O'REILLY®

# Book Proposal

## Proposed Book Title:

The Djen of Django

(Django is pronounced as Jango, so we can promote this as "The Zen of Django" )

## Proposed Book Subtitle:

Building practical projects with Django

**Author(s):** Shabda Raaj

**Author title(s) and affiliation(s)**

-

**Preferred mailing address(es):**

**Flat No 2,**

**COD Quarters,**

**Madhapur,**

**Hyderabad,**

**India**

**Preferred Email address(es):**
**shabda.raaj@gmail.com**

 

## Book Summary:

*In one sentence, tell me why the audience will want to buy your book?*

This book walks the user through building various applications, and with them shows the user techniques/best practices/hard to find information. Django is targeted at professionals, who are building web applications every day. In this book, while building the apps, we will solve various problems, which the user can refer back to.

*Summarize what the book is about. Please imagine that this text will appear on the back cover of your book. Also, please indicate why your book would be unique in the market place.*

[Text which might appear on the back cover of the book.]

Django, the web framework for perfectionists with deadlines. Build seven useful applications, and learn the hidden nooks and corners of Django. Filled with practical advice, techniques and best practices.

[Why the book would be unique in the marketplace.]

There are a few existing books about Django, however both of the popular ones are reference/tutorial books. This book would show how to build applications by building fully self contained applications.

## Technology Summary:

*How would you characterize the technology's stage of development (put an X in the column next to the stage that best applies)?*

| Answer | Stage | Description |
|---|---|---|
| | *Emerging* | *This is a new and cutting edge technology that is still in the experimental/lab stage.* |
| ***X*** | *Developing* | *A technology that is gaining momentum and has a lot of activity but is not a clear standard. High growth, but multiple competitors on the horizon.* |

 

| Answer | Stage | Description |
|---|---|---|
| | *Mature* | *An established technology with leading market/mind share. Strong or moderate growth.* |
| | *Declining* | *An old titan that may still be important and useful, but there is no further expected development work or growth. Alternative approaches threaten to make it obsolete.* |

*Briefly explain the technology and why it is important to computer professionals.*

Django is a python based web framework, which emphasizes rapid development, adhering to DRY principle, and automating commonalities in web development.

It is used for building web apps, and though it is flexible to handle many tasks, it shines when it is used for building content rich database backed web apps. It provides a powerful ORM, to abstract the interface with the Database, yet it allows to use raw SQL for the cases when going via the ORM is costly or infeasible. It provides a sandboxed templating language, so that business logic does not get mixed with the presentation logic. With other features like Admin, which provide full set of CRUD operations for defined Entities, NewForms to make interacting with Html forms easy Django works hard to abstract away the commonly occurring operations in web development.

## Audience:

*Explain who the primary audience is for your book. What professional positions does this audience hold?*

People who are using Django to build web apps professionally. They may be employed by Development shops, using Django, independent consultants, or PHP developers who want to move to Django.

*What knowledge do you assume of this audience?*

 

We assume knowledge, and reasonable mastery of Python. A basic knowledge of Django is also assumed. In particular, we assume they have at least read tutorial at Djangoproject.com. However we do not assume they have read all of Djangobook.com, or the whole Django documentation at Djangoproject.com. A basic familiarity with HTTP methods, (how does GET differ from POST) is also assumed.

*What books can you assume they have read?*

1. A book introducing Python. Either the python tutorial or Dive into Python, or a equivalent book is sufficient.
2. Parts 1-4 of Django tutorial.

*What skills can you assume they have mastered?*

1. Python
2. Basics of Django
3. Basics of how HTTP works.

*Please estimate as best you can how many people will use this technology? Please state any applicable statistics [web searches, web site traffic, blogs] indicating market use or market growth.*

It is very difficult for me to give a exact market size, however here are a few statistics,

1. 8534 members in Django-users Google group.
2. Around 100 posts every day. Total of around 50000 posts.
3. 3260 memebers in Django-developers Google group. Most of them very active.
4. Google's Appengine supports Django out of the box(excepting the ORM). Appengine got 10000 users within 24 hours of release, most of these people would be using Django.
5. Google trends. Please notice the recent surge.

