NAME

Catalyst::Manual::Tutorial::10_Appendices - Catalyst Tutorial - Chapter 10: Appendices

OVERVIEW

This is **Chapter 10 of 10** for the Catalyst tutorial.

DESCRIPTION

This chapter of the tutorial provides supporting information relevant to the Catalyst tutorial.

APPENDIX 1: CUT AND PASTE FOR POD-BASED EXAMPLES

You may notice that Pod indents example code with four spaces. This section provides some quick advice to "un-indent" this text in common editors.

"Un-indenting" with Vi/Vim

When cutting and pasting multi-line text from Pod-based documents, the following vi/vim regexs can be helpful to "un-indent" the inserted text (do NOT type the quotes, they are only included to show spaces in the regex patterns). *Note that all 3 of the regexs end in 4 spaces*:

- ":0,$s/^ "

  Removes four leading spaces from the entire file (from the first line, `0`, to the last line, `$`).

- "%s/^ "

  A shortcut for the previous item (`%` specifies the entire file; so this removes four leading spaces from every line).

- ":.,$s/^ "

  Removes the first four spaces from the line the cursor is on at the time the regex command is executed (".") to the last line of the file.

- ":.,44s/^ "

  Removes four leading space from the current line through line 44 (obviously adjust the `44` to the appropriate value in your example).

"Un-indenting" with Emacs

Although the author has not used Emacs for many years (apologies to the Emacs fans out there), here is a quick hint to get you started. To replace the leading spaces of every line in a file, use:

```
M-x replace-regexp<RET>
Replace regexp: ^    <RET>
with: <RET>
```

All of that will occur on the single line at the bottom of your screen. Note that "<RET>" represents the return key/enter. Also, there are four spaces after the "^" on the "Replace regexp:" line and no spaces entered on the last line.

You can limit the replacement operation by selecting text first (depending on your version of Emacs, you can either use the mouse or experiment with commands such as `C-SPC` to set the mark at the cursor location and `C-<` and `C->` to set the mark at the beginning and end of the file respectively.

Also, Stefan Kangas sent in the following tip about an alternate approach using the command `indent-region` to redo the indentation for the currently selected region (adhering to indent rules in the current major mode). You can run the command by typing M-x indent-region or pressing the default

keybinding C-M-\ in cperl-mode. Additional details can be found here:

http://www.gnu.org/software/emacs/manual/html_node/emacs/Indentation-Commands.html

APPENDIX 2: USING POSTGRESQL AND MYSQL

The main database used in this tutorial is the very simple yet powerful SQLite. This section provides information that can be used to "convert" the tutorial to use PostgreSQL and MySQL. However, note that part of the beauty of the MVC architecture is that very little database-specific code is spread throughout the system (at least when MVC is "done right"). Consequently, converting from one database to another is relatively painless with most Catalyst applications. In general, you just need to adapt the schema definition `.sql` file you use to initialize your database and adjust a few configuration parameters.

Also note that the purpose of the data definition statements for this section are not designed to take maximum advantage of the various features in each database for issues such as referential integrity and field types/constraints.

PostgreSQL

Use the following steps to adapt the tutorial to PostgreSQL. Thanks to Caelum (Rafael Kitover) for assistance with the most recent updates, and Louis Moore, Marcello Romani and Tom Lanyon for help with earlier versions.

- Chapter 3: More Catalyst Basics
  - Install the PostgreSQL server and client and DBD::Pg:

    If you are following along in Debian 6, you can quickly install these items via this command:

    ```
    sudo aptitude install postgresql libdbd-pg-perl libdatetime-format-pg-perl
    ```

    To configure the permissions, you can open `/etc/postgresql/8.3/main/pg_hba.conf` and change this line (near the bottom):

    ```
    # "local" is for Unix domain socket connections only
    local   all         all                               ident sameuser
    ```

    to:

    ```
    # "local" is for Unix domain socket connections only
    local   all         all                               trust
    ```

    And then restart PostgreSQL:

