuraki);
Shared.textureManager().addTextureId(b, "burbuja0" + _index,
false);
b.recycle();
textura = new TextureVo("burbuja0" + _index);
esfera.textures().addReplace(textura);
esfera.colorMaterialEnabled(false);
esfera.vertexColorsEnabled(false);
esfera.lightingEnabled();
scene.addChild(esfera);

//Log.v(Min3d.TAG, "ReCrea Burbuja=" + _index);

}

public int mover()
{
int moviendo=1;
posicion_x=variacion_x + ( contador * velocidad_x);
posicion_y=variacion_y + ( contador * velocidad_y);
posicion_z=variacion_z - ( contador * velocidad_z);
esfera.position().setAll(posicion_x, posicion_y, posicion_z);
if((posicion_x>limite_x)||(posicion_y>limite_y)||(posicion_z *
-1 > limite_z ))
{
esfera.clear();

moviendo=0;
}
//Log.v(Min3d.TAG, "Mueve Burbuja=" + _index + " posiciónx="+
posicion_x + " posicióny=" + posicion_y + " posiciónz=" +
posicion_z);
esfera.rotation().x=contador * -1.3f ;
esfera.rotation().y=contador * -1.3f;
esfera.rotation().z=contador * -1.3f;
esfera.scale().x = esfera.scale().y = esfera.scale().z = .1f +
(contador * 0.001f);
contador++;
```

```
return moviendo;

}

public float getLimite_x() {
return limite_x;
}
public void setLimite_x(float limite_x) {
this.limite_x = limite_x;
}
public float getLimite_y() {
return limite_y;
}
public void setLimite_y(float limite_y) {
this.limite_y = limite_y;
}
public float getLimite_z() {
return limite_z;
}
public void setLimite_z(float limite_z) {
this.limite_z = limite_z;
}
public Color4 getColor() {
return color;
}
public void setColor(Color4 color) {
this.color = color;
}
public float getVelocidad_x() {
return velocidad_x;
}
public void setVelocidad_x(float velocidad_x) {
this.velocidad_x = velocidad_x;
}
public float getVelocidad_y() {
return velocidad_y;
}
```

```
public void setVelocidad_y(float velocidad_y) {
this.velocidad_y = velocidad_y;
}
public float getVelocidad_z() {
return velocidad_z;
}
public void setVelocidad_z(float velocidad_z) {
this.velocidad_z = velocidad_z;
}
public float getPosicion_x() {
return posicion_x;
}
public void setPosicion_x(float posicion_x) {
this.posicion_x = posicion_x;
}
public float getPosicion_y() {
return posicion_y;
}
public void setPosicion_y(float posicion_y) {
this.posicion_y = posicion_y;
}
public float getPosicion_z() {
return posicion_z;
}
public void setPosicion_z(float posicion_z) {
this.posicion_z = posicion_z;
}
public float getVariacion_x() {
return variacion_x;
}
public void setVariacion_x(float variacion_x) {
this.variacion_x = variacion_x;
}
public float getVariacion_y() {
return variacion_y;
}
public void setVariacion_y(float variacion_y) {
```

```
this.variacion_y = variacion_y;
}
public float getVariacion_z() {
return variacion_z;
}
public void setVariacion_z(float variacion_z) {
this.variacion_z = variacion_z;
}
public Sphere getEsfera() {
return esfera;
}
public void setEsfera(Sphere esfera) {
this.esfera = esfera;
}
public long getIndex() {
return _index;
}
public void setIndex(long _index) {
this._index = _index;
}

}
```

Finalmente controlamos la distancia a la bombilla gracias al sensor GPS , activándolo si es necesario.

Podemos decidir activar la bombilla a cierta distancia , útil para luces de garages , por ejemplo que se encienda cuando falta 2 kilómetros para llegar con el coche.

En nuestro caso tenemos la esfera que nos indica la distancia , modificamos la textura para que nos muestre FAR si está lejos y NEAR si está cerca.

```
private void loadJipiEs()
{
// Acquire a reference to the system Location Manager
```

```
this.locationManager = (LocationManager)
this.getSystemService(Context.LOCATION_SERVICE);

if
(!this.locationManager.isProviderEnabled(LocationManager.GPS_P
ROVIDER))
{
createGpsDisabledAlert();
}
else
{
// List all providers:
List providers = this.locationManager.getAllProviders();

Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);

this.bestProvider =
this.locationManager.getBestProvider(criteria, false);

Location mylocation =
this.locationManager.getLastKnownLocation(this.bestProvider);

if(mylocation!=null)
{
this.getLocation(mylocation);
}
else
{
//this.afegir("Posición inicial vacía.");
}
}

}
private void getLocation(Location location)
{
if(location!=null)
{
```

```
boolean hasAltitude=false;
boolean hasAccuracy=false;
boolean hasBearing=false;
lat=location.getLatitude();
lon=location.getLongitude();
alt=location.getAltitude();

hasAltitude=location.hasAltitude();
hasAccuracy=location.hasAccuracy();
hasBearing=location.hasBearing();

distance=location.distanceTo(coords);
if(distance<5000) { distanceState="near"; } else {
distanceState="far"; } } else { distanceState="Unknown"; }
//Toast.makeText(getApplicationContext(), "Distance is : " +
distance, Toast.LENGTH_LONG).show(); /*runOnUiThread(new
Runnable() { public void run() {
Toast.makeText(getApplicationContext(), "Distance is : " +
distance, Toast.LENGTH_LONG).show();
missatger.setText("Distance is : " + distance); } });*/
escribir("Distance is : " + distance); } private void
createGpsDisabledAlert(){ AlertDialog.Builder builder = new
AlertDialog.Builder(this); builder.setMessage("Your GPS is
disabled! Would you like to enable it?") .setCancelable(false)
.setPositiveButton("Enable GPS", new
DialogInterface.OnClickListener(){ public void
onClick(DialogInterface dialog, int id){ showGpsOptions(); }
}); builder.setNegativeButton("Do nothing", new
DialogInterface.OnClickListener(){ public void
onClick(DialogInterface dialog, int id){ dialog.cancel(); }
}); AlertDialog alert = builder.create(); alert.show(); }
private void showGpsOptions(){ Intent gpsOptionsIntent = new
Intent(
android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS);
startActivity(gpsOptionsIntent); } }
```

En nuestro caso la luz la hemos activado a través de un botón externo a la escena , esta señal modifica la textura de la

esfera que nos indica el estado de la bombilla , según sea la respuesta del servidor , ON , OFF o sin conexión , si la bombila está ON la esfera que emula la bombilla encendida , que está en la lámpara , se hace visible rodeando una esfera más pequeña que hace de núcleo de la bombilla.

Tutorial Arduino parte 3

En este tutorial veremos como conectar la placa Arduino a la red de 220V de nuestra casa , y poder utilizarlo para controlar nuestros aparatos eléctricos .

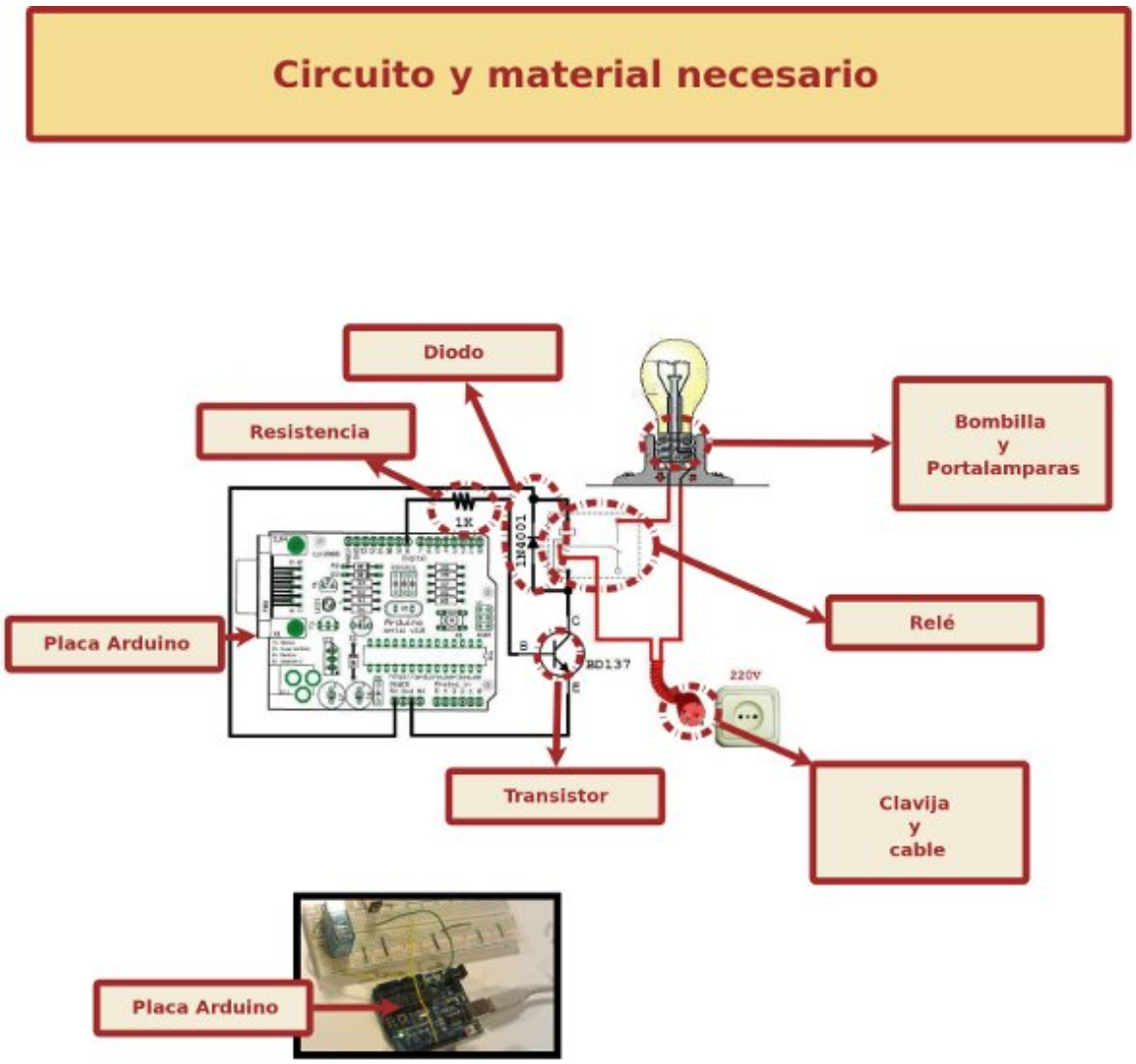
En este ejemplo mostraremos como encender una bombilla a 220V desde un lugar remoto .

Este tutorial está basado en un tutorial de la página de Arduino , ver referencias al final del post.

En el próximo tutorial explicaremos como utilizar una aplicación android en 3D para crear un interfaz para este tutorial , que podremos utilizar en nuestro smartphone Android.

Material necesario:

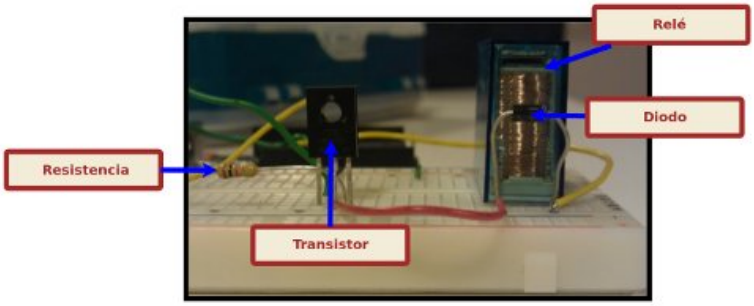
-Placa Arduino
-1 Relé de 5V ó 6V a 220V
-1 Diodo
-1 Transistor 5V
-1 Resistencia
-Cable
-1 Clavija
-1 Portalamparas
-1 Bombilla

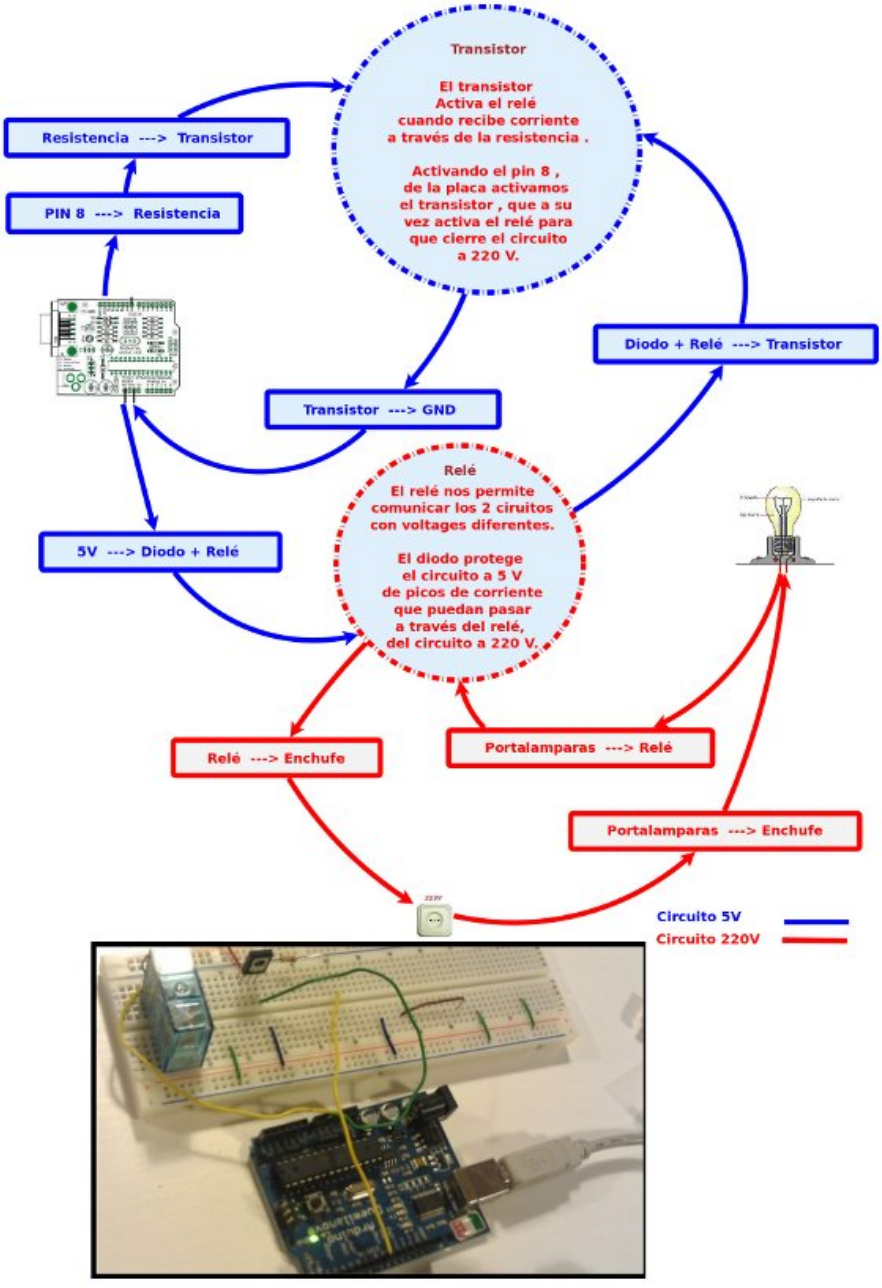


Material necesario

Una vez tengamos el material necesario procedemos a realizar el cableado de los circuitos .

Descripción del cableado de los circuitos





descripción circuitos

Una vez preparado el cableado , el código necesario para la placa Arduino es muy sencillo.
Utilizamos el pin 8 para activar el relé por medio del transistor.

Abrimos el puerto USB 115200 en modo lectura para recibir las notificaciones que nos enviará el servidor PHP.

Si por el puerto nos entra el caracter 2 , activa el relé , si es 1 lo desactiva.

Código Arduino:

