MouseDB Documentation

Release 0.1

Dave Bridges, Ph.D.

December 11, 2010

CONTENTS

Contents:

CHAPTER

ONE

MOUSEDB CONCEPTS

Data storage for MouseDB is separated into packages which contain information about animals, and information collected about animals. There is also a separate module for timed matings of animals. This document will describe the basics of how data is stored in each of these modules.

1.1 Animal Module

Animals are tracked as individual entities, and given associations to breeding cages to follow ancestry, and strains.

1.1.1 Animal

Most parameters about an animal are set within the animal object. Here is where the animals strain, breeding, parentage and many other parameters are included. Animals have foreignkey relationships with both Strain and Breeding, so an animal may only belong to one of each of those. As an example, a mouse cannot come from more than one Breeding set, and cannot belong to more than one strain.

Backcrosses and Generations

For this software, optional tracking of backcrosses and generations is available and is stored as an attribute of an animal. When an inbred cross is made against a pure background, the backcross increases by 1. When a heterozygote cross is made, the generation increases by one. As an example, for every time a mouse in a C57/BL6 background is crossed against a wildtype C57/B6 mouse, the backcross (but not the generation) increases by one. For every time a mutant strain is crosses against itself (either vs a heterozygote or homozygote of that strain), the generation will increase by one. Backcrosses should typically be performed against a separate colony of purebred mouse, rather than against wild-type alleles of the mutant strain.

1.1.2 Breeding Cages

A breeding cage is defined as a set of one or more male and one or more female mice. Because of this, it is not always clear who the precise parentage of an animal is. If the parentage is known, then the Mother and Father fields can be set for a particular animal. In the case of Active, if an End field is specified, then the Active field is set to False. In the case of Cage, if a Cage is provided, and animals are specified under Male or Females for a Breeding object, then the Cage field for those animals is set to that of the breeding cage. The same is true for both Rack and Rack Position.

1.1.3 Strains

A strain is a set of mice with a similar genetics. Importantly strains are separated from Backgrounds. For example, one might have mice with the genotype ob/ob but these mice may be in either a C57-Black6 or a mixed background. This difference is set at the individual animal level. The result of this is that a query for a particular strain may then need to be filtered to a specific background.

1.2 Data Module

Data (or measurements) can be stored for any type of measurement. Conceptually, several pieces of data belong to an experiment (for example several mice are measured at some time) and several experiments belong to a study. Measurements can be stored independent of experiments and experiments can be performed outside of the context of a study. It is however, perfered that measurements are stored within an experiment and experiments are stored within studies as this will greatly facilitate the organization of the data.

1.2.1 Studies

In general studies are a collection of experiments. These can be grouped together on the basis of animals and/or treatment groups. A study must have at least one treatment group, which defines the animals and their conditions.

1.2.2 Experiments

An experiment is a collection of measurements for a given set of animals. In general, an experiment is defined as a number of measurements take in a given day.

1.2.3 Measurements

A measurement is an animal, an assay and a measurement value. It can be associated with an experiment, or can stand alone as an individual value. Measurements can be viewed in the context of a study, an experiment, a treatment group or an animal by going to the appropriate page.

1.3 Timed Matings Module

Timed matings are a specific type of breeding set. Generally, for these experiments a mating cage is set up and pregnancy is defined by a plug event. Based on this information, the age of an embryo can be estimated. When a breeding cage is defined, one option is to set this cage as a timed mating cage (ie Timed_Mating=True). If this is the case, then a plug event can be registered and recorded for this mating set. If the mother gives birth then this cage is implicitly set as a normal breeding cage.

1.4 Groups Module

This app defines generic Group and License information for a particular installation of MouseDB. Because every page on this site identifies both the Group and data restrictions, at a minimum, group information must be provided upon installation (see installation instructions).

CHAPTER

TWO

MOUSEDB INSTALLATION

2.1 Configuration

MouseDB requires both a database and a webserver to be set up. Ideally, the database should be hosted separately from the webserver and MouseDB installation, but this is not necessary, as both can be used from the same server. If you are using a remote server for the database, it is best to set up a user for this database that can only be accessed from the webserver. If you want to set up several installations (ie for different users or different laboratories), you need separate databases and MouseDB installations for each. You will also need to set up the webserver with different addresses for each installation.

