doorbell Documentation

Tim Hartman

Nov 09, 2018

Contents:

You have a visitor.

doorbell provides a visitor pattern implementation. This implies two basic classes, a `Visitor` and a subject that is visited, the `Visitee`.

Implementations of `Visitee` are mainly left to the user, while doorbell seeks to provide a number of `Visitor` classes for various purposes.

CHAPTER 1

Usage

1.1 The Visitee

doorbell provides `Visitee`, an abstract base class with a single method, `Visitee.accept()`. Implementations of `Visitee.accept()` typically only consist of a single line:

```
def accept(self, visitor):
   return visitor.visit_MyType(self)
```

where *visit_MyType* is the method on the visitor which applies to this particular object. Typically, only the object (*self*) is passed, although any arguments will be passed along to the visitor's method.

1.2 The Visitor

The base `Visitor` class and its children are the main products of doorbell. Your visitor class inherits from `Visitor` or its children, and implements a set of methods which are called from a `Visitee.accpet()`. By default, any method whose name begins *visit_* is considered a visitor method. However, the decorators:

- `visitor_method()`
- `non_visitor_method()`

override this default. Any method decorated with `visitor_method()` will be considered a visitor method, while any method decorated with `non_visitor_method()` will *not* be considered a visitor method. All visitor methods are wrapped by `Visitor._visit_method()`.

The following visitor classes are provided:

`Visitor`
`CascadingVisitor`
`WrappingVisitor`

CHAPTER 2

API

doorbell

CHAPTER 3

Indices and tables

- genindex
- search