```
int ledPin = 8;

int number_in = 0;

void setup() {

pinMode(ledPin, OUTPUT);

Serial.begin(115200);

}

void loop() {

if (Serial.available() > 0) {

number_in = Serial.read();

}

if (number_in > 0) {

if(number_in==2)
{
digitalWrite(ledPin, HIGH);
}
else
{
```

```
if(number_in==1)
{
digitalWrite(ledPin, LOW);
}
}

}

number_in = 0;

}
```

Utilizaremos un simple servidor php para enviar la señal a la placa Arduino conectada a nuestro servidor GNU/linux , en este caso a través del cable USB.

En nuestro caso la ruta al dispositivo que vamos a utilizar es /dev/ttyUSB0.

El puerto de la conexión USB que utilizaremos para comunicarnos con la placa Arduino será el 115200 .

Para encender o apagar la bombilla enviaremos por get a la ruta de nuestro servidor la variable hight , para encender , y la variable low para apagar.

Finalmente el servidor responde con un mensaje si ha recibido una señal correcta.

Código PHP:

```
<?php // Nonzero number to be sent to Arduino $c = 0; if(isset($_GET["hight"])) { $c=2; } if(isset($_GET["low"])) { $c=1; } if($c>0)
{
// Include the PHP serial class
require_once("phpSerial.php");

// Start a new serial class

$serial = new phpSerial;
```

```
// Specify the device being used

$serial->deviceSet("/dev/ttyUSB0");

// Set baud rate

$serial->confBaudRate(115200);

$serial->confParity("none");

$serial->confCharacterLength(8);

$serial->confStopBits(1);

$serial->confFlowControl("none");

// Open the device

$serial->deviceOpen();

// Write to the device

$serial->sendMessage(chr($c));

// Close the port

$serial->deviceClose();
$message= "Signal ".$c." Received! .
";

die(json_encode(array('status' => 'success', 'data' =>
$message)));
}
else
{
$message= "No Signal Received! .";
die(json_encode(array('status' => 'failed', 'data' =>
$message)));
}
?>
```

Y ya lo tenemos , tenemos una página php que nos funciona como

un servicio para poder encender o apagar una bombilla .

Podríamos crear en este punto una pequeña interfaz html , aprovechando el mismo fichero .php , comentando las líneas die(… , creando unos botones para encender y apagar la bombilla , pero nuestra intención es utilizarlo como servicio para una aplicación Android que os explicaremos en el próximo tutorial.

En este ejemplo hemos utilizado la librería phpSerial.php ,no recuerdo de donde la saqué así que os dejo el código , aunque en principio cualquier librería que os permita comunicar con el puerto serie debería valer.

este es el código:

```
<?php define ("SERIAL_DEVICE_NOTSET", 0); define ("SERIAL_DEVICE_SET", 1); define ("SERIAL_DEVICE_OPENED", 2); /** * Serial port control class * * THIS PROGRAM COMES WITH ABSOLUTELY NO WARANTIES ! * USE IT AT YOUR OWN RISKS ! * * @author Rémy Sanchez  * @thanks Aurélien Derouineau for finding how to open serial ports with windows
* @thanks Alec Avedisyan for help and testing with reading
* @copyright under GPL 2 licence
*/
class phpSerial
{
var $_device = null;
var $_windevice = null;
var $_dHandle = null;
var $_dState = SERIAL_DEVICE_NOTSET;
var $_buffer = "";
var $_os = "";

/**
* This var says if buffer should be flushed by sendMessage (true) or manualy (false)
*
```

```
* @var bool
*/
var $autoflush = true;

/**
* Constructor. Perform some checks about the OS and setserial
*
* @return phpSerial
*/
function phpSerial ()
{
setlocale(LC_ALL, "en_US");

$sysname = php_uname();

if (substr($sysname, 0, 5) === "Linux")
{
$this->_os = "linux";

if($this->_exec("stty --version") === 0)
{
register_shutdown_function(array($this, "deviceClose"));
}
else
{
trigger_error("No  stty  availible,  unable  to  run.",
E_USER_ERROR);
}
}
elseif(substr($sysname, 0, 7) === "Windows")
{
$this->_os = "windows";
register_shutdown_function(array($this, "deviceClose"));
}
else
{
trigger_error("Host OS is neither linux nor windows, unable tu
run.", E_USER_ERROR);
```

```
exit();
}
}

//
// OPEN/CLOSE DEVICE SECTION -- {START}
//

/**
* Device set function : used to set the device name/address.
* -> linux : use the device address, like /dev/ttyS0
* -> windows : use the COMxx device name, like COM1 (can also
be used
* with linux)
*
* @param string $device the name of the device to be used
* @return bool
*/
function deviceSet ($device)
{
if ($this->_dState !== SERIAL_DEVICE_OPENED)
{
if ($this->_os === "linux")
{
if (preg_match("@^COM(\d+):?$@i", $device, $matches))
{
$device = "/dev/ttyS" . ($matches[1] - 1);
}

if ($this->_exec("stty -F " . $device) === 0)
{
$this->_device = $device;
$this->_dState = SERIAL_DEVICE_SET;
return true;
}
}
elseif ($this->_os === "windows")
{
```

```
if (preg_match("@^COM(\d+):?$@i", $device, $matches) and
$this->_exec(exec("mode " . $device)) === 0)
{
$this->_windevice = "COM" . $matches[1];
$this->_device = "\\.\com" . $matches[1];
$this->_dState = SERIAL_DEVICE_SET;
return true;
}
}

trigger_error("Specified serial port is not valid",
E_USER_WARNING);
return false;
}
else
{
trigger_error("You must close your device before to set an
other one", E_USER_WARNING);
return false;
}
}

/**
* Opens the device for reading and/or writing.
*
* @param string $mode Opening mode : same parameter as fopen()
* @return bool
*/
function deviceOpen ($mode = "r+b")
{
if ($this->_dState === SERIAL_DEVICE_OPENED)
{
trigger_error("The device is already opened", E_USER_NOTICE);
return true;
}

if ($this->_dState === SERIAL_DEVICE_NOTSET)
{
```

```
trigger_error("The device must be set before to be open",
E_USER_WARNING);
return false;
}

if (!preg_match("@^[raw]\+?b?$@", $mode))
{
trigger_error("Invalid opening mode : ".$mode.". Use fopen()
modes.", E_USER_WARNING);
return false;
}

$this->_dHandle = @fopen($this->_device, $mode);

if ($this->_dHandle !== false)
{
stream_set_blocking($this->_dHandle, 0);
$this->_dState = SERIAL_DEVICE_OPENED;
return true;
}

$this->_dHandle = null;
trigger_error("Unable to open the device", E_USER_WARNING);
return false;
}

/**
* Closes the device
*
* @return bool
*/
function deviceClose ()
{
if ($this->_dState !== SERIAL_DEVICE_OPENED)
{
return true;
}

if (fclose($this->_dHandle))
```

```
{
$this->_dHandle = null;
$this->_dState = SERIAL_DEVICE_SET;
return true;
}

trigger_error("Unable to close the device", E_USER_ERROR);
return false;
}

//
// OPEN/CLOSE DEVICE SECTION -- {STOP}
//

//
// CONFIGURE SECTION -- {START}
//

/**
* Configure the Baud Rate
* Possible rates : 110, 150, 300, 600, 1200, 2400, 4800, 9600,
38400,
* 57600 and 115200.
*
* @param int $rate the rate to set the port in
* @return bool
*/
function confBaudRate ($rate)
{
if ($this->_dState !== SERIAL_DEVICE_SET)
{
trigger_error("Unable to set the baud rate : the device is
either not set or opened", E_USER_WARNING);
return false;
}

