Epydoc / reStructuredText

Antonio Espín Herranz

Epydoc y reStructuredText

- **pydoc** es que es **muy simple**, y **no permite añadir semántica o modificar estilos** de la documentación.

- **EpyDoc**
  - Es una de las herramientas de **generación de documentación** para Python más utilizadas.
  - **Soporta texto plano** y un formato propio: **epytext**, soporta **reStructuredText** y sintaxis **Javadoc** (al estilo **Java**).

Epydoc y reStructuredText

- **Epydoc**
  - Se puede descargar desde su página web en forma de instalador exe para Windows, paquete RPM para Fedora o similares, o en archivos zip y tar.gz que incluyen scripts de instalación: http://epydoc.sourceforge.net/
  - https://pypi.python.org/pypi/epydoc
  - También se encuentra en los repositorios de varias distribuciones Linux.

Epydoc y reStructuredText

- El script **epydoc**, que consiste en una aplicación de **línea de comandos**, y el script **epydocgui** (epydoc.pyw en Windows), que ofrece una interfaz gráfica.
- Además también podemos acceder a la funcionalidad de epydoc **programáticamente**, como en el caso de pydoc.

Ejemplo

```
"""Modulo para ejemplificar el uso de epydoc."""

class Persona:
    """Mi clase de ejemplo."""

    def __init__(self, nombre):
        """Inicializador de la clase Persona."""
        self.nombre = nombre
        self.mostrar_nombre()

    def mostrar_nombre(self):
        """Imprime el nombre de la persona"""
        print ("Esta es la persona %s" % self.nombre)

class Empleado(Persona):
    """Subclase de Persona."""
    pass

if __name__ == "__main__":
    raul = Persona("Raul")
```

Formato por defecto HTML
epydoc ejemplo.py
o bien
epydoc --html ejemplo.py

epidoc

- epydoc **--opción** ejemplo.py

- **Opciones**:

  -- **html**: por defecto en HTML.

  -- **pdf**: Genera en PDF pero hay que instalar **LaTeX**.

  --**graph classtree**: generar un gráfico con las clases.

reStructuredText

- **Funcionalidades**:
  - *itálica* -> *itálica*
  - **negrita** -> **negrita**
  - “monoespacio” -> monoespacio
  - * es un carácter especial -> * es un carácter especial.

- **Para describir propiedades de los elementos que estamos documentando se utilizan los campos o fields.**

- En **reStructuredText** los campos **comienzan** con **‘:’**, le sigue el nombre del campo y opcionalmente sus argumentos, y se **cierra** de nuevo con **‘:’**, para terminar con el cuerpo del campo.

- Lista de CAMPOS que soporta Epydoc:

Campos Epydoc

Funciones y métodos	
`:param p: Un parámetro`	Describe el parámetro p.
`:type p: str`	Especifica el tipo esperado para el parámetro p.
`:return: True si son iguales`	Valor de retorno.
`:rtype: str`	Tipo del valor de retorno.
`:keyword p: Un paráme-tro`	Descripción del parámetro con valor por defecto y nombre p.
`:raise e: Si el paráme-tro es cero`	Describe las circunstancias para las que se lanza la excepción e.

Campos Epydoc

Variables	
`:ivar v: Una variable`	Descripción de la instancia v.
`:cvar v: Una variable`	Descripción de la variable estática de clase v.
`:var v: Una variable`	Descripción de la variable v del módulo.
`:type v: str`	Tipo de la variable v.

Campos Epydoc

Notas	
`:note: Una nota`	Una nota sobre el objeto.
`:attention: Importante`	Una nota importante sobre el objeto.
`:bug: No funciona para el valor 0`	Descripción de un error en el objeto.
`:warning: Cuidado con el valor 0`	Una advertencia acerca de un objeto.
`:see: Ver 'Python para todos'`	Para indicar información relacionada.

Campos Epydoc

Estado	
`:version: 1.0`	Versión actual del objeto.
`:change: Versión inicial`	Listado de cambios.
`:todo: Internacionalización`	Un cambio planeado para el objeto.
`:status: Versión estable`	Estado del objeto.

Campos Epydoc

Autoría	
`:author: Raul Gonzalez`	Autor o autores del objeto.
`:organization: Mundo geek`	Organización que creó o mantiene el objeto.
`:license: GPL`	Licencia del objeto.
`:contact: zootropo en gmail`	Información de contacto del autor.

Epydoc

- Para que Epydoc sepa que utilizamos **reStructuredText** es necesario **indicarlo (dos posibilidades):**
  - Variable **__docformat__** en el código, o bien
  - La opción **--docformat** de línea de comandos.
    - Las opciones posibles son:
      - epytext,
      - plaintext,
      - restructuredtext.
      - javadoc.

Ejemplo

```
"""Modulo para ejemplificar el uso de *epydoc*.
    :author: Raul Gonzalez
    :version: 0.1"""

__docformat__ = "restructuredtext"

class Persona:
    """Modela una persona."""

    def __init__(self, nombre, edad):
        """Inicializador de la clase `Persona`.
        :param nombre: Nombre de la persona.
        :param edad: Edad de la persona"""
        self.nombre = nombre
        self.edad = edad
        self.mostrar_nombre()

    def mostrar_nombre(self):
        """Imprime el nombre de la persona"""
        print ("Esta es la persona %s" % self.nombre)
```

Ejemplo

```
class Empleado(Persona):
    """Subclase de `Persona` correspondiente a las
       personas que trabajan para la organización.
    :todo: Escribir implementación."""
    pass
    if __name__ == "__main__":
        juan = Persona("Juan", 26)
```

reStructuredText

- Soporta un **segundo tipo de campos** en el que el cuerpo del campo es una lista. De esta forma podemos, por ejemplo, describir todos los parámetros de una función o método con un solo campo :**Parameters**:, en lugar de con un campo :**param**: para cada parámetro.

- class Persona:
  - “””Modela una persona.”””

  - def __init__(self, nombre, edad):
    - “””Inicializador de la clase `Persona`.
    - :Parameters:
      - - `nombre`: Nombre de la persona.
      - - `edad`: Edad de la persona.
    - “””
    - self.nombre = nombre
    - self.edad = edad
    - self.mostrar_nombre()