2.2 Software Dependencies

1. **MouseDB source code**. Download from one of the following:

1. http://github.com/davebridges/mousedb/downloads for the current release
2. http://github.com/davebridges/mousedb for the source code via Git

Downloading and/or unzipping will create a directory named mousedb. You can update to the newest revision at any time either using git or downloading and re-installing the newer version. Changing or updating software versions will not alter any saved data, but you will have to update the localsettings.py file (described below).

2. **Python**. Requires Version 2.6, is not yet compatible with Python 3.0. Download from http://www.python.org/download/.
3. **Django**. Download from http://www.djangoproject.com/download/. MouseDB currently requires at least Django 1.2.
4. **Database software**. Typically MySQL is used, but PostgreSQL, Oracle or SQLite can also be used. You also need to install the python driver for this database (unless you are using SQLite, which is internal to Python 2.5+). See http://docs.djangoproject.com/en/dev/topics/install/database-installation - Django Database Installation Documentation for more information.

2.3 Database Setup

1. Create a new database. You need to record the user, password, host and database name. If you are using SQLite this step is not required.
2. Go to localsettings_empty.py and edit the settings:

- ENGINE: Choose one of 'django.db.backends.postgresql_psycopg2','django.db.backends.postgresql', 'django.db.backends.mysql', 'django.db.backends.sqlite3', 'django.db.backends.oracle' depending on the database software used.
- NAME: database name
- USER: database user
- PASSWORD: database password
- HOST: database host

3. Save this file as localsettings.py in the main MouseDB directory.

2.4 Web Server Setup

You need to set up a server to serve both the django installation and saved files. For the saved files. I recommend using apache for both. The preferred setup is to use Apache2 with mod_wsgi. See http://code.google.com/p/modwsgi/wiki/InstallationInstructions for instructions on using mod_wsgi. The following is a httpd.conf example where the code is placed in /usr/src/mousedb:

Alias /robots.txt "usrsrcmousedbmediarobots.txt" Alias /favicon.ico "usrsrcmediafavicon.ico"

Alias /mousedb-media "usrsrcmousedbmedia"

<Directory "usrsrcmousedb"> Order allow,deny Allow from all

</Directory>

WSGIScriptAlias /mousedb "usrsrcmousedbapachedjango.wsgi"

Then open up the mousedb/apache/django.wsgi file and replace the location of the mousedb directory on the sys.path.append line.

If you want to restrict access to these files, change the Allow from all directive to specific domains or ip addresses (for example Allow from 192.168.0.0/99 would allow from 192.168.0.0 to 192.168.0.99)

2.5 Final Configuration and User Setup

Go to a command prompt, navigate to inside the mousedb directory and enter the following to get to a python prompt:

```
python manage.py shell
```

Go to servername/mousedb/admin/groups/group/1 and name your research group and select a license if desired

Go to servername/mousedb/admin/auth/users/ and create users, selecting usernames, full names, password (or have the user set the password) and then choose group permissions.

CHAPTER

THREE

ANIMAL DATA ENTRY

3.1 Newborn Mice or Newly Weaned Mice

1. Go to Breeding Cages Tab
2. Click on Add/Wean Pups Button
3. Each row is a new animal. If you accidentaly enter an extra animal, check off the delete box then submit.
4. Leave extra lines blank if you have less than 10 mice to enter
5. If you need to enter more than 10 mice, enter the first ten and submit them. Go back and enter up to 10 more animals (10 more blank spaces will appear)

3.2 Newborn Mice

1. Enter Breeding Cage under Cage
2. Enter Strain
3. Enter Background (normally Mixed or C57BL/6-BA unless from the LY breeding cages in which case it is C57BL/6-LY5.2)
4. Enter Birthdate in format YYYY-MM-DD
5. Enter Generation and Backcross

3.3 Weaning Mice

1. If not previously entered, enter data as if newborn mice
2. Enter gender
3. Enter Wean Date in format YYYY-MM-DD
4. Enter new Cage number for Cage