$validBauds = array (
110 => 11,
```

```
150 => 15,
300 => 30,
600 => 60,
1200 => 12,
2400 => 24,
4800 => 48,
9600 => 96,
19200 => 19,
38400 => 38400,
57600 => 57600,
115200 => 115200
);

if (isset($validBauds[$rate]))
{
if ($this->_os === "linux")
{
$ret = $this->_exec("stty -F " . $this->_device . " " . (int)
$rate, $out);
}
elseif ($this->_os === "windows")
{
$ret = $this->_exec("mode " . $this->_windevice . " BAUD=" .
$validBauds[$rate], $out);
}
else return false;

if ($ret !== 0)
{
trigger_error ("Unable to set baud rate: " . $out[1],
E_USER_WARNING);
return false;
}
}
}

/**
* Configure parity.
```

```
* Modes : odd, even, none
*
* @param string $parity one of the modes
* @return bool
*/
function confParity ($parity)
{
if ($this->_dState !== SERIAL_DEVICE_SET)
{
trigger_error("Unable to set parity : the device is either not
set or opened", E_USER_WARNING);
return false;
}

$args = array(
"none" => "-parenb",
"odd" => "parenb parodd",
"even" => "parenb -parodd",
);

if (!isset($args[$parity]))
{
trigger_error("Parity mode not supported", E_USER_WARNING);
return false;
}

if ($this->_os === "linux")
{
$ret = $this->_exec("stty -F " . $this->_device . " " .
$args[$parity], $out);
}
else
{
$ret = $this->_exec("mode " . $this->_windevice . " PARITY=" .
$parity{0}, $out);
}

if ($ret === 0)
```

```
{
return true;
}

trigger_error("Unable to set parity : " . $out[1],
E_USER_WARNING);
return false;
}

/**
* Sets the length of a character.
*
* @param int $int length of a character (5 <= length <= 8) *
@return bool */ function confCharacterLength ($int) { if
($this->_dState !== SERIAL_DEVICE_SET)
{
trigger_error("Unable to set length of a character : the
device is either not set or opened", E_USER_WARNING);
return false;
}

$int = (int) $int;
if ($int < 5) $int = 5; elseif ($int > 8) $int = 8;

if ($this->_os === "linux")
{
$ret = $this->_exec("stty -F " . $this->_device . " cs" .
$int, $out);
}
else
{
$ret = $this->_exec("mode " . $this->_windevice . " DATA=" .
$int, $out);
}

if ($ret === 0)
{
return true;
```

```
}

trigger_error("Unable to set character length : " .$out[1],
E_USER_WARNING);
return false;
}

/**
* Sets the length of stop bits.
*
* @param float $length the length of a stop bit. It must be
either 1,
* 1.5 or 2. 1.5 is not supported under linux and on some
computers.
* @return bool
*/
function confStopBits ($length)
{
if ($this->_dState !== SERIAL_DEVICE_SET)
{
trigger_error("Unable to set the length of a stop bit : the
device is either not set or opened", E_USER_WARNING);
return false;
}

if ($length != 1 and $length != 2 and $length != 1.5 and
!($length == 1.5 and $this->_os === "linux"))
{
trigger_error("Specified stop bit length is invalid",
E_USER_WARNING);
return false;
}

if ($this->_os === "linux")
{
$ret = $this->_exec("stty -F " . $this->_device . " " .
(($length == 1) ? "-" : "") . "cstopb", $out);
}
```

```
else
{
$ret = $this->_exec("mode " . $this->_windevice . " STOP=" .
$length, $out);
}

if ($ret === 0)
{
return true;
}

trigger_error("Unable to set stop bit length : " . $out[1],
E_USER_WARNING);
return false;
}

/**
* Configures the flow control
*
* @param string $mode Set the flow control mode. Availible
modes :
* -> "none" : no flow control
* -> "rts/cts" : use RTS/CTS handshaking
* -> "xon/xoff" : use XON/XOFF protocol
* @return bool
*/
function confFlowControl ($mode)
{
if ($this->_dState !== SERIAL_DEVICE_SET)
{
trigger_error("Unable to set flow control mode : the device is
either not set or opened", E_USER_WARNING);
return false;
}

$linuxModes = array(
"none" => "clocal -crtscts -ixon -ixoff",
"rts/cts" => "-clocal crtscts -ixon -ixoff",
```

```
"xon/xoff" => "-clocal -crtscts ixon ixoff"
);
$windowsModes = array(
"none" => "xon=off octs=off rts=on",
"rts/cts" => "xon=off octs=on rts=hs",
"xon/xoff" => "xon=on octs=off rts=on",
);

if ($mode !== "none" and $mode !== "rts/cts" and $mode !==
"xon/xoff") {
trigger_error("Invalid  flow  control  mode  specified",
E_USER_ERROR);
return false;
}

if ($this->_os === "linux")
$ret = $this->_exec("stty -F " . $this->_device . " " .
$linuxModes[$mode], $out);
else
$ret = $this->_exec("mode " . $this->_windevice . " " .
$windowsModes[$mode], $out);

if ($ret === 0) return true;
else {
trigger_error("Unable to set flow control : " . $out[1],
E_USER_ERROR);
return false;
}
}

/**
* Sets a setserial parameter (cf man setserial)
* NO MORE USEFUL !
* -> No longer supported
* -> Only use it if you need it
*
* @param string $param parameter name
* @param string $arg parameter value
```

```
* @return bool
*/
function setSetserialFlag ($param, $arg = "")
{
if (!$this->_ckOpened()) return false;

$return = exec ("setserial " . $this->_device . " " . $param .
" " . $arg . " 2>&1");

if ($return{0} === "I")
{
trigger_error("setserial: Invalid flag", E_USER_WARNING);
return false;
}
elseif ($return{0} === "/")
{
trigger_error("setserial:  Error  with  device  file",
E_USER_WARNING);
return false;
}
else
{
return true;
}
}

//
// CONFIGURE SECTION -- {STOP}
//

//
// I/O SECTION -- {START}
//

/**
* Sends a string to the device
*
* @param string $str string to be sent to the device
```

```
* @param float $waitForReply time to wait for the reply (in
seconds)
*/
function sendMessage ($str, $waitForReply = 0.1)
{
$this->_buffer .= $str;

if ($this->autoflush === true) $this->flush();

usleep((int) ($waitForReply * 1000000));
}

/**
* Reads the port until no new datas are availible, then return
the content.
*
* @pararm int $count number of characters to be read (will
stop before
* if less characters are in the buffer)
* @return string
*/
function readPort ($count = 0)
{
if ($this->_dState !== SERIAL_DEVICE_OPENED)
{
trigger_error("Device must be opened to read it",
E_USER_WARNING);
return false;
}

if ($this->_os === "linux")
{
$content = ""; $i = 0;

if ($count !== 0)
{
do {
if ($i > $count) $content .= fread($this->_dHandle, ($count -
```

```
$i));
else $content .= fread($this->_dHandle, 128);
} while (($i += 128) === strlen($content));
}
else
{
do {
$content .= fread($this->_dHandle, 128);
} while (($i += 128) === strlen($content));
}

return $content;
}
elseif ($this->_os === "windows")
{
/* Do nohting : not implented yet */
}

trigger_error("Reading serial port is not implemented for
Windows", E_USER_WARNING);
return false;
}

/**
* Flushes the output buffer
*
* @return bool
*/
function flush ()
{
if (!$this->_ckOpened()) return false;

if (fwrite($this->_dHandle, $this->_buffer) !== false)
{
$this->_buffer = "";
return true;
}
else
```

```
{
$this->_buffer = "";
trigger_error("Error while sending message", E_USER_WARNING);
return false;
}
}

//
// I/O SECTION -- {STOP}
//

//
// INTERNAL TOOLKIT -- {START}
//

function _ckOpened()
{
if ($this->_dState !== SERIAL_DEVICE_OPENED)
{
trigger_error("Device must be opened", E_USER_WARNING);
return false;
}

return true;
}

function _ckClosed()
{
if ($this->_dState !== SERIAL_DEVICE_CLOSED)
{
trigger_error("Device must be closed", E_USER_WARNING);
return false;
}

return true;
}

function _exec($cmd, &$out = null)
{
```

```
$desc = array(
1 => array("pipe", "w"),
2 => array("pipe", "w")
);

$proc = proc_open($cmd, $desc, $pipes);

$ret = stream_get_contents($pipes[1]);
$err = stream_get_contents($pipes[2]);

fclose($pipes[1]);
fclose($pipes[2]);

$retVal = proc_close($proc);

if (func_num_args() == 2) $out = array($ret, $err);
return $retVal;
}

//
// INTERNAL TOOLKIT -- {STOP}
//
}
?>
```

Referencias:

[Tutorial de la página de Arduino]

[Vídeo del turorial]

Próximamente:

Tutorial Arduino parte 4

En el próximo tutorial explicaremos como utilizar una aplicación android en 3D para crear un interfaz para este tutorial , que podremos utilizar en nuestro celular Android.

C/C++ en Linux

Introducción

En este post voy a comentar alguna librerías y otros útiles para la programación de c/c++ en GNU\Linux. Cuando trabajamos con GNU\Linux todo programa debe tener un adecuado archivo de configuración, debe aceptar parámetros, debe poder recibir señales y actuar en función de ellas, así mismo también debería implementar un sistema de logs. También puede ser interesante acceder a MySql o tener un pequeño servidor web para mostrar información en una simple web html sin necesidad de instalar un Apache entero.

Trataremos

- Señales -> Mostraremos un pequeño codigo con el que podemos captar señales y de esta manera actuar en consecuencia.
- Dotconf -> Mostraremos un ejemplo de esta librería que biene a ser un parser para archivos de configuración estandar.
- Log4cxx -> Esta libería nos permite trabajar con logs comodamente, se bas en un xml de configuración donde podemos especificar como será formatación de las cadenas de loggin por pantalla hasta la configuración de logs rotativos escritos a disco.
- MySql – libmysql++ -> Con esta librería podremos

realizar operaciones sobre bases de datos MySql.

- Swill Server -> Esta librería nos permitirá levantar un pequeño servidor web.
- Jansson -> Esta librería nos permitirá parsear estructuras fácilmente.

Cabe decir que no se va profundizar, esto pretende ser una aproximación a buenas «maneras» de trabajar de programar con GNU\Linux.

Señales

En esencia una señal es una notificación asíncrona enviada a un a un programa. En caso de que no se haya programado un handle el programa será el que trate las señal, sinó se ejecutará la accción por defecto para esa señal. En este ejemplo veremos como programar el handle para poder manejar las señales que recibamos.

Bien para compilarlo: g++ main.cpp -o signal

Codigo:

```
*
* main.cpp
* Author: ruben
*/

#include
#include
#include
#include

void kctrlc(int signum){
printf("He recibido un ctrl + c (%d). Deberia abortar lo que este haciendo ahora.\n", signum);
}
```

