Un vistazo a Vala

Víctor Manuel Jáquez Leal
vjaquez@igalia.com

¿Asistieron a la charla de Jürg?

- ¿Si?
  - No tengo mucho más que agregar.
  - Gracias.

Historia de Vala

- 1997 nace el proyecto Gnome
- Entre otras razones no usar Qt/C++
  - Actualmente esta razón ya no es sustentable
  - Pero ahora existen dos comunidades con cultura propia

C como lenguaje orientado a objectos

- La biblioteca gráfica de Gnome es *Gtk+*
  - Está implementada usando los conceptos de orientación a objetos
  - Estos conceptos fueron evolucionando hasta la aparición de la biblioteca *GObject*

GObject

- Programación orientada a objetos utilizando C
  - C es "difícil" (administración de memoria manual)
  - GObject es verboso (hay que escribir mucho "boilerplate")
  - Le falta el azúcar sintáctico de un POO (mindset)

Modelo de componentes

- La apuesta tecnológica de Gnome era *CORBA*
  - Ofrecer interacción entre componentes de software independientemente de lenguaje, plataforma de software y de hardware.
  - Componentes *Bonobo*
  - Demasiado complejo^W código "extra"

La otra opción: los bindings

- De manera paralela se desarrollaron "bindings" para otros lenguajes
  - Perl, Python, Scheme, Ruby, Java, etc.
  - Independencia de lenguaje... y nada más.
  - No añaden complejidad
  - Necesitan de mantenimiento

En busca de soluciones

- Miguel ve en .Net la respuesta
  - C# es un lenguaje de programación interesante
  - Una máquina virtual
  - Un conjunto de bibliotecas
    - Código nativo = manejado por la VM
  - Bindings (P/Invoke)

.Net

- .Net promete
  - Mejorar la productividad de aplicaciones verticales
  - Ocultar detalles específicos de la plataforma

Otra solución

- En el 2000 apareció GOB
  - Preprocesador para la generación autómatica de código para GObject
  - Código híbrido: parte C, parte GOB

Vala

- En el 2005 nace Vala
- Es un lenguaje de programación en pleno derecho
- Tiene una sintáxis parecida al C#
- El código intermedio generado por el compilador es C (usando GObject "de manera opcional")