3.4 Cage Changes (Not Weaning)

1. Find mouse either from animal list or strain list

2. Click the edit mouse button
3. Change the Cage, Rack and Rack Position as Necessary

3.5 Genotyping or Ear Tagging

1. Find mouse either from animal list or strain list, or through breeding cage
2. Click the edit mouse button or the Eartag/Genotype/Cage Change/Death Button
3. Enter the Ear Tag and/or select the Genotype from the Pull Down List

3.6 Marking Mice as Dead

3.6.1 Dead Mice (Single Mouse)

1. Find mouse from animal list or strain list
2. Click the edit mouse button
3. Enter the death date in format YYYY-MM-DD
4. Choose Cause of Death from Pull Down List

3.6.2 Dead Mice (Several Mice)

1. Find mice from breeding cages
2. Click the Eartag/Genotype/Cage Change/Death Button
3. Enter the death date in format YYYY-MM-DD
4. Choose the Cause of Death from Pull Down List

CHAPTER

FOUR

STUDIES AND EXPERIMENTAL SETUP

Set up a new study at /mousedb/admin/data/study/ selecting animals

You must put a description and select animals in one or more treatment groups

If you have more than 2 treatment groups save the first two, then two more empty slots will appear. For animals, click on the magnifying glass then find the animal in that treatment group and click on the MouseID. The number displayed now in that field will not be the MouseID, but don't worry its just a different number to describe the mouse. To add more animals, click on the magnifying glass again and select the next animal. There should be now two numbers, separated by commas in this field. Repeat to fill all your treatment groups. You must enter a diet and environment for each treatment. The other fields are optional, and should only be used if appropriate. Ensure for pharmaceutical, you include a saline treatment group.

CHAPTER

FIVE

MEASURMENT ENTRY

5.1 Studies

If this measurement is part of a study (ie a group of experiments) then click on the plus sign beside the study field and enter in the details about the study and treatment groups. Unfortunately until i can figure out how to filter the treatment group animals in the admin interface, at each of the subsequent steps you will see all the animals in the database (soon hopefully it will only be the ones as part of the study group).

5.2 Experiment Details

- Pick experiment date, feeding state and resarchers
- Pick animals used in this experiment (the search box will filter results)
- Fasting state, time, injections, concentration, experimentID and notes are all optional

5.3 Measurements

- There is room to enter 14 measurements. If you need more rows, enter the first 14 and select "Save and Continue Editing" and 14 more blank spots will appear.
- Each row is a measurement, so if you have glucose and weight for some animal that is two rows entered.
- For animals, click on the magnifying glass then find the animal in that treatment group and click on the MouseID. The number displayed now in that field will not be the MouseID, but don't worry its just a different number to describe the mouse.
- For values, the standard units (defined by each assay) are mg for weights, mg/dL for glucose and pg/mL for insulin). You must enter integers here (no decimal places). If you have several measurements (ie several glucose readings during a GTT, enter them all in one measurement row, separated by commas and *NO spaces*).

CHAPTER

SIX

AUTOMATED DOCUMENTATION

6.1 Data Package

The data module describes the conditions and collection of data regarding experimental animals.

Data (or measurements) can be stored for any type of measurement. Conceptually, several pieces of data belong to an experiment (for example several mice are measured at some time) and several experiments belong to a study. Measurements can be stored independent of experiments and experiments can be performed outside of the context of a study. It is however, perfered that measurements are stored within an experiment and experiments are stored within studies as this will greatly facilitate the organization of the data.

6.1.1 Studies

In general studies are a collection of experiments. These can be grouped together on the basis of animals and/or treatment groups. A study must have at least one treatment group, which defines the animals and their conditions.

6.1.2 Experiments

An experiment is a collection of measurements for a given set of animals. In general, an experiment is defined as a number of measurements take in a given day.

6.1.3 Measurements

A measurement is an animal, an assay and a measurement value. It can be associated with an experiment, or can stand alone as an individual value. Measurements can be viewed in the context of a study, an experiment, a treatment group or an animal by going to the appropriate page.