```
void ksigusr(int signum){
printf("He recibido un sigusr 1 o 2 (%d)\n", signum);
}

void ksigt(int signum){
printf("He recibido un sigterm (%d). Deberia acabar lo que
este haciendo...\n", signum);
}

void ksigalrm(int signum){
printf("He recibido un sigalrm (%d).\n", signum);
}
void ksighup(int signum){
printf("He recibido un sighup (%d). Deberia releer la
conmfiguracion de sete programa.\n", signum);
}

void kgeneric(int signum){
printf("He recibido otra senal (%d)\n", signum);
}

void ksigsegv(int signum){
printf("Segmentation fault (%d).\n", signum);
exit(1);
}

void ksigbus(int signum){
printf("bus error (%d).\n", signum);
exit(2);
}

/* Esta funcion es un distribuidor, capta las senales y en
funcion de eso
* llama a una u otra funcion.
*/
void ksighandler(int signum){
switch (signum){
case SIGINT:
kctrlc(signum);
```

```
break;
case SIGUSR1:
case SIGUSR2:
ksigusr(signum);
break;
case SIGTERM:
ksigt(signum);
break;
case SIGHUP:
ksighup(signum);
break;
case SIGALRM:
ksigalrm(signum);
break;
case SIGSEGV:
ksigsegv(signum);
break;
case SIGBUS:
ksigbus(signum);
break;
default:
kgeneric(signum);
break;
}
/* reprogramamos la senal que ha llegado para que vuelva ha
hacer lo mismo
*/
signal(signum, ksighandler);
}

int main(){
int i;
/* haciendo 'kill -l' veremos que hay 64 senales... les
reprogramaremos
* para que cuando llegue una se ejecute la funcion la funcio
ksighandler.
*/
```

```
for (i = 1; i <= 64; i++){ signal(i, ksighandler); } /* programamos una alarma para que en 30 segundons envie un SIGALRM (14) * al proceso */ alarm(30); /* con este bucle infinito veremos el pid del proceso, para * poder enviarle las senales que queramos con 'kill -num $PID' o */ while (1){ printf("Soy el pid %d y espero senales\n", getpid()); sleep(2); } }
```

Dotconf

Esta librería nos permitirá manejar fácilmente archivos standar de configuración.

Para compilar: g++ conf.cpp -o conf -ldotconf

Archvo configuración

```
# the default behaviour for dot.conf is to stop parsing as
# soon as the first unquoted, unescaped #-sign is found
# you can override it by giving the flag NO_INLINE_COMMENTS to dotconf_create()

var1 'upper case' #inline comment 1
var2 '1' # inline comment 2
var3 '2' # inline comment 3
var4 '3' #inline comment 4
```

Archivo fuente

```
/*
* main.cpp
* Author: ruben
*/

#include
#include
#include

/*
tabsize: 4
shiftwidth: 4
```

```
*/

DOTCONF_CB(cb_noinline){
int i;
int j=0;
char sport[200];
char sport2[200];
char sport3[200];
char sport4[200];

printf("[test.conf] Have %d args\n", cmd->arg_count);

for (i = 0; i < cmd->arg_count; i++){

if(j==0){
strcpy (sport,cmd->data.list[i]);
printf("Arg: %s\n", sport);
j=1;
}else if (j== 1){
strcpy (sport2,cmd->data.list[i]);
printf("Arg: %s\n", sport2);
j=2;
}else if(j==2){
strcpy (sport3,cmd->data.list[i]);
printf("Arg: %s\n", sport3);
j=3;
}else if(j==3){
strcpy (sport4,cmd->data.list[i]);
printf("Arg: %s\n", sport4);
j=0;
}
}

return NULL;
}

static configoption_t options[] = {
{"var1", ARG_LIST, cb_noinline, NULL, 0},
{"var2", ARG_LIST, cb_noinline, NULL, 0},
```

```
{"var3", ARG_LIST, cb_noinline, NULL, 0},
{"var4", ARG_LIST, cb_noinline, NULL, 0},
LAST_OPTION
};

void readit(int flags){
configfile_t *configfile;

configfile = dotconf_create("test.conf", options, 0, flags);
if (!dotconf_command_loop(configfile))
fprintf(stderr, "Error reading config file\n");
dotconf_cleanup(configfile);
}

int main(int argc, char **argv){
//printf("Reading the configuration with NO_INLINE_COMMENTS enabled\n");
//readit(NO_INLINE_COMMENTS);

//printf("\n\n");
printf("Reading the configuration\n");
readit(0);

//printf("%s\n",cmd->data.list[0]);

return 0;
}
```

Log4cxx

Esta librería nos permitirá gestionar los log's de nuestra aplicación adecuadamente.

Para compilar: g++ log.cpp -I/opt/include /usr/lib/liblog4cxx.a -lapr-1 -laprutil-1

Archvo configuración

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

```
#include
#include

using namespace log4cxx;
using namespace log4cxx::xml;
using namespace log4cxx::helpers;

// Define static logger variable
LoggerPtr loggerMyMain(Logger::getLogger( "main"));
LoggerPtr logtest(Logger::getLogger( "Function A"));

void functionA(){
LOG4CXX_INFO(logtest, "Executing functionA.");
LOG4CXX_INFO(logtest, "exiting functionA.");
}

int main(){
// Load configuration file
DOMConfigurator::configure("conf.xml");

LOG4CXX_TRACE(loggerMyMain, "this is a debug message for
detailed code discovery.");
LOG4CXX_DEBUG(loggerMyMain, "this is a debug message.");
LOG4CXX_INFO (loggerMyMain, "this is a info message,
ignore.");
LOG4CXX_WARN (loggerMyMain, "this is a warn message, not too
bad.");
```

```
LOG4CXX_ERROR(loggerMyMain, "this is a error message, something serious is happening.");
LOG4CXX_FATAL(loggerMyMain, "this is a fatal message!!!");
functionA();

return 0;
}
```

MySql

Esta librería nos permitirá acceder y utilizar bases de datos Mysql comodamente.

Archvo configuración sql

```
-- phpMyAdmin SQL Dump
-- version 3.4.10.1deb1
-- http://www.phpmyadmin.net
--
-- Servidor: localhost
-- Temps de generació: 31-08-2012 a les 15:27:03
-- Versió del servidor: 5.5.24
-- Versió de PHP : 5.3.10-1ubuntu3.2

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Base de dades: `emp`
```

```
--

-- --------------------------------------------------------

--
-- Estructura de la taula `emp`
--

CREATE TABLE IF NOT EXISTS `emp` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`name` text NOT NULL,
`surname` text NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;

--
-- Bolcant dades de la taula `emp`
--

INSERT INTO `emp` (`id`, `name`, `surname`) VALUES
(1, 'test', 'test1'),
(2, 'test2', 'test3');

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT
*/;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS
*/;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION
*/;

Para compilar: gcc conexion.cpp main.cpp -o conexion2 -Wno-
deprecated -I/usr/include/mysql -L/usr/lib/mysql -lmysqlclient
-lstdc++ -lz
/*
* main.cpp
* Author: ruben
*/

#include
```

```
#include
#include

int main(){

MYSQL_RES *result;
MYSQL_ROW row;
MYSQL *connection, mysql;

int state;

mysql_init(&mysql);

connection                                                    =
mysql_real_connect(&mysql,"localhost","user","pass","emp",0,0,
0);

if (connection == NULL){
printf("Error: %s\n",mysql_error(&;mysql));
return 1;
}

state = mysql_query(connection, "SELECT * FROM emp");

if (state !=0){
printf("Error: %s\n",mysql_error(connection));
return 1;
}

result = mysql_store_result(connection);
printf("Rows: %d\n",mysql_num_rows(result));

while ( ( row=mysql_fetch_row(result)) != NULL ){
printf("id -> %s, name -> %s, surname -> %s\n", (row[0] ?
row[0] : "NULL"), (row[1] ? row[1] : "NULL"),(row[2] ? row[2]
: "NULL"));
}

mysql_free_result(result);
mysql_close(connection);
```

```
return 0;
};
```

Sqlite3

No siempre enecesitamos guardar datos en una gestor como pueda ser MySql, aveces no requerimos de todo lo que nos puede ofrecer y con SqLite podemos tener una compensación entre recursos consumidos y funcionalidad muy buena

Archvo configuración sql

```
BEGIN TRANSACTION;
CREATE TABLE emp (id TEXT, name TEXT, surname TEXT);
INSERT INTO emp VALUES(1000,'test','test2');
COMMIT;
```

Para compilar: g++ main.cpp -o sqlite -lsqlite3

```
/*
* main.cpp
* Author: ruben
*/

#include "sqlite3.h"
#include
#include
#include
#include

#include
#include
#include

#include
#include

char missatge[1000];
char sentence[1000];
sqlite3 *db;
```

```
char *zErrMsg = 0;
int rc;

char **result;

int nrow;
int ncol;
int i;

void insertar(){

rc = sqlite3_open("emp", &db);

if( rc ){
sprintf(missatge, "Test: Can't open database %s to insert row
for DB persistence\n", sqlite3_errmsg(db));
sqlite3_close(db);
exit(1);
}

i++;

sprintf(sentence, "insert into emp ('id', 'name', 'surname')
values ('%d', 'test', 'test2')", i);

rc = sqlite3_get_table(
db,
sentence,
&result,
&nrow,
&ncol,
&zErrMsg
);

sqlite3_free_table(result);
sqlite3_close(db);

if (zErrMsg != NULL){
printf("Error:%s\n",zErrMsg);
std::cin.ignore(); // ignores the newline
```

```
std::cin.get(); //waits for character
}else{
printf("Insertado correctamente\n");
std::cin.ignore(); // ignores the newline
std::cin.get(); //waits for character
}
}

void borrar(){

rc = sqlite3_open("emp", &db);

if( rc ){
sprintf(missatge, "SpoolAndScheduler: Can't open database %s
to delete row for DB persistence\n", sqlite3_errmsg(db));
sqlite3_close(db);
exit(1);
}

std::string cons="delete from emp where id='1'";

rc = sqlite3_get_table(
db,
cons.c_str(),
&result,
&nrow,
&ncol,
&zErrMsg
);

sqlite3_free_table(result);
sqlite3_close(db);

if (zErrMsg != NULL){
printf("Error:%s\n",zErrMsg);
std::cin.ignore(); // ignores the newline
std::cin.get(); //waits for character
}else{
printf("Borrado correctamente\n");
```

```
std::cin.ignore(); // ignores the newline
std::cin.get(); //waits for character
}

}

void listar(){

rc = sqlite3_open("emp", &db);

if( rc ){
sprintf(missatge, "SpoolAndScheduler: Can't open database %s
to select rows for DB persistence\n", sqlite3_errmsg(db));
sqlite3_close(db);
exit(1);
}

std::string cons="select * from emp";//where ds='as'";

rc = sqlite3_get_table(
db,
cons.c_str(),
&result,
&nrow,
&ncol,
&zErrMsg
);

printf("\n --------RESULT-------- \n");

//printf("Voltes->%d\n",(nrow*ncol)+ncol);
int j=(nrow*ncol)+ncol;

for(i=0 ; i < j; i=i+3){ if(i>=3){
printf("i->%d - ",i);
printf("id: %s - ",result[i]);
printf("name: %s - ",result[i+1]);
printf("surname: %s\n",result[i+2]);
}
}
```

```
sqlite3_free_table(result);

sqlite3_close(db);

if (zErrMsg != NULL){
printf("Error:%s\n",zErrMsg);
std::cin.ignore(); // ignores the newline
std::cin.get(); //waits for character
}else{
std::cin.ignore(); // ignores the newline
std::cin.get(); //waits for character
}
}

int main(){

int fin = 0;
int opcion;

while (fin == 0){
system("clear");
printf("Menú simple\n\n");

printf("\t1] Insertar.\n");
printf("\t2] Borrar.\n");
printf("\t3] Listar.\n");
printf("\t4] Sortir.\n");

printf("\n\nOpción: ");
scanf("%i", &opcion);

switch(opcion){
case 1:
insertar();
break;
case 2:
borrar();
break;
case 3:
```

```
listar();
break;
case 4:
fin = 1;
break;
default:
fin = 0;
break;
}

}

return 0;
}
```

Swill Server

SWILL (***S**imple **W**eb **I**nterface **L**ink **L**ibrary*) es una librería que nos permite de manera fácil añadir una interfaz web a un programa en C/C++. Deberemos bajarnos la librería y ejecutar los siguientes comandos para descomprimirla desde Swill e instalarla:

