Examples of Code Listings in LaTeX

Andrew Pinkham

July 1, 2013

Abstract

This project is meant to help jump start writers looking to display source code in LaTeX documents. The project provides a set of TeX files and a PDF of the output of said TeX files. The intention is that a reader may view the various code listing demonstrations in the PDF, and then read, learn, or simply copy the LaTeX necessary to display their code in the same manner. Basic knowledge of LaTeX is assumed.

We display code using the following: `verbatim`, `listings` (not to be confused with `listing`), and `minted`.

To avoid a monolithic LaTeX document and make perusing the code simpler, each subsection is it's own TeX file, as is the preamble.

Note that all code is licensed under the Simplified BSD License, a copy of which is included with the project code, found on github.

Contents

List of Code Examples

1 Basic Tools and Techniques for Listing Code

1.1 `verbatim` Environment

The `verbatim` environment, supplied by LaTeX, is the most basic environment a writer may use to display source code. Below is an example.

```
from django.db import models

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

The `verbatim` environment is important, as it is the basis for all other code listing environments demonstrated in this document. This means that packages that affect `verbatim` listings will also affect these other listings.

For example, the `upquote` package, which allows for normal single quotes usage in `verbatim` environments (as opposed to LaTeX's backtick-quote combination), will work in `listings` and `minted` environments, because they are built on top of `verbatim`. Read more about `upquote` at CTAN.

1.1.1 `verb` Command

Along with the environment, LaTeX supplies the inline `\verb` command, which is used on and off throughout the document to format the names of packages and environments.

Note, however, that the `\verb` command should be used in neither headings (such as `\section`) nor captions. Use of `\verb` in captions causes compilation errors. Use of `\verb` in headings will cause problems with the `hyperref` package, and may lead to over- or under- full box errors.

Instead, use the `\texttt` command, as demonstrated in the headings of this document.

If you absolutely must use `\verb` in a caption, you may import the `cprotect` package, which you can read more about on the documentation provided by CTAN, and precede the caption with the `\cprotect` command. Code Example 1.3.3 on page 4 demonstrates usage.

1.2 listings Package

The following uses the listings package to create an environment.

Code Example 1.2.1: models.py from Django Tutorial using Listings

```
from django.db import models

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

On top of an environment, the listings package also provides a way to format inline text using \lstinline{}, replacing the verb command. Note that the settings for \lstinline{} must be set in the options of \lstset{} in the preamble. For more on the listings package, please see the wikibook on the subject, or else the documentation provided by CTAN. Note that while not discussed here, it is possible to create a list of listings (a table of contents of all the listings).

1.3 minted Package

The last environment I'll demonstrate is the minted package. It takes code in the environment, and runs it through Python's pygments library. The resulting colored LaTeX syntax is placed in a verbatim environment.

```
from django.db import models

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

While minted does not provide an inline formatting command, it does provide a shortcut command mint for short code snippets.

```
fib = lambda n: n if n < 2 else fib(n-1) + fib(n-2)
```

The minted environment also provides the ability to be inserted into listings, with labels and captions.

```
from django.db import models

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

Code Example 1.3.3: models.py from Django Tutorial using Minted

2 Custom Environments and Macros

This section will demonstrate various macros defined in the preamble that you can use to make your document clearer.

2.1 Aside

While not for code, per se, this aside environment demonstrates the use of the `mdframed` package.

Aside - Lorem Ipsum Example

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

2.2 Python Code Macro

Here is an example of the macro in the preamble for listing Python code.

Code Example 2.2.1: `models.py` using Minted Macro
File: `/models.py`

```python
from django.db import models

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```