Django Bingo

Business Intelligence for the Django community

An overview report from Aperte
A Work In Progress 25/02/2010

Introduction

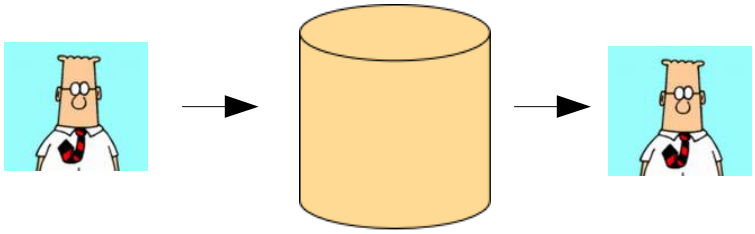
Aperte is a small software consultancy company that specializes in Open Source Software and focuses on web application development.

Aperte has delivered a number of successful projects, all powered by Django. For most of these projects Business Intelligence (BI) features were required, which lead to repeated development of software that could easily be reused for current and future projects.

Django Bingo is a first attempt at integrating BI with Django from Aperte's experience in both areas. It is not a complete BI solution, however Django Bingo does aim to provide BI-like features to Django applications.

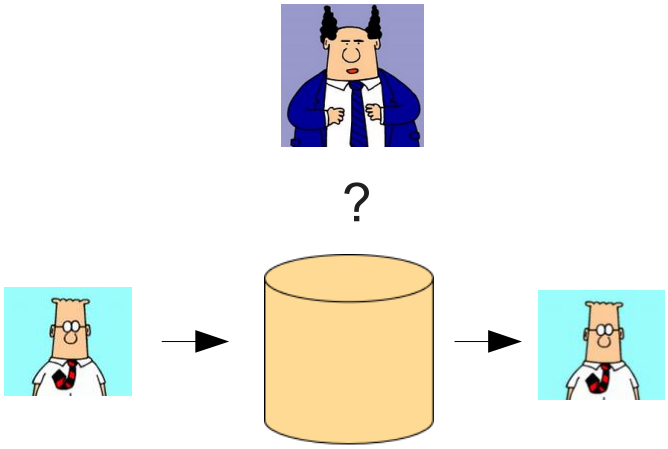
Why would BI be useful for Django projects?

Django is very useful for projects where data goes in and out. Django automates a lot of the drudgery around validating and exposing data. It even offers an easy-to-use administrative interface for users who need to get data in or out of the system. For projects that require input and output (100% of them) Django is a perfect fit.



CONFIDENTIAL - Summary of all projects

The problem Django Bingo hopes to solve is the case where managers (or any user in general) would like to know what is going on inside the system. Django currently doesn't provide a system to easily summarize the data within the system and present that data in a format that the user can comprehend. [1]



The Recurring Problem

1 Apologies to Scott Adams, Dilbert creator, for using his masterpieces

Business Intelligence

The Business Intelligence world also stands to gain from Django Bingo. Currently there are many large systems within the BI area that provide solutions to solve the above problem. The problem with these systems are their lack of ease-of-use, heavy system requirements, high maintenance requirements and methodologies that don't suit the needs of current customers.

A number of proprietary BI solution providers:

- SAP
- IBM Cognos
- Orcale, Siebel
- MS Analysis Services

There are also a few BI projects within the Open Source community:

- JasperSoft / JasperReports
- Eclipse BIRT
- Pentaho

These systems all require expert knowledge in the BI area, dedicated servers and maintenance to run. All for showing a bit of output from a database!

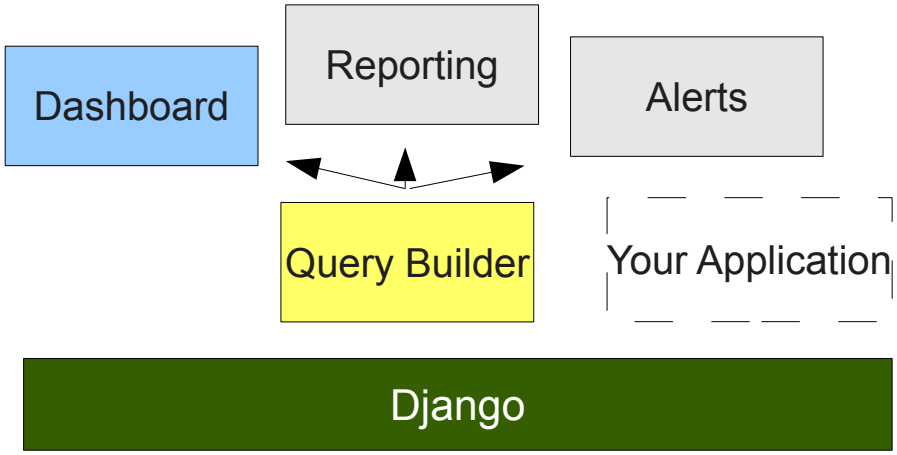
Django Bingo takes a different approach:

- Tightly coupled to Django to allow advanced introspection and automation
- Web-based interface without overloading a user with BI terms
- As light-weight as you would want or as feature-full as you need
- Ability to display data in real-time
- A focus on Django applications and databases (if you want XML/CSV data displayed, make an application out of it)

Our Goal

Django Bingo sits next to your application and queries your database via the same Django layer that you use. This allows Django Bingo to introspect your application and automate the initial work for your Dashboards, Reporting and Alerts.