```
tar xvzf swill-0.3.tgz
cd SWILL-0.3
./configure
make
make install
```

Para compilar: g++ main.cpp -o swill_server -I/usr/local/include/swill/ -lswill

```
/*
* main.cpp
* Author: ruben
*/

#include
```

```
void count_to_ten(FILE *f) {
int i;
for (i = 0; i < 10; i++) { fprintf(f,"%d\n", i); } } int
main()  {  swill_init(8181);  swill_handle("ten.txt",
count_to_ten, 0); swill_file("index.html",0); while (1) {
swill_serve(); } swill_shutdown(); }
```

Jansson

Esta librería nos permitirá manejar fácilmente archivos json.

Para compilar: gcc main.c -o json -ljansson

Archvo json

```
{
"errors": [
{
"id": "1",
"message": "string not found"
},
{
"id": "2",
"message": "system error"
}

]
}
```

Archivo fuente

```
/*
* Author: Ruben
*/

#include
#include
```

```
#include

#define BUFFER_SIZE (256 * 1024) /* 256 KB */

#define                                          URL_FORMAT
"http://github.com/api/v2/json/commits/list/%s/%s/master"
#define URL_SIZE 256

/* Return the offset of the first newline in text or the
length of
text if there's no newline */
static int newline_offset(const char *text){
const char *newline = strchr(text, '\n');
if(!newline)
return strlen(text);
else
return (int)(newline - text);
}

struct write_result{
char *data;
int pos;
};

int main(int argc, char *argv[]){
size_t i;

json_t *root;
json_error_t error;
json_t *errors;

root = json_load_file("file.json", 0, &error);

if(!root){
fprintf(stderr, "error: on line %d: %s\n", error.line,
error.text);
return 1;
}

errors = json_object_get(root, "errors");
```

```
if(!json_is_array(errors)){
fprintf(stderr, "error: errors is not an array\n");
return 1;
}

for(i = 0; i < json_array_size(errors); i++){ json_t *error, *id, *message; const char *message_text; error = json_array_get(errors, i); if(!json_is_object(error)){ fprintf(stderr, "error: error %d is not an object\n", i + 1); return 1; } id = json_object_get(error, "id"); if(!json_is_string(id)){ fprintf(stderr, "error: error %d: id is not a string\n", i + 1); return 1; } message = json_object_get(error, "message"); if(!json_is_string(message)) { fprintf(stderr, "error: error %d: message is not a string\n", i + 1); return 1; } message_text = json_string_value(message); printf("%.8s %.*s\n", json_string_value(id), newline_offset(message_text), message_text); } json_decref(root); return 0; }
```

Observaciones

Bien, espero que esto pueda ser de ayuda. No es más que un poco de cada. Pero al fin y al cabo estas utilidades siempre puede ir bien.

Ruben

Python XMPP Client (Twisted)

Introducción

En este post vamos a realizar un cliente de xmpp en python. Es de obligada mención Twisted Matrix, una increíble framework para python el cual facilita muchíssimo la programación

enfocada a redes. En cuanto a protocolos podemos trabajar con HTTP, XMPP, NNTP, IMAP, SSH, IRC, FTP y muchos más, también podemos trabajar con varias arquitecturas (TCP, UDP, SSL/TLS, IP Multicast, Unix domain sockets).

Características

- Separación de los protocolos y transportes

El diseño de Twisted se basa en la separación completa entre los protocolos lógicos (que por lo general dependen de la conexión semántica basada en streams –flujos–, como el HTTP o POP3) y el transporte en capas físicas soportado como la semántica basada en streams (como archivos, bibliotecas sockets o SSL). La conexión entre un protocolo lógico y una capa de transporte que ocurre en el último momento posible, justo antes de la información se pase a la instancia de protocolo lógico. El protocolo lógico es informado de la instancia de capa de transporte, y puede utilizarlo para enviar mensajes de un lado para comprobar la identidad del otro extremo. Tenga en cuenta que todavía es posible, en el código de protocolo, para consultar profundamente la capa de transporte en cuestiones de transporte (como la comprobación de un certificado SSL del lado del cliente). Naturalmente, el código de dicho protocolo, se producirá un error (lanzar una excepción) si la capa de transporte no es compatible con tales semánticas.

- Deferreds

El modelo central de aplicación para Twisted es el concepto de un deferred (predefinir algo que se usara como valor futuro). Un deferred es un valor que no se ha calculado todavía, por ejemplo, porque las necesidades de datos desde un equipo remoto. Los deferreds se pueden transferir, al igual que los objetos normales, pero no se puede pedir por su valor. Cada deferred es compatible con una cadena de devolución de llamada. Cuando el deferred toma el valor, es transferido a

través de la cadena de devolución de llamada, con el resultado de cada de callback (devolución) siendo la entrada (input) para la siguiente. Esto permite que operen en los valores de un deferred sin saber lo que son. Por ejemplo, si un deferred devuelve una cadena desde un equipo remoto con una dirección IP en formato quad, un callback se puede adjuntar para traducirla a un número de 32 bits. Cualquier usuario del deferred puede ahora tratarlo como deferred de retorno de un número de 32 bits. Esto, y la capacidad de relación para definir «errbacks» (callbacks que son llamados como controladores de errores), permite que el código que se ve como si fuera de serie, mientras que todavía mantiene la abstracción por eventos.

- Soporte de Thread (hilos o subprocesos)

Twisted soporta una abstracción sobre threads en crudo usando un thread como una fuente deferred. Por lo tanto, un deferred que es retornado inmediatamente, recibirá un valor cuando finalice el thread. Los callbacks se pueden adjuntar cuando corran en el thread principal, a fin de aliviar la necesidad de soluciones complejas de bloqueo. Un buen ejemplo de tal uso, que viene de las bibliotecas de soporte de Twisted, es usar este modelo para llamadas en bases de datos. La llamada de la base de datos misma pasa de un thread exterior, pero el análisis del resultado que sucede en el thread principal.

- Soporte de bucle de externos

Twisted se puede integrar con bucles de eventos externos, tales como los de GTK+, Qt y Cocoa (a través de PyObjC). Esto le permite el uso de Twisted como la capa de soporte de red en aplicaciones GUI, usando todas sus colecciones sin tener que añadir una sobrecarga de thread-por-socket, como lo haria cualquier biblioteca nativa de Python. Se pude integrar en proceso un completo web server con una aplicación interfaz gráfica utilizando este modelo, por ejemplo.

Cabe decir que no es objetivo de este post profundizar en twisted matrix, pues se necesitarían sería muy extenso (de echo harían falta muchos :p). También hace falta añadir, que hay una muy buena documentación en la página oficial.

Instalación

Podemos descargar el framework des de la página oficial:
Twisted Matrix Downloads
O bien odemos instalarlo fácilmente en Debian y derivados, puesto que esta en los repositorios (sino en el link de la página oficial antes citada te facilitan la PPA):

```
$>sudo apt-get install python-twisted python-twisted-words
```

El modulo twisted-words contiene las librerías de jabber.

Codigo del Cliente

```
#!/usr/bin/python

# Twisted Imports
from twisted.words.protocols.jabber import client, jid , xmlstream
from twisted.words.xish import domish
from twisted.internet import reactor

name = None
server = None
resource = None
password = None
me = None

thexmlstream = None
tryandregister = 1

def initOnline(xmlstream):
     # creamos los observadores hacia las respuestas xml message y una general (podemos incluir presence, iq...)
    global factory
    print 'Initializing...'
```

```
    xmlstream.addObserver('/message', gotMessage)
    xmlstream.addObserver('/*', gotSomething)

def authd(xmlstream):
    # Autentificacion
    global thexmlstream
    thexmlstream = xmlstream
    print "we've authd!"
    print repr(xmlstream)

    # se envia la presencia a los demas clientes
    presence = domish.Element(('jabber:client', 'presence'))
    presence.addElement('status').addContent('Online')
    xmlstream.send(presence)

    initOnline(xmlstream)

def send(author, to, msg):
    # esta funcion envia los mensajes
    global thexmlstream
    message = domish.Element(('jabber:client','message'))
    message["to"] = jid.JID(to).full()
    message["from"] = jid.JID(author).full()
    message["type"] = "chat"
    message.addElement("body", "jabber:client", msg+ "- ya lo
sabia adicotalainformatica1")

    thexmlstream.send(message)

def gotMessage(el):
    # esta funcion parsea los mensajes recibidos
    global me
    # print 'Got message: %s' % str(el.attributes)
    from_id = el["from"]

    body = "empty"
    for e in el.elements():
        if e.name == "body":
            body = unicode(e.__str__())
        break
```

```
    send(me, from_id, body)

def gotSomething(el):
    # Observador general
      print 'Got something: %s -> %s' % (el.name,
str(el.attributes))

def authfailedEvent(xmlstream):
    global reactor
    print 'Auth failed!'
    reactor.stop()

def invaliduserEvent(self,xmlstream):
    print "Invalid User"

def registerfailedEvent(self,xmlstream):
    print 'Register failed!'

if __name__ == '__main__':
    #Parametrizamos la conexion
    PASSWORD = '123456'
    myJid = jid.JID('adictoalainformatica2@antitot-linux')
    me = 'adictoalainformatica2@antitot-linux'
    factory = client.XMPPClientFactory(myJid, PASSWORD)

    # Registramos las callbacks de autentificacion
    print 'register callbacks'
    factory.addBootstrap(xmlstream.STREAM_AUTHD_EVENT, authd)
factory.addBootstrap(client.BasicAuthenticator.INVALID_USER_EV
ENT, invaliduserEvent)
factory.addBootstrap(client.BasicAuthenticator.AUTH_FAILED_EVE
NT, authfailedEvent)
factory.addBootstrap(client.BasicAuthenticator.REGISTER_FAILED
_EVENT, registerfailedEvent)