6.1.4 Models

class `data.models.`**`Assay`**(**args*, ***kwargs*)

Assay(id, assay, assay_slug, notes, measurement_units)

exception **`DoesNotExist`**

exception `Assay.`**`MultipleObjectsReturned`**

`Assay.`**`measurement_set`**

class `data.models.`**`Diet`**`(`*`*args`*`, `*`**kwargs`*`)`
Diet(id, vendor_id, description, product_id, fat_content, protein_content, carb_content, irradiated, notes)

exception **`DoesNotExist`**

exception `Diet.`**`MultipleObjectsReturned`**

`Diet.`**`treatment_set`**

`Diet.`**`vendor`**

class `data.models.`**`Environment`**`(`*`*args`*`, `*`**kwargs`*`)`
Environment(id, building, room, temperature, humidity, notes)

exception **`DoesNotExist`**

exception `Environment.`**`MultipleObjectsReturned`**

`Environment.`**`contact`**

`Environment.`**`treatment_set`**

class `data.models.`**`Experiment`**`(`*`*args`*`, `*`**kwargs`*`)`
Experiment(id, date, notes, experimentID, feeding_state, fasting_time, injection, concentration, study_id)

exception **`DoesNotExist`**

exception `Experiment.`**`MultipleObjectsReturned`**

`Experiment.`**`animals`**

`Experiment.`**`get_absolute_url`**`(`*`*moreargs`*`, `*`**morekwargs`*`)`

`Experiment.`**`get_feeding_state_display`**`(`*`*moreargs`*`, `*`**morekwargs`*`)`

`Experiment.`**`get_injection_display`**`(`*`*moreargs`*`, `*`**morekwargs`*`)`

`Experiment.`**`get_next_by_date`**`(`*`*moreargs`*`, `*`**morekwargs`*`)`

`Experiment.`**`get_previous_by_date`**`(`*`*moreargs`*`, `*`**morekwargs`*`)`

`Experiment.`**`measurement_set`**

`Experiment.`**`researchers`**

`Experiment.`**`study`**

class `data.models.`**`Implantation`**`(`*`*args`*`, `*`**kwargs`*`)`
Implantation(id, implant, vendor_id, product_id, notes)

exception **`DoesNotExist`**

exception `Implantation.`**`MultipleObjectsReturned`**

`Implantation.`**`surgeon`**

`Implantation.`**`treatment_set`**

`Implantation.`**`vendor`**

class `data.models.`**`Measurement`**`(`*`*args`*`, `*`**kwargs`*`)`
Measurement(id, animal_id, experiment_id, assay_id, values)

exception **`DoesNotExist`**

exception `Measurement.`**`MultipleObjectsReturned`**

`Measurement.`**`animal`**

`Measurement.`**`assay`**

Measurement.**experiment**

class data.models.**Pharmaceutical**(**args*, ***kwargs*)
Pharmaceutical(id, drug, dose, recurrance, mode, vendor_id, notes)

exception DoesNotExist

exception Pharmaceutical.**MultipleObjectsReturned**

Pharmaceutical.**get_mode_display**(**moreargs*, ***morekwargs*)

Pharmaceutical.**treatment_set**

Pharmaceutical.**vendor**

class data.models.**Researcher**(**args*, ***kwargs*)
Researcher(id, first_name, last_name, name_slug, email, active)

exception DoesNotExist

exception Researcher.**MultipleObjectsReturned**

Researcher.**environment_set**

Researcher.**experiment_set**

Researcher.**implantation_set**

Researcher.**transplantation_set**

Researcher.**treatment_set**

class data.models.**Study**(**args*, ***kwargs*)
Study(id, description, start_date, stop_date, notes)

exception DoesNotExist

exception Study.**MultipleObjectsReturned**

Study.**experiment_set**

Study.**get_absolute_url**(**moreargs*, ***morekwargs*)

Study.**strain**

Study.**treatment_set**

class data.models.**Transplantation**(**args*, ***kwargs*)
Transplantation(id, tissue, transplant_date, notes)

exception DoesNotExist

exception Transplantation.**MultipleObjectsReturned**

Transplantation.**donor**

Transplantation.**get_next_by_transplant_date**(**moreargs*, ***morekwargs*)

Transplantation.**get_previous_by_transplant_date**(**moreargs*, ***morekwargs*)