*Please provide some scenarios that indicate how the audience will use your book. For example, will readers refer to it daily as a reference?? Will they read it once to learn the concepts and then refer to it occasionally?*

 

Scenario 1. User reads the whole book. He is building a CMS, where changes to the tables must be logged, and entities must be versionable. He refers back to the Chapter where we built a wiki and showed this specific techique.

Secnario 2. User is trying to learn about Newforms. In particular they might be building a "quick entry form", where multiple forms of a type exist on a page, and user needs to fill only as many as they need. They refer back to chapters where we introduce newforms, and advanced newform techniques, to see how to do this.

Scenario 3. User has been hired to create a custom Blog application. They read the chapter on Blog application, and use this as starting point.

Scenario 4. User has read a few books on Django, but wants to learn more by reading up code.

*Use the following table to describe where the audience for your book typically gets its information and to whom it looks for guidance and leadership.*

| | |
|---|---|
| What websites do they use? | Www.djangoproject.com, groups.google.com/django-user, |
| What publications (magazines, journals, newspapers) do they read? | There is no dedicated publication for Django. So Python magazines would be good way to target this audience. |
| What conferences do they attend? | Similarly to previous question, PyCon would be a good place to target this audience. Many of these people would be interested in Open source, so Osconf would be another good place. Many people in Rails community would be looking to widen their skill set, especially the may be looking to deploy their apps to Appengine. So targeting Railsconf might be another good place. |
| What tradeshows do they attend? | |
| To what professional organizations do | In this PyCon, setting up of Django Software Foundation was announced. However the paerwork |

 

| | |
|---|---|
| they belong? | for this has not been completed. |
| Who are the leaders and key influencers in the field who you feel might make great reviewers or endorsers of your book? | Adrian Holovaty, Jacob-Kaplan-Moss, Malcolm Tredinnick, James Bennett (However he is writing a book with same audience) |

## Key Topic Coverage:

*What are the top five topics that will be covered in the book? Why are they the top five?*

1. Power uses of Django ORM: Django ORM is extremely powerful, and can do many thing for which it seems that we would need to drop to SQL. Exmples of how this can achieved would be very useful. However there are some cases when using SQL directly would be the best option. This topic would explain when this should be done.

2. Performance tuning Django applications: When ORM is doing interfacing with the Database, it becomes difficult to understand what is going behind the scenes. If you are not careful, what you though would fire a single query can end up firing ten or twenty, and bring the application to its knees. As for webapps, time spent in the database is largest, optimizing the queries can improve the performace very much.

3. Writing Ajax applications with Django: Django has no in built Javascript helper library, it just has *simplejson* module, using which javascript objects can be sent and received. However using any javascript library, writing Ajax applications is very easy. This topic explains this, and the best practices associated with this.

4. Writing reusable Django applications: Django applications are meant to be reusable. So for example, the registration module you develop for one application should be able to be reused for the next project, if you follow certain simple guidelines. By showing some such modules we describe how reusable applications are built.

 

5.

*What problems does this book solve for its users?*

There are many features which can solve common web development tasks. And these features are also well documented. Yet unless the users see how these features are used in real life, it is difficult to gain an appreciation of how to use them. For example Django's signals can be used to build a plugin system, but it is a litle hard to see how to use this feature, unless you have saw code which does this.

*List the four or five topics covered or features included that will provide the greatest benefit to readers or will be the most likely to excite them?*

1. Using Django with Appengine: With launch of Appengine, Django has become very easy to deploy. However Django ORM does not work with Appengine. So we need to work around this.

2. Example of using existing Django applications: In various chapters we will reuse various existing Django apps. This will give the user a knowledge of the apps they might be most likely to use, and also teach them how to create reusable applications.

3. Integrating Django with Amazon S3: S3 is a very cost effective way to store and serve large files, and with Django not *yet* having streaming file uploads, is the only viable option, if the files are large. This topic would explain how to  file management with S3 in face of authetication requirements.