Each of these 3 applications hook into the Query Builder application. Via this interface a user can set up the parameters of each query, resulting in different results depending on what the query is used for.



Currently Django Bingo only supports other Django applications. In the future it will reuse Django's database introspection to also allow queries on databases of other types of applications, however our focus is strictly on Django-based applications.

Django Bingo also sits next to your applications. This can influence the performance of your system, especially if long-running queries are required (either for your current application or for Django Bingo).

For a production environment we recommend mirroring the database to another server and running Django Bingo on the latter environment. Beyond the performance cost for mirroring the database (which is minimal in most cases) the production environment will not be affected. This will also allow you to run performance-intensive real-time queries and allow you to experiment on the data without any consequences.

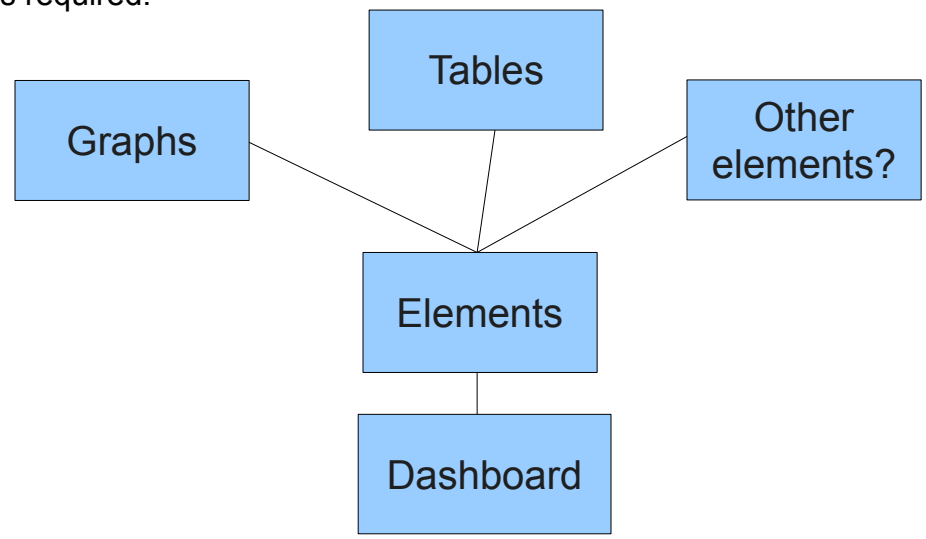
Dashboard

The Dashboard application's purpose is to display queries from the Query Builder in the form appropriate for users. Currently Dashboard provides Graphs and Tables, in various forms and types.

With the term “Dashboards” we're following the description in “Information Dashboard Design” by Stephen Few:

- All information a user would typically require is available on a single screen
- Frivolous design elements are removed (minimize distractions)
- User interface elements are small and non-intrusive (minimize confusion)
- Elements displaying data use neutral tones, except for out-of-the-ordinary values
- Allow the designer to present information in a clear and efficient manner

Naturally the Dashboard application will not provide a perfect dashboard out of the box. It does automate the creation of dashboards, after which a designer can easily adapt the dashboard as required.



Each dashboard (you can have multiple ones, in which case they are tabbed) contains a number of elements that are displayed in a 2 or 3-column layout. Elements can be reordered, optionally refresh their data (configurable per second) and can hidden & shown per dashboard. By toggling between 'design' and 'view' mode the superuser can determine the optimal layout for his application.

(*add examples of dashboard layout & design*)

Query Builder

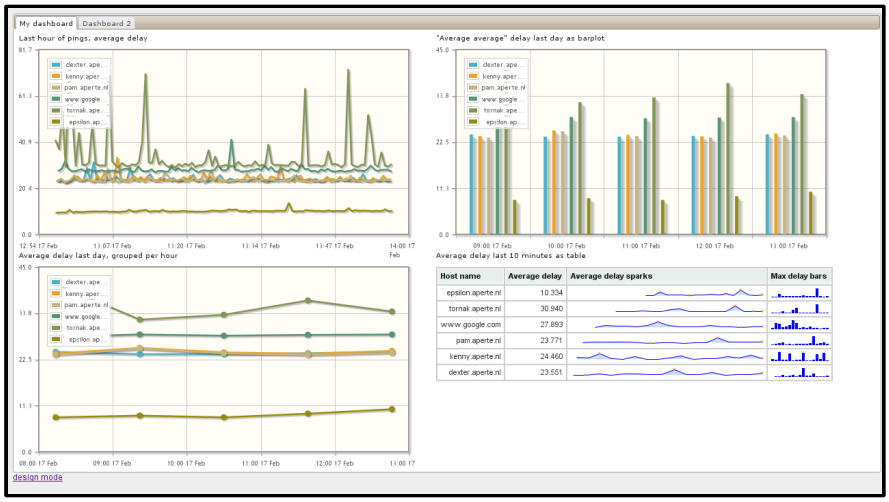
The Query Builder application's only purpose is to provide an easy way to retrieve data from other applications in the database.

Each query is built up by representation using models within the Query Builder (DataSelection being the main model other applications can interface with). Via get_data() a QuerySet is obtained following the limitations specified within the query.

Also in the Query Builder is the ability to add Thresholds. Currently only available for Integer and Decimal fields, thresholds allow the application using the data to visually annotate values that are out-of-the-ordinary.

Django Bingo in its current form

(*to complete*)



An example of Django Bingo's Dashboard