Transplantation.**surgeon**

Transplantation.**treatment_set**

class data.models.**Treatment**(**args*, ***kwargs*)
Treatment(id, treatment, study_id, diet_id, environment_id, transplantation_id, notes)

exception DoesNotExist

exception Treatment.**MultipleObjectsReturned**

Treatment.**animals**

Treatment.**diet**

Treatment.**environment**

Treatment.**get_absolute_url**(**moreargs*, ***morekwargs*)

Treatment.**implantation**

Treatment.**pharmaceutical**

Treatment.**researchers**

Treatment.**study**

Treatment.**transplantation**

class data.models.**Vendor**(**args*, ***kwargs*)

Vendor(id, vendor, website, email, ordering, notes)

exception DoesNotExist

exception Vendor.**MultipleObjectsReturned**

Vendor.**diet_set**

Vendor.**get_ordering_display**(**moreargs*, ***morekwargs*)

Vendor.**implantation_set**

Vendor.**pharmaceutical_set**

6.1.5 Forms

class data.forms.**ExperimentForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.util.ErrorList'>*, *label_suffix=':'*, *empty_permitted=False*, *instance=None*)

This is the configuration for the experiment form.

This form is used to set up and modify an experiment. It uses a datepicker widget for the date, and autocomplete forms for the animals.

class Media

class ExperimentForm.**Meta**

model

alias of Experiment

ExperimentForm.**media**

class data.forms.**MeasurementForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.util.ErrorList'>*, *label_suffix=':'*, *empty_permitted=False*, *instance=None*)

Form definition for adding and editing measurements.

This form excludes experiment, which must be passed as a filtering parameter from the view. This form is used for formsets to add or modify measurements from within an experiment.

class Meta

model
alias of Measurement

MeasurementForm.**media**

class data.forms.**StudyExperimentForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.util.ErrorList'>*, *label_suffix=':'*, *empty_permitted=False*, *instance=None*)

class Meta

model
alias of Experiment

StudyExperimentForm.**media**

class data.forms.**StudyForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.util.ErrorList'>*, *label_suffix=':'*, *empty_permitted=False*, *instance=None*)
This is the configuration for the study form.

This form is used to create and modify studies. It uses an autocomplete widget for the animals.

class Media

class StudyForm.**Meta**

model
alias of Study

StudyForm.**media**

class data.forms.**TreatmentForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.util.ErrorList'>*, *label_suffix=':'*, *empty_permitted=False*, *instance=None*)
Form class for study treatment groups.

In the case of studies, animals are defined in the treatment group rather than in the study group. A treatment consists of a study, a set of animals and the conditions which define that treatment. This includes related fields for environment, diet, implants and transplants.

class Meta

model
alias of Treatment

TreatmentForm.**media**

6.1.6 Views and URLs

data.views.**add_measurement**(*request*, **args*, ***kwargs*)

data.views.**experiment_detail**(*request*, **args*, ***kwargs*)

data.views.**experiment_detail_all**(*request*, **args*, ***kwargs*)

data.views.**experiment_list**(*request*, **args*, ***kwargs*)

data.views.**study_experiment**(*request*, **args*, ***kwargs*)

6.1.7 Administrative Site Configuration

class `data.admin.`**`AssayAdmin`**(*model*, *admin_site*)

> **`media`**

class `data.admin.`**`DietAdmin`**(*model*, *admin_site*)

> **`media`**

class `data.admin.`**`EnvironmentAdmin`**(*model*, *admin_site*)

> **`media`**

class `data.admin.`**`ExperimentAdmin`**(*model*, *admin_site*)

> **`media`**

class `data.admin.`**`ImplantationAdmin`**(*model*, *admin_site*)

> **`media`**

class `data.admin.`**`MeasurementAdmin`**(*model*, *admin_site*)

> **`media`**

class `data.admin.`**`MeasurementInline`**(*parent_model*, *admin_site*)

> **`media`**
>
> **`model`**
>
> alias of `Measurement`

class `data.admin.`**`PharmaceuticalAdmin`**(*model*, *admin_site*)

> **`media`**

class `data.admin.`**`ResearcherAdmin`**(*model*, *admin_site*)

> **`media`**

class `data.admin.`**`StudyAdmin`**(*model*, *admin_site*)

> **`media`**

class `data.admin.`**`TransplantationAdmin`**(*model*, *admin_site*)

> **`media`**

class `data.admin.`**`TreatmentAdmin`**(*model*, *admin_site*)

> **`media`**

class `data.admin.`**`TreatmentInline`**(*parent_model*, *admin_site*)

> **`media`**
>
> **`model`**
> alias of `Treatment`

class `data.admin.`**`VendorAdmin`**(*model*, *admin_site*)

> **`media`**

6.2 Animals Package

The animal app contains and controls the display of data about animals.