4. This book would have examples of working code which the users can just take out and reuse, or build upong to fit their needs. So when they need to build, say a wiki module, reading the chapter where we build the wiki would be the first stop.

## Other Book Features:

*Is there a companion web site? If so, what do you plan to include on the site?*

Djenofdjango.com

 

Django is still an emerging technology, and the APIs are not set in stone. In particular there are two branches of Django which may be merged soon in the trunc. There are Backwards incompatible changes done sometimes. So this site would track the changes needed with these changes in Django. We can also have forum to discuss the book.

## Competition:

*What books or on-line resources compete with this book? Please list title and author. In each case, how will your book be different or better in timing, content, coverage, approach, or tone?*

The Django Documentation: Authors: Various cotributors to Django. Django has a very good and comprehensive documentation. However this is a reference manual, and fills a different need.

Djangobook.com: This is a book by Adrian Holovaty and Jacob Kaplan-Moss, lead developers of Django. This is also more into reference/tutorial of each feature in Django. This does not show build full applications, so focus of our book is different.

Teach your self Django in 24 hours: Author: Brad Dayley. I have not read this book, but from the title I infer that this an introductory book.

Practical Django Projects: Author: James Bennet and has yet to be released. This book targets a similar market to the Book I am proposing. Without reading the book it is difficult to say how can we be different form this book. Yet we would be solving a different set of problems in this book. Integrating with Appengine, Django for the PHP programmer, Django for the JSP programmer chapters would make this title attractive to a different set of people.

## Author Biography:

*Why are you uniquely qualified to write this book? Why are you the best person to be writing this book compared to others who might approach O'Reilly or other publishers?*

I have been working with Django for an year. I worked at Oracle, from June 2006 to Jan 2008, before leaving Oracle to work full time on Django.

 

After leaving Oracle I have been involved with building two Django sites Dashbard.com and 42topics.com, both are still works in progress. Before that I built 7days7apps.com, a proof of concept to build 7 useful Django applications in 7 days. You can see some of my open source applications at code.google. Or you can read some of my recent Django tutorials 1, 2, 3 (Access 3 using orel/orel)

Using the Applications I developed in 7days7apps.com, I can reuse the code for this book. I am extremely proficient with Django now, yet I learnt Django proficiently only about six months back. So I can still remember which areas were difficult to understand, and stumbling blocks, and explain those areas in detail.

## Book Outline:

*Until we can envision your book in exceedingly concrete terms, we cannot know whether to sign up for it. The outline should include:*

** An explanation of the rationale for the overall chapter organization of the book.*

** A statement of purpose in the form of a paragraph per chapter outlining the goals and topic coverage of the chapter.*

- *A fine-grained breakdown of the contents of each chapter, including, at a minimum, two levels of headings (but see below).*

Rationale for organization of the chapters.

---

We assume that the readers have read the Django tutorials, and so just rush through introductory chapters, giving pointers where to get help if they need it.

The chapters each introduce a set of features or areas of Django. We organize the chapters so that simple apps come before complex apps. In earlier chapters, we introduce the simpler and more frequently used concepts.

 

The model which I would like to follow in this book is of Mark Pilgrim's `Dive into Python`, where at the beginning of the chapter we give the most important code. Then the rest of the chapter dives into and tackles the new concepts introduced in the book.

Chapter 1: Introduction

---

Downloading and Installing.

Why Django . Why Django instead of any other framework. Why Django instead of PHP.

Tutorial. Quick overview. Refer reader to Django tutorial if they are new to Django.

Chapter 2. Building a personal CD library.

---

(In this chapter we introduce Django ORM and Admin. We have not yet introduced Templates so we do not write any custom views.)

Diving in. [Code listing]

Introduction to Django ORM.

Declaring your entities. (How to define the models.)

Talking to the database. (Filter, get, exclude)

Introduction to Django Admin.

Using Django's autogenerated Admin.

Extending the Admin.

 

Chapter 3. Building a Pastebin.