    # Paarametrizamos y arrancamos el reactor (encargado de
mantener y gestionar las callbacks
    # producidas por las acciones de la conexion)
    reactor.connectTCP('localhost', 5222, factory)
```

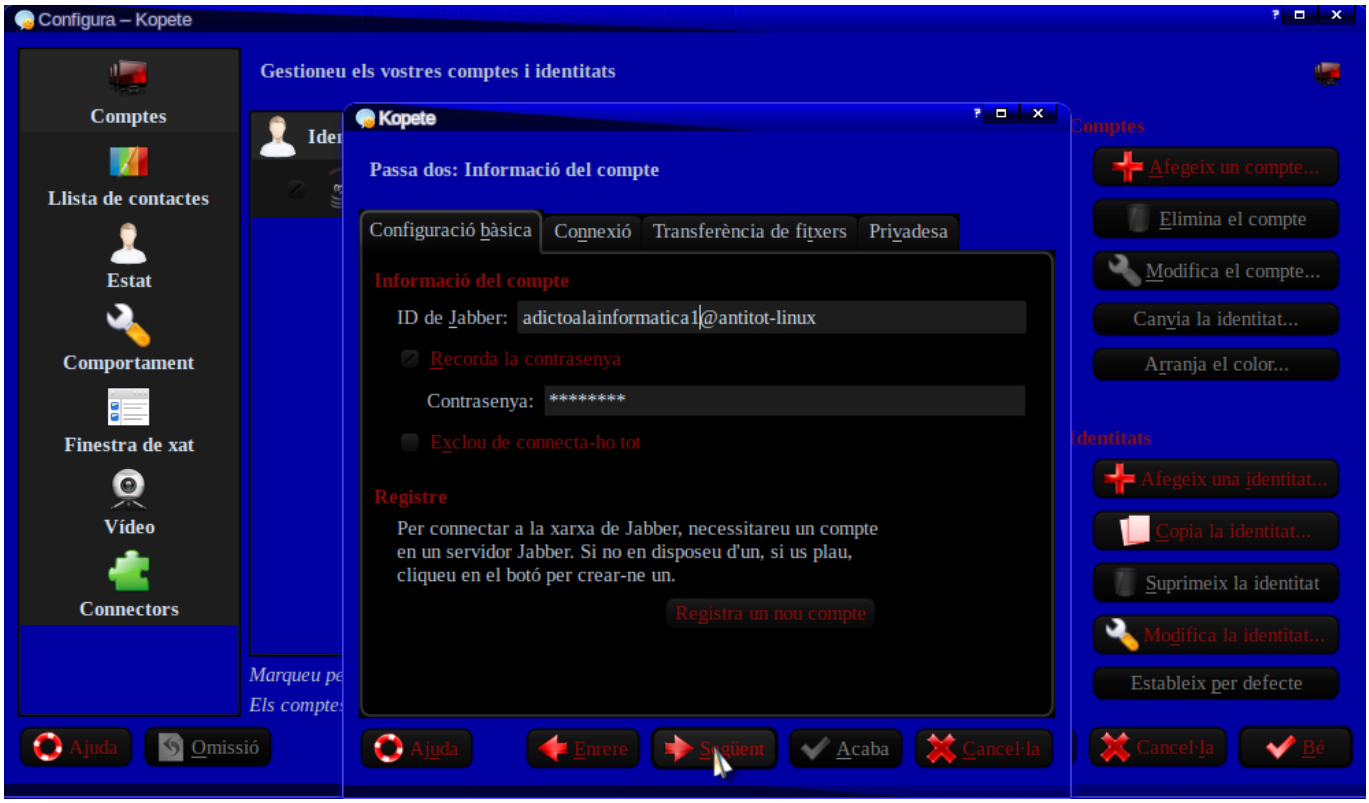
Test

Para realizar el test haremos referencia al post anteriror sobre xmpp donde configuramos y creamos dos usuarios (adictosalainformatica1, adictosalainformatica2) con el servidor openfire. Aprovecharemos los usuarios
para configurar el script e indicaremos al reactor donde esta el servidor xmpp:

```
#Parametrizamos la conexion
PASSWORD = '123456'
myJid = jid.JID('adictoalainformatica2@antitot-linux')
me = 'adictoalainformatica2@antitot-linux'
factory = client.XMPPClientFactory(myJid, PASSWORD)
reactor.connectTCP('antitot-linux', 5222, factory)
```

y cualquier cliente como por ejemplo Kopete. Abriremos Kopete e hiremos a configuraciones -> configura. Se abrirá una nueva ventana en esta pincharemos en añadir nuevo elemento.
Seguidamente deberemos escoger el protocolo:

Configuramos la cuenta:



Y ya esta, cuando des del usuario adictosalainformatica enviemos un mensaje a nuestro script éste le responderá el mismo mensaje con la coletilla » – ya lo sabia adicotalainformatica2″.

Código del cliente Xmpp

Puedes descargar el código de este post desde la cuenta de GitHub

Observaciones

Y con este post cierro el tema de redes xmpp dando una vista superficial pero intuitiva de lo que puede ofrecernos este nuevo protocolo tanto a nivel de mensajería como a nivel de distribuidor de comandos. espero que haya sido de ayuda.

Fuentes

- Twisted Matrix
- Wikipedia – Twisted Matrix

Ruben

Android sdk en Ubuntu unknown device ???????????

Tras instalar android sdk , eclipse , actualizar etc , nos podemos encontrar con que el emulador funciona,
pero no podemos debugar en el dispositivo.

Debugar en el dispositivo , nos ayuda a crear aplicaciones para los sensores , y nos permite probarlo al mismo tiempo , como los sensores de nivel , gps , etc , sin el dispositivo sería muy difícil .

Problema en DDMS , dispositivo desconocido ???????????
No funciona el debug con el dispositivo.

/android-sdk/platform-tools$ ./adb devices devuelve :

List of devices attached
??????????? ??
unknown device ???????????
Fuente de la solución
En esta zona podemos ver la solución:

> *14:25 W/ddms: Unable to get frame buffer: device (???????????) request*
> *rejected: insufficient permissions for device*
> ==
>
> *I then checked*
>
> ==
> *$ ./adb devices*
> *List of devices attached*
> *??????????? no permissions*

> ==
>
> There appears to be a permission problem. I then wrote a
> `51-android.rules' file (with chmod a+r) in /etc/udev/rules.d/, such that
>
> ==
> # cat 51-android.rules
> SUBSYSTEM==»usb», ATTRS{idVendor}==»0bb4″, ATTRS{idProduct}==»0c87″,
> MODE=»0666″
> ==
>
> (the idVendor is the HTC one). I then restarted udev, but to no avail:
> same problems. Any idea?
>
>
>

_
Merciadri Luca
See http://www.student.montefiore.ulg.ac.be/~merciadri/
I use PGP. If there is an incompatibility problem with your mail
client, please contact me.

Old 08-13-2010, 02:33 PM
Merciadri Luca

Default Android phone is not recognized by Android SDK, despite udev conf
Problem solved: restarting udev from CLI was not sufficient. I needed to
restart.

Resumiendo:

1:)-> crear el fichero /etc/udev/rules.d/51-android.rules :
$ sudo gedit /etc/udev/rules.d/51-android.rules
con el siguiente contenido:
SUBSYSTEM=="usb", ATTRS{idVendor}=="0bb4", ATTRS{idProduct}=="0c87",
MODE="0666"

2:)-> darle permisos $ sudo chmod a+r /etc/udev/rules.d/51-android.rules
3:)-> reiniciar

Bueno , ya podemos utilizar el dispositivo , en la carpeta donde esté instalado adb ,
./adb devices , devuelve en mi caso

List of devices attached
HT0C4RX21596 device

ya podemos debugar , probamos creando un proyecto Android en eclipse , pulsamos el botón de debug y ya tenemos nuestro hello world …. que tanto deseábamos.

PHP ON COUCH

Introducción

Php On Couch es una librería en php para atacar bases de datos, en adelante bbdd, de CouchDB . Por el uso que le he dado he podido comprobar su estabilidad y ademas se actualiza regularmente, lo cual es de agradecer. Cabe decir también que esta bajo licencia GPL.

En este ejemplo veremos como conectar a una base de datos y extraer la información de un documento, obviamente esta librería ofrece muchísimas más funcionalidades, pero este post

no pretende ser una explicación de cada una de ellas. Pretende mostrar una utilización práctica de la librería, en él se creará un gráfico con Google Chart Api a partir de datos guardados en CouchDB.

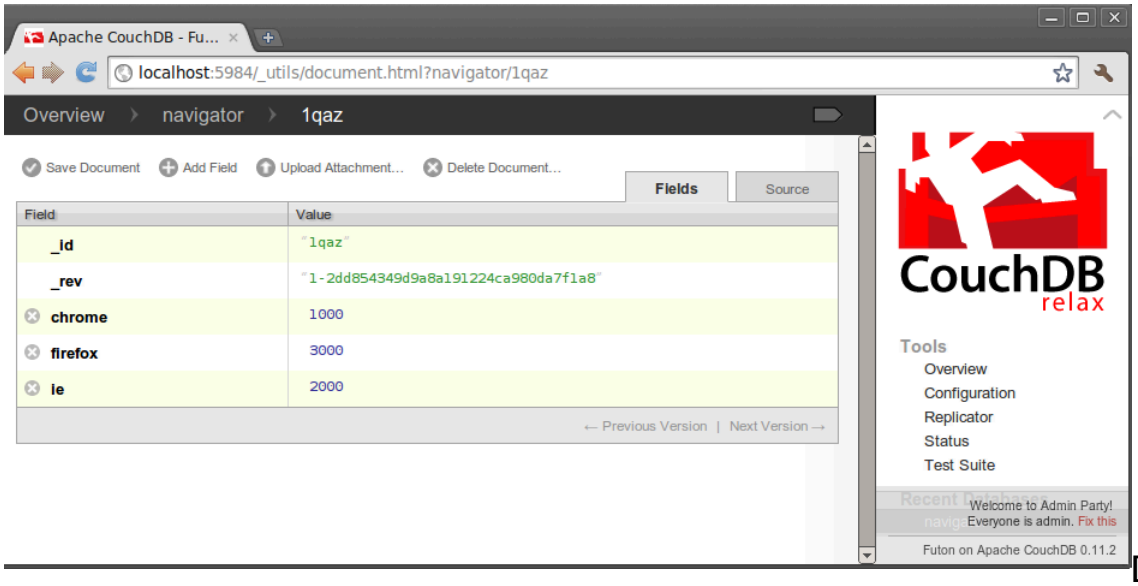
Estructura

Dispondremos de los siguiente ficheros:

- index.php: archivo principal.
- couch.php: clase que utilizaremos para acceder a CouchDB.
- config.php: archivo de configuración.

Crearemos una base de datos con las siguientes características:

- El nombre de la base de datos será navigator.
- El id del documento al que accederemos tendrá el siguiente id: 1qaz.
- El documento almacenara un valor para los siguientes tags: ie, firefox y chrome.



Debemos descargarnos la librería Php On CouchDB y estructurar el proyecto conforme la la carpeta lib, de dicha librería, se encuentre en el mismo nivel que los archivos PHP.

index.php

Este archivo incluirá la la clase couch.php, la instanciará y llamará al método chart para que este le retorne el gráfico.

```
/**
 * User: rmiguel
 * Date: 22/07/2015
 * Time: 20:00
 */

require_once "CouchWrapper.php";

print "

                        Couchdb Test

«;     $con     =     new     CouchWrapper();     print $con->chart(«600×200″,»1qaz»);
```

config.php

Este archivo es el típico utilizado para parametrizar las WebApp que tienen acceso a bbdd, no comentare el archivo por su simpleza y los descriptivos nombres de las variables. Dado que esto es un ejemplo, me permito la licencia de utilizar variables globales, no cabe decir que en una aplicación de uso está sería una mala implementación de uso para un archivo de configuración dada la información que contiene. Personalmente yo creo un xml que parseo con un método privado en la clase, el cual guarda el valor de los tags en variables privadas. De esta manera aseguramos que este método y las variables son usadas únicamente por la clase protegiéndolas dada la importancia de su valor.

```
/**
 * User: rmiguel
 * Date: 22/07/2015
 * Time: 20:00
 */
//Couch_db config
```

```
$COUCHDB_HOST = "127.0.0.1";
$COUCHDB_PORT = "5984";
$COUCHDB_TABLE = "navigator";
```

couch.php

Este archivo es el que contiene la clase _couch. Primero incluimos las dependencias (de la librería y el archivo de configuración) .

A continuación explicaremos la estructura de la classe:

Variables

- $client será donde se guardará la instancia cliente que ataca a la bbdd.
- $couchdsn será la url y el puerto (http://127.0.0.1:5894)
- $couch_db será la base de datos (navigator)

Métodos

- _couch -> es el constructor en él se realizará la conexión a la bbdd.
- get_doc($id) -> retornara una variable con la estructura del documento correspondiente al id que se le pasa por parámetro.
- chart -> retornará una url del tipo pie chart que atacara a google chart api para crear un gráfico en función del id de documento y la resolución que se especifique.