Animals are tracked as individual entities, and given associations to breeding cages to follow ancestry, and strains.

6.2.1 Animal

Most parameters about an animal are set within the animal object. Here is where the animals strain, breeding, parentage and many other parameters are included. Animals have foreignkey relationships with both Strain and Breeding, so an animal may only belong to one of each of those. As an example, a mouse cannot come from more than one Breeding set, and cannot belong to more than one strain.

Backcrosses and Generations

For this software, optional tracking of backcrosses and generations is available and is stored as an attribute of an animal. When an inbred cross is made against a pure background, the backcross increases by 1. When a heterozygote cross is made, the generation increases by one. As an example, for every time a mouse in a C57/BL6 background is crossed against a wildtype C57/B6 mouse, the backcross (but not the generation) increases by one. For every time a mutant strain is crosses against itself (either vs a heterozygote or homozygote of that strain), the generation will increase by one. Backcrosses should typically be performed against a separate colony of purebred mouse, rather than against wild-type alleles of the mutant strain.

6.2.2 Breeding Cages

A breeding cage is defined as a set of one or more male and one or more female mice. Because of this, it is not always clear who the precise parentage of an animal is. If the parentage is known, then the Mother and Father fields can be set for a particular animal.

6.2.3 Strains

A strain is a set of mice with a similar genetics. Importantly strains are separated from Backgrounds. For example, one might have mice with the genotype ob/ob but these mice may be in either a C57-Black6 or a mixed background. This difference is set at the individual animal level. The result of this is that a query for a particular strain may then need to be filtered to a specific background.

6.2.4 Models

This module describes the Strain, Animal, Breeding and Cage data models.

This module stores all data regarding a particular laboratory animal. Information about experimental data and timed matings are stored in the data and timed_matings packages. This module describes the database structure for each data model.

class `animal.models.`**`Cage`**(**args*, ***kwargs*)

This data model stores information about a particular cage.

This model, which is not yet implemented will be used by both breeding and non-breeding cages and will facilitate easier tracking and storage of cages. To implement this, it will be necessary to automatically generate a new cage (if a novel barcode is entered), or to use a current cage if the barcode is already present in the database

exception **`DoesNotExist`**

exception `Cage.`**`MultipleObjectsReturned`**

6.2.5 Forms

Forms for use in manipulating objects in the animal app.

class `animal.forms.`**`AnimalForm`**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.util.ErrorList'>*, *label_suffix=':'*, *empty_permitted=False*, *instance=None*)

This modelform provides fields for modifying animal data.

It includes all fields except CageID (will be deprecated) and Alive (which is automattically set upon saving the animal). This form also automatically loads javascript and css for the datepicker jquery-ui widget. It also includes auto

class **`Media`**

class `AnimalForm.`**`Meta`**

`model`

alias of `Animal`

`AnimalForm.`**`media`**

class `animal.forms.`**`BreedingForm`**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.util.ErrorList'>*, *label_suffix=':'*, *empty_permitted=False*, *instance=None*)

This form provides most fields for creating and entring breeding cage data.

This form is used from the url /mousedb/breeding/new and is a generic create view. The only excluded field is CageID, as this feature will be deprecated. This view includes a datepicker widget for Stat and End dates and autocomplete fields for the Females and Male fields

class **`Media`**

class `BreedingForm.`**`Meta`**

`model`

alias of `Breeding`

`BreedingForm.`**`media`**

6.2.6 Views and URLs

6.2.7 Administrative Site Configuration

6.3 Timed Mating Package

This package defines the timed_mating app.

Timed matings are a specific type of breeding set. Generally, for these experiments a mating cage is set up and pregnancy is defined by a plug event. Based on this information, the age of an embryo can be estimated. When a breeding cage is defined, one option is to set this cage as a timed mating cage (ie Timed_Mating=True). If this is the case, then a plug event can be registered and recorded for this mating set. If the mother gives birth then this cage is implicitly set as a normal breeding cage.