---

(Topics introduced: URL configuration, Templates, Generic views, Standalone Django scripts)

Diving in. [Code listing]

URL Configuration. [Of course the previous chapter had, but that would just have one entry, to Admin's urlconf. In this chapter we explain writing URLConfs.]

How URLConfs map views to Urls.

Passing variables to views.

Using Templates

Templates are not programming language.

In build templates tags and filters.

Template inheritance

Using generic views.

Why generic views.

Build the same views without generic views and show the difference.

Detailed explanation of Generic views used in this chapter - django.views.generic.create_update.create_object and django.views.generic.list_detail

 

Standalone Django Scripts

(We would like to delete old code pastes, so we create a standalone script)

Chapter 4. Building a Blog

---

(Topics introduced: Authentication, Session management, NewForms, Generating simple RSS feeds. Date based generic views.)

Diving in. [Code listing]

Authentication. [In chapter 2, all views were managed by Admin, so we did not need to handle authentication. In chapter 3, all views would be public, so no authentication was needed. In this chapter we need to restrict access to only logged in user, so we introduce it here.]

- Using django.contrib.auth
- Using login_required decorator to restrict access to logged in users.
- Using request.user in views, for finer control over views.
- Using context processors to use logged in users in templates.

Session Management. [So once user has commented, their name/email information is stored.]

- The machinery behind sessions framework.
- Using cookies.
- Using session to abstract handling cookies.

 

Newforms. [Comment, Post, Settings form]

Using newforms.

Using model form to auto generate forms for model.

Creating complex forms programatically. (Instead of defining them.)

Generating RSS feeds.

Using django.contrib.syndication to generate feeds for the Blog.

Using date based generic views to generate monthly and weekly archives for the blog.

Chapter 5. Building a Wiki

---

(Topics introduced: Managing user registration using django-regsitration. Advanced ORM tricks. Overriding save for Entities. Signals. Websearches using Yahoo Devloper API.)

Diving in. [Code listing]

Managing user registration using django-regsitration.

In the previous chapters we did not allow external users to create a user. Here a user can create an account on the wiki. This is managed using django-registration.

Explanation of Django registration, its views and templates.

 

Discussion of making reusable Django applications, taking Django-registration as example.

Advanced ORM

model.extra, How to use it to handle complex queries.

Defining customer manager for models. (With a wiki, we reattach model.objects to a manager which only gets the latest object. Another custom manager, model.allobjects gets us all the elements.)

Manager methods vs. classmethods.

Overiding save, delete amd other model objects to enable versioning for objects.

Introduce signals. (However we would not have used this in our code.)

Show how signals could have been used for the same purpose, and compare.

Websearches using Yahoo Developer API.

We want to make the wiki searchable, so instead of going via a local search system like Lucene, we use Yahoo Developer API. In this we can explain to use django.util.simplejson, to talk to external APIs which provide a JSON interface.

Chapter 6. Building a Yahoo Answer's like site

---

(Topics introduced: Transactions, Middleware, Permissions, Messages.)

Diving in. [Code listing]

 

Transactions. (Till previous chapters we were not using transactions and all database actions were in Autocommit mode. Here we want many views to work as part of a transaction.)

Using TransactionMiddleware to tie Http requests to transactions.

Finer grained control over transaction using commit_manually

Permissions. (We would define various user levels, and different user levels have different permissions)

Introducing permissions.

Creating cutsom permissions.

How Groups, Users and permissions work together.

Using permission_required decorator.

Message. (For example when the user gets a reply to her question.)

Using get_and_delete_messages()

Chapter 7. Building a Project management application

---

(Topics introduced: File uploads, Integrating with Amazon S3, Complex RSS feeds-builds on chapter 4, Generating graphics using PIL. Sending Email with Django. Generating PDFs for pages. Exporting Data.)

 

(This and the next chapter would have larger amount of code than the previous chapters. These chapters would be a rehash of previous chapters, and would show how all these concepts work together.)

Diving in. [Code listing]

File uploads. (We allows users to upload files in this app.)

- Using the Django file widget to upload files.
- The problem with large files.
- Using S3, as a file store.
- Restricting access to S3 files, using Django authentication.