```
/**
 * User: rmiguel
 * Date: 22/07/2015
 * Time: 20:00
 */
//-----------------------
require_once "lib/couch.php";
require_once "lib/couchClient.php";
require_once "lib/couchDocument.php";
```

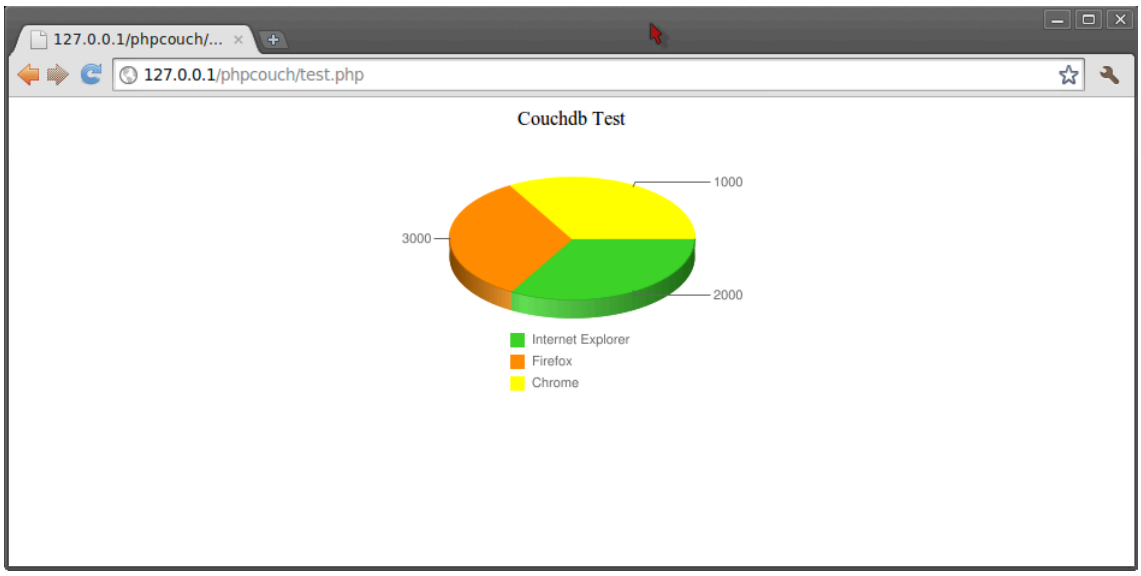
```
require_once "config.php";
//-----------------------------------
class CouchWrapper {
    public $client;
    public $couch_dns;
    public $couch_db;
    /**
     * Constructor
     */
    function __construct(){
                                                    $couch_dsn
=".$GLOBALS['COUCHDB_HOST'].":".$GLOBALS['COUCHDB_PORT'];
        $couch_db = $GLOBALS['COUCHDB_TABLE'];
        try{
                                    $this->client   =   new
couchClient($couch_dsn,$couch_db);
        } catch (Exception $e) {
                        echo  "Error:".$e->getMessage()."
(errcode=".$e->getCode().")\n";
        }
    }
    /**
     * @param $id
     * @return document values
     */
    function get_doc($id){
        try{
            $doc = $this->client->getDoc($id);
        } catch (Exception $e) {
            if ( $e->code() == 404 ) {
                echo "Document \"some_doc\" not found\n";
            } else {
                        echo "Something weird happened:
".$e->getMessage()." (errcode=".$e->getCode().")\n";
            }
            exit(1);
        }
        return $doc;
    }
    /**
     * @param $resolution
```

```
     * @param $id_document
     * @return chart image tag
     */
    function chart($resolution,$id_document){
        $doc = $this->get_doc($id_document);
        $green = $doc->ie;
        $orange = $doc->firefox;
        $yellow = $doc->chrome;
        $chart = "
```

```
«; return $chart; } }
```

Resultado

Una vez ejecutemos en nuestro servidor web index.php deberíamos obtener este resultado si todo ha funcionado correctamente:



Observaciones

Bien con este post doy por finalizada la breve introducción al mundo de las bbdd NoSql, espero y deseo que hayan sido de ayuda.

Posts relacionados

-> NoSql

-> NoSql – CouchDB

Código fuente en GitHub

PHPonCouch GitHub

Fuentes

- Wiki couchdb -> Getting_started_with_PHP
- Google Api Char

Ruben

Tutorial Javafx + Inkscape + Netbeans

Primer tutorial Javafx en adictosalainformatica.com.

JavaFX es un lenguaje de programación de scripts desarrollado por SUN , que nos permite hacer aplicaciones para móviles , televisores , aplicaciones de escritorio (ventana) , y Applets (incrustados en páginas web).

Inkscape (Programa de diseño gráfico vectorial) , nos permite crear un gráfico y exportarlo como código javafx .

Este será el primero de una serie de tutoriales JavaFX , y el primero también de Inkscape.

Utilizaremos Netbeans 6.9 con JavaFX instalado .

Primero crearemos un proyecto en Netbeans en File/New Project/JavaFX/JavaFX Script Application.
Le ponemos el nombre que queramos

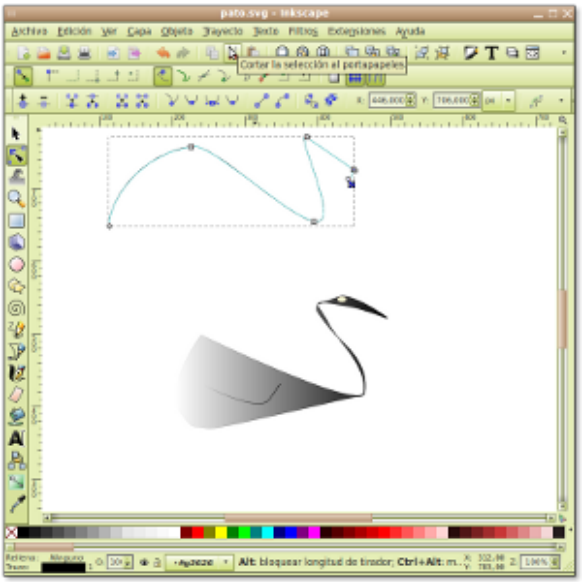
Esto nos genera un proyecto con una carpeta src , un fichero main.fx , lo modificamos para que quede así:

```
Stage {
title: "Adictos a la informática" //cabezera de la aplicación
scene: Scene {
width: 705
height:500
content: [
Text {//caja de texto , opcional
font : Font {
size : 16
}
effect: Reflection {//Opcional , para aplicarle reflejo al texto
fraction: 0.75
topOffset: 0.0
topOpacity: 0.5
bottomOpacity: 0.0
}
x: 10
y: 10
content: "Scuraki" //texto Poned lo que querais , opcional
}
]
}
}
```

Podemos probar como se ve pulsando el botón de run o el de debug.

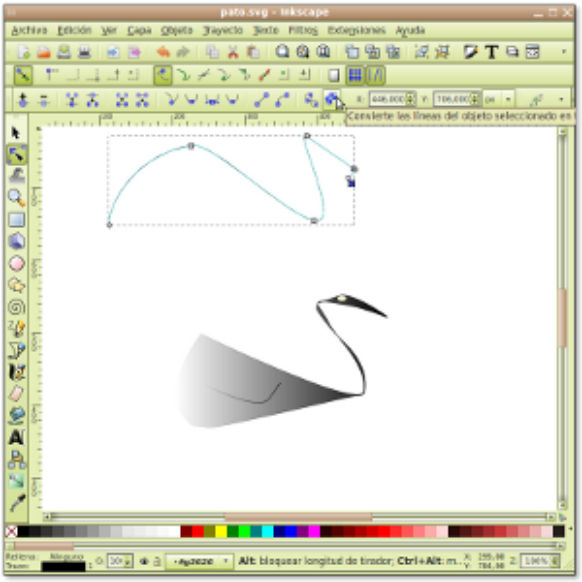
Pasamos a inkscape , y realizamos un gráfico .
En mi caso he dibujado un pato en 1,5 minutos , he trazado una curva,
después pulsando el botón de convertir las lineas del objeto seleccionado en trayectos ,
esto nos permite seleccionar los nodo y crear una figura .Es

posible que tengamos demasiados nodos y
nos sea difícil , si dejamos pocos nodos nos será más fácil , seleccionamos los que queremos borrar y pulsamos la tecla Supr.



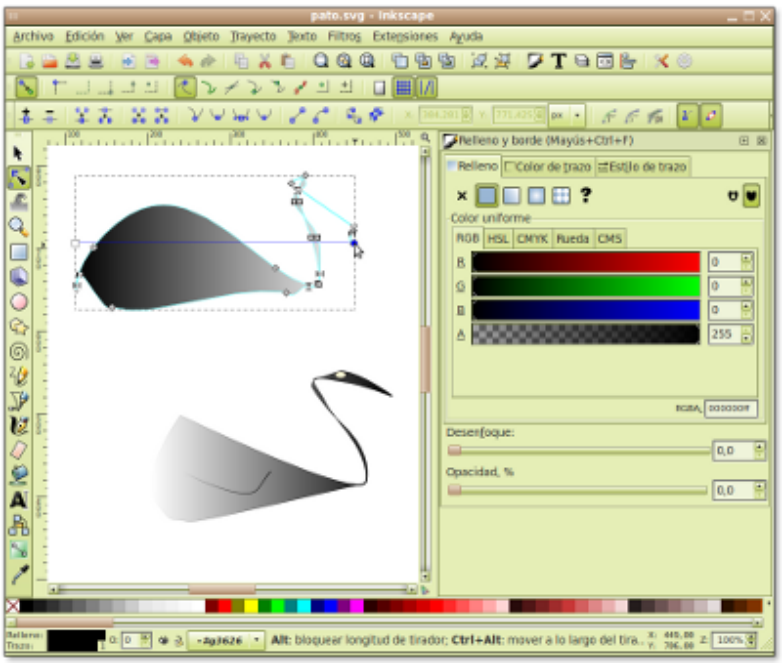
Pantallazo-pato.svg - Inkscape

Para colorear nuestro dibujo en el menú objeto abrimos la opción relleno y borde.
Esto nos abre el cuadro de colores , donde nos permite cambiar el color , utilizar un degradado lineal , un degradado circular , o un patrón .Seleccionamos el degradado lineal .



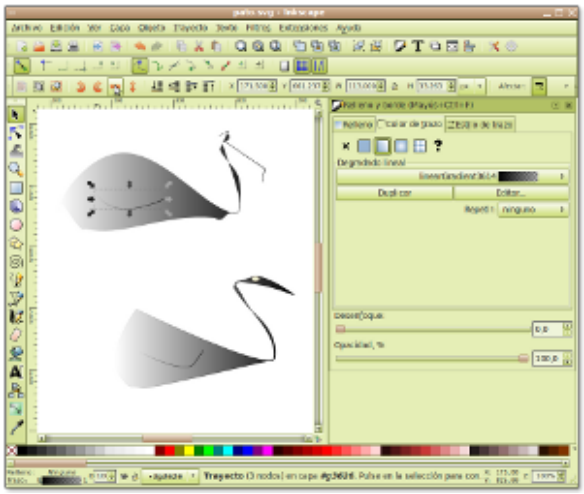
Pantallazo-pato.svg - Inkscape-1

Si queremos podemos girar el degradado como mejor nos quede , en el modo de edición de nodos, en el degradado se ven 2 nodos enlazados con una línea , si movemos los nodos movemos el degradado.



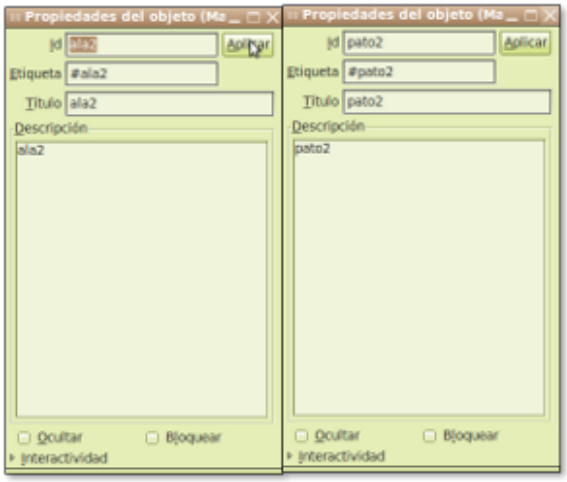
Pantallazo-pato.svg - Inkscape-3

Bueno , tenemos una figura en mi caso es como un pato , le pondré un ala , para que de mejor impresión, dibujo una curva y pongo el degradado que he utilizado con el pato , para ahorrar y la situo en el pato , si es necesario se puede voltear girar , rotar .



Pantallazo-pato.svg - Inkscape-6

Para un mejor seguimiento de las partes de nuestro diseño , con botón derecho en el pato le damos a propiedades , y le llamamos pato , hacemos lo propio con el ala , y la llamamos ala.



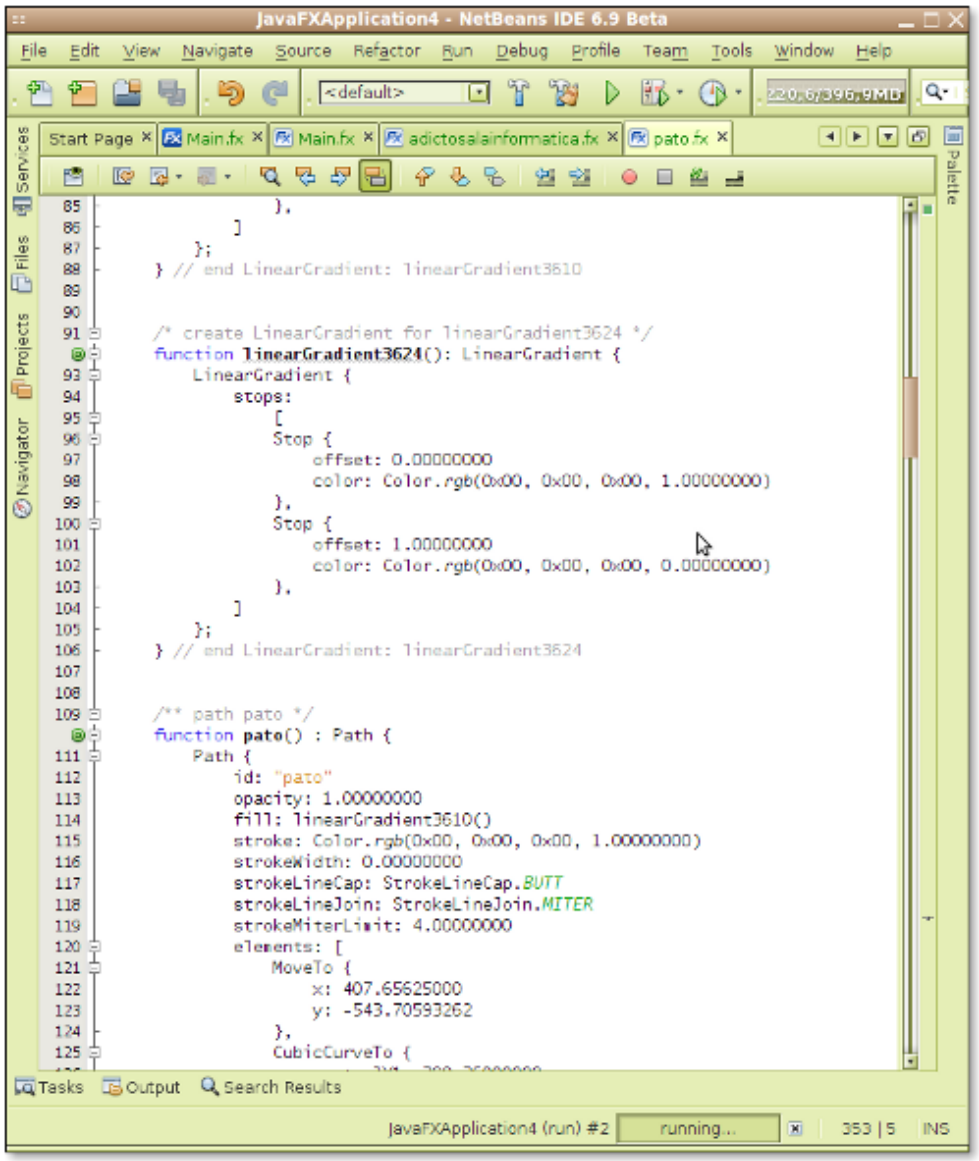
Pantallazo-Propiedades del objeto

Seleccionamos los 2 objetos y en el menú objeto seleccionamos agrupar (Alt + G también agrupa).

Bueno , ya tenemos un pato agrupado , ahora lo vamos a convertir en código JavaFX , guardamos el documento como svg de inkscape para no perderlo, y volvemos a guardar , pero esta vez lo guardaremos como fichero .fx (desplegar el combo y

seleccionar .fx de javafx) , en la carpeta src de nuestro proyecto de Netbeans que hemos creado antes.

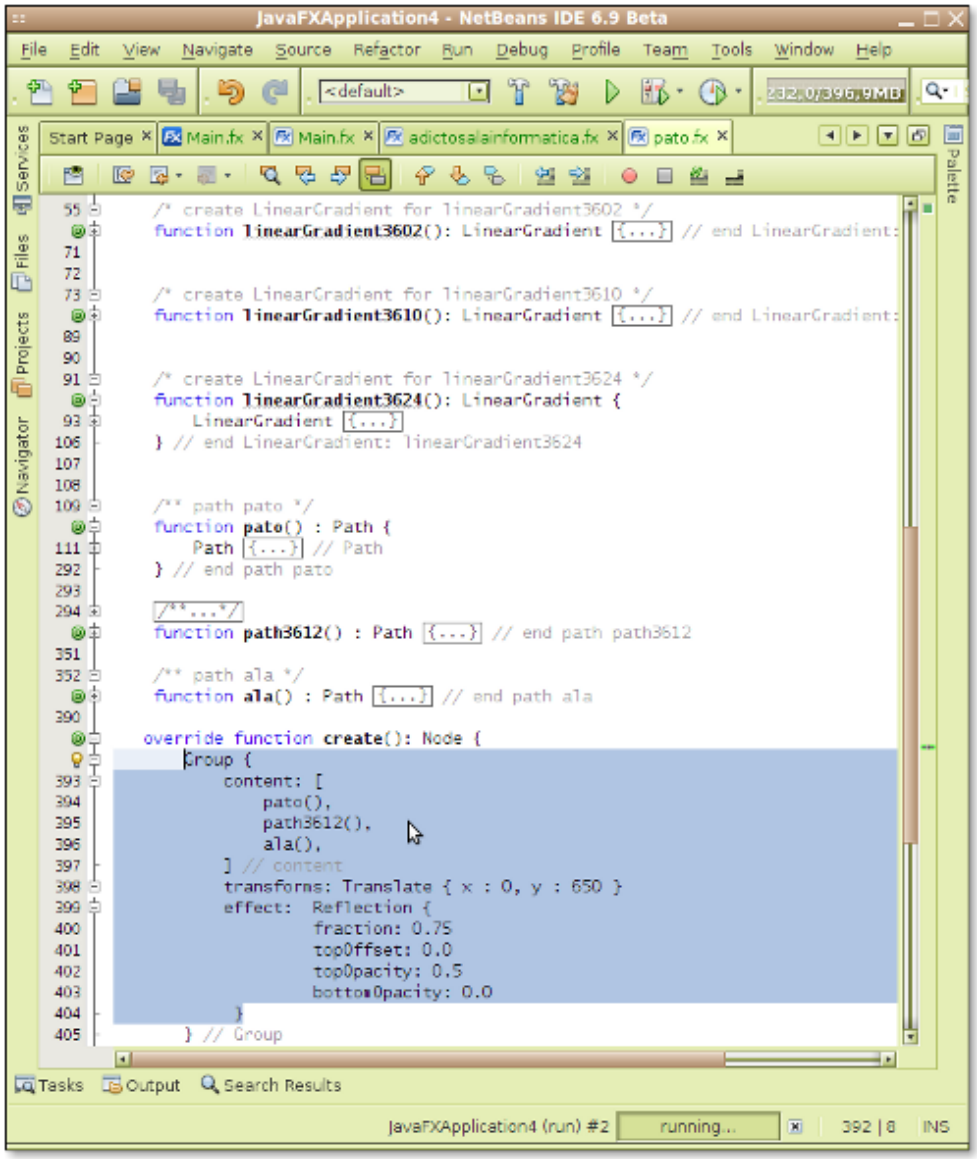
Volvemos a Netbeans , y veremos nuestro fichero fx , en micaso pato.fx.
Editamos el fichero pato.fx , y observamos que hay una clase pato que extiende de CustomNode con un método para cada parte de nuestro dibujo , el pato , el ala , los degradados que hemos creado , un un grupo .



Pantallazo-JavaFXApplication4 - NetBeans IDE 6.9 Beta-1

También hay un método que se llama create que es el encargado de devolver el contenido , en el group , le aplicaremos un

efecto de reflejo , para que quede más bello.



Pantallazo-JavaFXApplication4 - NetBeans IDE 6.9 Beta-2

```
Group { //grupo
content: [
pato(),//dibuja el pato
path3612(),//esto varia segun el diseño
ala(),//dibuja el ala
] // content
transforms: Translate { x : 0, y : 650 } //posicionamos el grupo en la escena , viene con valores
```

```
effect: Reflection { //efecto de reflejo
fraction: 0.75
topOffset: 0.0
topOpacity: 0.5
bottomOpacity: 0.0
}
} // Group
```

Ahora vamos al fichero main.fx , donde llamaremos a nuestro pato para que se pinte por pantalla.
Creamos un objeto de tipo Node , var pato:Node; , y le asignamos el valor creando un objeto pato , pato=new pato(); .

Añadimos el pato a la escena
Ejecutamos el proyecto , y graduamos el ancho de la escena en main.fx Scene , y la posición del gráfico donde os comenté arriba.

En mi caso tengo también 2 gráficos , me quedó así:

```
package javafxapplication4;

import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
import javafx.scene.Node;
import javafx.scene.effect.Reflection;

/**
* @author scuraki
*/
var adicto:Node;
adicto=new adictosalainformatica();
var patito:Node;
patito=new pato();
Stage {
title: «Adictos a la informática»
```

```
scene: Scene {
width: 705
height:500
content: [
Text {
font : Font {
size : 10
}
effect: Reflection {
fraction: 0.75
topOffset: 0.0
topOpacity: 0.5
bottomOpacity: 0.0
}
x: 10
y: 10
content: «Scuraki»
},adicto,patito //añade los gráficos vectoriales
]
}
}
```

Pantallazo-Adictos a la informática

Bueno hasta aquí el tutorial , se puede apreciar algunos errores en el Path del ala y en la posición del degradado , pero puede ser útil ¿no os parece? , espero que os haya

resultado interesante , hasta la próxima.