6.3.1 Models

This defines the data model for the timed_mating app.

Currently the only data model is for PlugEvents.

class `timed_mating.models.`**`PlugEvents`**`(`**args*, ***kwargs*`)`

This defines the model for PlugEvents.

A PlugEvent requires a date. All other fields are optional. Upon observation of a plug event, the PlugDate, Breeding Cage, Femalem, Male, Researcher and Notes can be set. Upon sacrifice of the mother, then genotyped alive and dead embryos can be entered, along with the SacrificeDate, Researcher and Notes.

`Breeding`

exception **`DoesNotExist`**

exception `PlugEvents.`**`MultipleObjectsReturned`**

`PlugEvents.`**`PlugFemale`**

`PlugEvents.`**`PlugMale`**

`PlugEvents.`**`Researcher`**

`PlugEvents.`**`get_absolute_url`**`(`**moreargs*, ***morekwargs*`)`

`PlugEvents.`**`get_next_by_PlugDate`**`(`**moreargs*, ***morekwargs*`)`

`PlugEvents.`**`get_previous_by_PlugDate`**`(`**moreargs*, ***morekwargs*`)`

`PlugEvents.`**`save`**`()`

Over-rides the default save function for PlugEvents.

If a sacrifice date is set for an object in this model, then Active is set to False.

6.3.2 Forms

This package describes forms used by the Timed Mating app.

class `timed_mating.forms.`**`BreedingPlugForm`**`(`*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.util.ErrorList'>*, *label_suffix=':'*, *empty_permitted=False*, *instance=None*`)`

This form is used to enter Plug Events from a specific breeding cage.

class **Meta**

model
alias of `PlugEvents`

`BreedingPlugForm.`**media**

6.3.3 Views and URLs

This package defines custom views for the timed_mating application.

Currently all views are generic CRUD views except for the view in which a plug event is defined from a breeding cage.

`timed_mating.views.`**breeding_plugevent**(*request*, **args*, ***kwargs*)
This view defines a form for adding new plug events from a breeding cage.

This form requires a breeding_id from a breeding set and restricts the PlugFemale and PlugMale to animals that are defined in that breeding cage.

`timed_mating.urls.`**change_plugevents**(*request*, **args*, ***kwargs*)

`timed_mating.urls.`**create_plugevents**(*request*, **args*, ***kwargs*)

`timed_mating.urls.`**delete_plugevents**(*request*, **args*, ***kwargs*)

`timed_mating.urls.`**limited_object_detail**(*request*, **args*, ***kwargs*)

`timed_mating.urls.`**limited_object_list**(*request*, **args*, ***kwargs*)

6.3.4 Administrative Site Configuration

Settings to control the admin interface for the timed_mating app.

This file defines a PlugEventsAdmin object to enter parameters about individual plug events/

class `timed_mating.admin.`**PlugEventsAdmin**(*model*, *admin_site*)
This class defines the admin interface for the PlugEvents model.

media

6.4 Groups Package

This package defines the Group application. This app defines generic Group and License information for a particular installation of MouseDB. Because every page on this site identifies both the Group and data restrictions, at a minimum, group information must be provided upon installation (see installation instructions).

6.4.1 Models

class `groups.models.`**Group**(**args*, ***kwargs*)
This defines the data structure for the Group model.

The only required field is group. All other fields (group_slug, group_url, license, contact_title, contact_first, contact_last and contact_email) are optional.

exception **DoesNotExist**

exception `Group.`**`MultipleObjectsReturned`**

`Group.`**`get_contact_title_display`**`(`**moreargs*, ***morekwargs*`)`

`Group.`**`license`**

6.4.2 Views and URLs

6.4.3 Administrative Site Configuration

class `groups.admin.`**`GroupAdmin`**`(`*model*, *admin_site*`)`

Defines the admin interface for Groups.

Currently set as default.

`media`

class `groups.admin.`**`LicenseAdmin`**`(`*model*, *admin_site*`)`

Defines the admin interface for Licences.

Currently set as default.

`media`

CHAPTER

SEVEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

INDEX