Advanced RSS feeds.

- Generating RSS feeds per project.
- Password protecting RSS feeds.

Using PIL. (We generate charts for the project, so we use PIL)

- Using PIL to generate charts for the project.

Sending email.

- The logs for the project are sent to the user via mail.

Genrating PDFs.

 

The reports for the project are available as PDF. This is done using HTML2PDF library.

Exporting PDF.

The data for a project is accessible in CSV format. Here we show exporting of a data on a per project, or a more granular level.

Chapter 8. Building a Social news Site

---

(Topics introduced: This chapter uses techniques learnt in previous chapters, and introduce Caching and Testing.)

Diving in. [Code listing]

Introduce caching.

The various caching backends.

Page level caching.

More granular caching.

Introduce testing for Django.

Testing Django models using doctests, and unittests.

Testing Django views.

 

Walking though the code.

In this chapter, we walk through the code, to see how all these fit together.

Performance tuning the code.

Logging the queries used, through a middleware.

When select related makes sense.

Profiling Django applications.

Appendices.

---

Using Django with Appengine

---

How Appengine differs from Django.

Introduction to Appengine ORM.

Rewriting sessions to use Appengine ORM.

How other constructs be ported to Appengine ORM.

Using Appengine User class instead of Django User.

Getting back login_required decorator, and why you might not need it.

Django for the PHP programmer

---

How PHP differs from Django.

 

Django templates are not programming language.

Mapping PHP constructs to Django.

Django for the JSP/Sevelet programmer.

---

How JSP differs from Django.

How Servelets differ from Django.

How do doGet, and other do* map to Django.

Mapping JSP/Servelt constructs to Django.

Hacking Django Django internals.

---

[Here we would some of what goes on behind the scenes.]

Templates.

How they work.

Swapping out Django template to use another templating engine.

ORM.

How does Django use metaclasses to generate a class from Model class declared by you.

[This would be a high level overview where we would like a topic to be introduced. However, we will introduce things as they become used in code. For example we introduce templates and in built tags in chapter 3,

 

however we would not give description of each template tag at that place. As we use a new template tag, we would give a description of that.]

## Specs and Schedule

*How many pages do you expect the book to be?*

I am not sure of the form factor to be used in the book, and the words per page. So I am giving a word count. With eight chapters, and 4 appendices, it should be between 40000 – 50000 words.

*What use you will make of illustrations or screenshots?*

At the beginning of each chapter we will give a screenshot, of what we will build, to get the user excited about that chapter.

*What special considerations apply to you plans for the book, including unusual format, use of color, hard-to-get illustrations, or anything else calling for unusual resources?*

*When to you anticipate delivering a complete draft of the manuscript or technical review?*

I have previous engagements till 31May. After that I would be able to work full time on this book, and deliver a complete draft of the book by 1 June.

*[Answering a question raised in mail]How does the market for Django compare with Grok, or Turbogears.*

Grok, is based on Zope3, and hence has many of strengths and weaknesses. It tries to be agile, but in my opinion is harder to learn and

 

use than Django/Turbogears. It is trying to target to a different market. The main competition for Django in web framework space is Turbogears, and Cherrypy. To take the number of users in the associated form as a measure of popularity,

Django: 8561

Turbogears: 3353

Cherrypy: 1101

It is difficult to take Google trends measurement for Turbogears vs Django, as Django also refers to a Jazz musician as well, so searches tend to mix up.

Django also has got a much larger mindshare, and is being more actively developed. (Django has around 7500 commits, Tuurbogears about 2000)

| |
|---|
| Please answer every question in these guidelines and provide a writing sample (samples can include a sample chapter, whitepaper, or magazine article). Thank you for giving O'Reilly an opportunity to consider your proposal.<br><br>Sample attached as a zip. |

## Recommended Reading:

The Forest for the Trees: An Editor's Advice to Writers, Betsy Lerner. Though the author's opinions were formed largely by her work in editing fiction, the book contains plenty of good advice for first-time, would-be authors regardless of their field of interest.

 