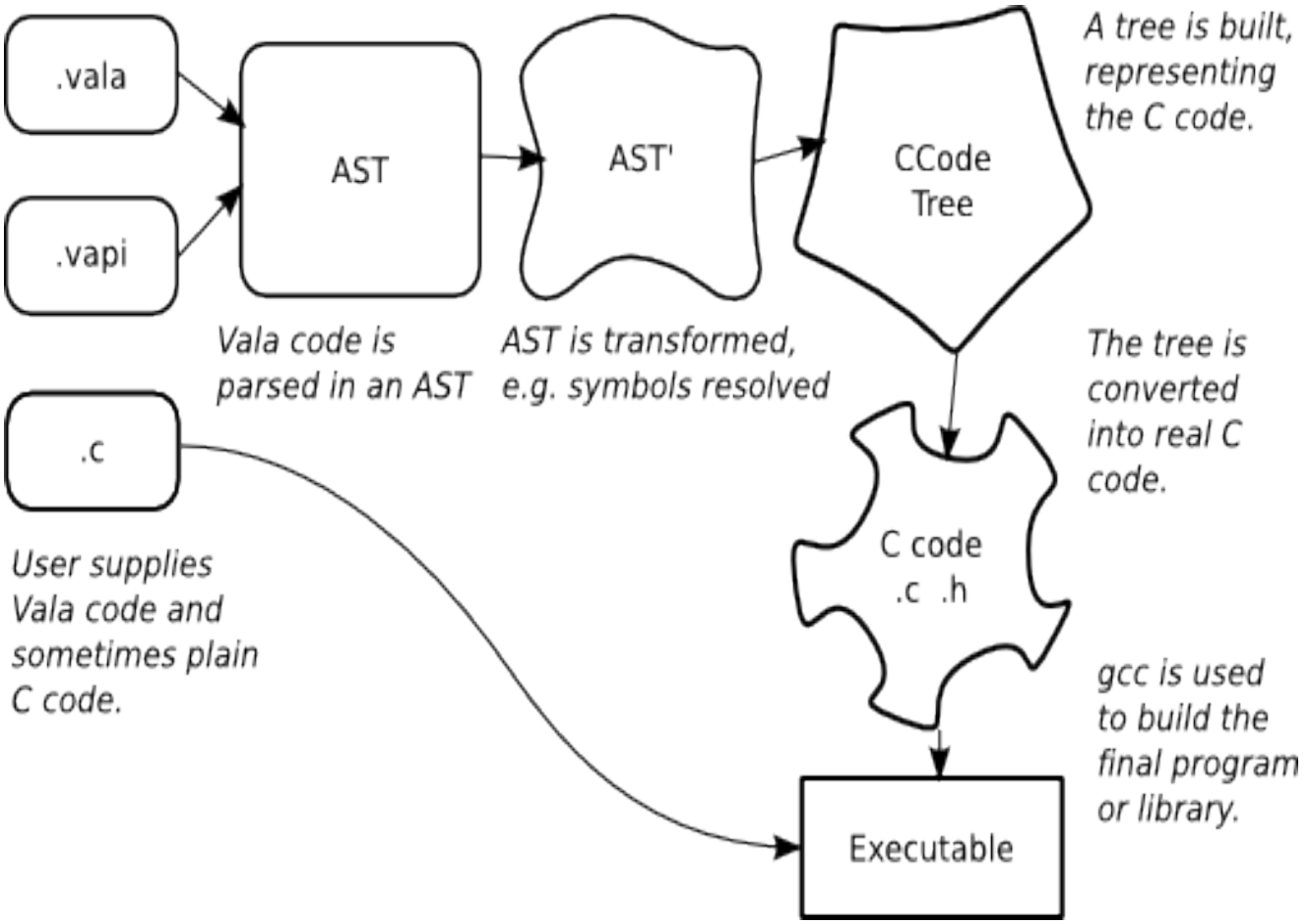
Elementos de Vala

- Interfaces
- Propiedades
- Señales
- Iteradores (foreach)
- Funciones anónimas
- Inferencia automática de tipos
- Programación genérica
- Tipos no-nulos
- Administración de memoria "asistida", *no* automática
- Gestión de excepciones
- Mecanismos para la carga dinámica de módulos (plugins)

Ventajas

- Vala, siendo un lenguaje de última generación, usando el lenguaje C y la infraestructura de GObject, genera programas binarios nativos.
  - Intenta resolver el problema de la "productividad"
  - Facilita el desarrollo de aplicaciones nativas en Gnome
  - Sin usar una máquina virtual

El proceso de compilación



Bindings a bibliotecas externas

- Ficheros VAPI
  - Contienen una descripción de la interfaz pública de una biblioteca
  - Un mapeo entre los símbolos en C y la sintaxis de Vala

Automatizar la generación de bindings

Introspección

- Se está trabajando en mecanismos de introspección genéricos para GObject

  http://live.gnome.org/GObjectIntrospection

- Vala provee su propio mecanismo para la introspección de bibliotecas en Gobject:

  `vala-gen-introspect`

Metadata

- Sin embargo las puras cabeceras en C no contienen toda la información necesaria
  - Anotaciones
  - Custom code

Haciendo bindings para una biblioteca GObject

libgdata

- `libgdata/ligdata.files`
  - Ficheros .h: Las clases, estructuras y métodos
  - Fichero .so: Extrae las propiedades y las señales
- `libgdata/libgdata.namespace`
  - El espacio de nombres de la biblioteca
- `libgdata/ligdata.metadata`

Metadatos (semántica no declarada en C)

- is_out
- is_ref
- no_array_length
- transfer_ownership
- is_value_type
- weak
- hidden
- name
- nullable
- ellipsis
- type_arguments
- errordomain
- throws

Haciendo bindings para una biblioteca GObject (2)

libgdata

- `libgdata/ligdata-custom.vala`
  - Sobrecargas al código generado
- `ligdata/ligdata.deps`
  - Paquetes de los que depende

La introspección en Vala

```
vala-gen-introspect ligdata ligdata
```

- `ligdata/ligdata.gi`
  - Es un archivo en xml
  - Expresa la API pública de la biblioteca

Generando el fichero VAPI

```
vapigen --library libgdata libgdata/libgdata.gi
```

- `libgdata.vapi`
  - Es un archivo en vala
  - Expresa la API pública de la biblioteca
  - Contiene *anotaciones* con metadatos

Anotaciones Vala

- Vala.Namespace
  - CCode
- Vala.Class
  - CCode
  - ErrorBase
- Vala.Struct
  - CCode
  - SimpleType
  - IntegerType
  - FloatingType
- Vala.Interface
  - CCode
  - DBusInterface
    - DbusGProxy
- Vala.Enum
  - CCode
  - Flags

Anotaciones en Vala

- Vala.Method
  - CCode
  - ReturnsModifiedPointer
  - FloatingReference
  - NoArrayLength
  - PrintfFormat
  - Import
- Vala.FormalParameter
  - Ccode
- Vala.Property
  - Notify
  - NoAccessorMethod
  - Description
    - nick
    - Blurb
- Vala.Delegate
  - Ccode
  - NoArrayLength
- Vala.Constant
  - CCode

Anotaciones en Vala

- Vala.Field
  - CCode
  - NoArrayLength
- Vala.Signal
  - HasEmitter

CCode

- El `CCode` influye en cómo Vala generará el código en C

```
[CCode (argument0=value0, argument1=value1, ...)]
```

- array_length_pos
- cheader_filename
- cname
- const_cname
- copy_function
- cprefix
- free_function

Bindings para bibliotecas sin GObject

- Se hace el VAPI a mano

Preguntas

Gracias