

li atio e elo me t it E je

Title: Applicati onel pme t wit E e
Su title

Content

1 Background

Structure of the philosophy application **7**

2.1 Decomposition of the architecture

Introduction to Edge **11**

3.1 The flow of data 13

3.2 Available libraries 1

The fundamentals in practice **17**

3.1 Working with the canvas 1

3.2 Interacting with the objects 2

Introduction to objects

Chapter 1

Chapter Introduction

Structure of a graphical application. If we have a set of data, we can use it to create a many possible applications available, we can use it to create a many possible applications available. This chapter introduces the reader to the structure, the concept of a Graphical User Interface (GUI) and how it deals with the interaction between the user and the application.

Decomposition of the GUI. There are three approaches to the creation of a GUI-

apter

Structure of graphical application

So... you want to create a GUI application? I assume so since you chose this book as instructive, or at least a starting point, material. You could call the "GUI" a "library" that feels overwhelmed by the large number of development libraries available. As you look through the sources (your preferred source) I guarantee you will realize that all of them, and the applications that use them, share a common structure. This is the 27 chapters of the 2nd edition of the book.

.1. C IN H FR N N

language. In the case, a AP was used by the application level per the assembly interface elements from more primitive objects, in the tier case the AP was used to include objects already defined by a library. A y class is a interface object by simplistic style modifications that be submitted by the user to the level per. May utilities were only slightly less difficult than behavior classes. The idea of working with the interface objects and their composition in a live environment, like a web level per es, was pretty much utilizable.



T ay, Ec re e c mpasses a l list i m ules pr perly amespace a preixe wit

Chapter 4

The foundation in practice

Why pick up the Erlang Foundation libraries are designed in a way that makes it easier to write code that is more idiomatic but it simplifies the mental image of the structure of the Erlang API in the reader's imagination.

With the pitchforks are back in the bar all we need to put it in more clear terms with a simple example:

```
Erlang > button = gen_server  
button = gen_server_add(gen_server, "button")  
gen_server_file_set(button, "the server", "button")
```

This is a simple snippet that could be translated into a more tactical speaking ()
better to be like Python as:

```
button = gen_server()  
button.file_set("the server", "button")
```

The differences between the two snippets could be seen

itservtest me t p2T 32 p2TF()-33p2T bar s. u 33 file ua e me t 21)- 33 bul)]TJlas

y default, Ec re aware ess is limite t system si als like HUP r K ■■. A iti al