    ```
    sudo /etc/init.d/postgresql-8.3 restart
    ```

  - Create the database and a user for the database (note that we are using "<catalyst>" to represent the hidden password of "catalyst"):

    ```
    $ sudo -u postgres createuser -P catappuser
    Enter password for new role: <catalyst>
    Enter it again: <catalyst>
    Shall the new role be a superuser? (y/n) n
    Shall the new role be allowed to create databases? (y/n) n
    Shall the new role be allowed to create more new roles? (y/n) n
    CREATE ROLE
    $ sudo -u postgres createdb -O catappuser catappdb
    CREATE DATABASE
    ```

  - Create the `.sql` file and load the data:
    - Open the `myapp01_psql.sql` in your editor and enter:

      ```
      --
      -- Drops just in case you are reloading
      ---
      DROP TABLE IF EXISTS books CASCADE;
      DROP TABLE IF EXISTS authors CASCADE;
      DROP TABLE IF EXISTS book_authors CASCADE;
      DROP TABLE IF EXISTS users CASCADE;
      DROP TABLE IF EXISTS roles CASCADE;
      DROP TABLE IF EXISTS user_roles CASCADE;

      --
      -- Create a very simple database to hold book and author information
      --
      CREATE TABLE books (
          id          SERIAL PRIMARY KEY,
          title       TEXT ,
          rating      INTEGER,
          -- Manually add these later
          -- created     TIMESTAMP NOT NULL DEFAULT now(),
          -- updated     TIMESTAMP
      );

      CREATE TABLE authors (
          id          SERIAL PRIMARY KEY,
          first_name  TEXT,
          last_name   TEXT
      ```

```
);

-- 'book_authors' is a many-to-many join table between books & authors
CREATE TABLE book_authors (
    book_id     INTEGER REFERENCES books(id) ON DELETE CASCADE ON UPDATE CASCADE,
    author_id   INTEGER REFERENCES authors(id) ON DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (book_id, author_id)
);

---
--- Load some sample data
---
INSERT INTO books (title, rating) VALUES ('CCSP SNRS Exam Certification Guide', 5);
INSERT INTO books (title, rating) VALUES ('TCP/IP Illustrated, Volume 1', 5);
INSERT INTO books (title, rating) VALUES ('Internetworking with TCP/IP Vol.1', 4);
INSERT INTO books (title, rating) VALUES ('Perl Cookbook', 5);
INSERT INTO books (title, rating) VALUES ('Designing with Web Standards', 5);
INSERT INTO authors (first_name, last_name) VALUES ('Greg', 'Bastien');
INSERT INTO authors (first_name, last_name) VALUES ('Sara', 'Nasseh');
INSERT INTO authors (first_name, last_name) VALUES ('Christian', 'Degu');
INSERT INTO authors (first_name, last_name) VALUES ('Richard', 'Stevens');
INSERT INTO authors (first_name, last_name) VALUES ('Douglas', 'Comer');
INSERT INTO authors (first_name, last_name) VALUES ('Tom', 'Christiansen');
INSERT INTO authors (first_name, last_name) VALUES ('Nathan', 'Torkington');
INSERT INTO authors (first_name, last_name) VALUES ('Jeffrey', 'Zeldman');
INSERT INTO book_authors VALUES (1, 1);
INSERT INTO book_authors VALUES (1, 2);
INSERT INTO book_authors VALUES (1, 3);
INSERT INTO book_authors VALUES (2, 4);
INSERT INTO book_authors VALUES (3, 5);
INSERT INTO book_authors VALUES (4, 6);
INSERT INTO book_authors VALUES (4, 7);
INSERT INTO book_authors VALUES (5, 8);
```

- Load the data:

```
$ psql -U catappuser -W catappdb -f myapp01_psql.sql
Password for user catappuser:
psql:myapp01_psql.sql:8: NOTICE:  CREATE TABLE will create implicit sequence "books_id_seq" for serial column "books.id"
psql:myapp01_psql.sql:8: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "books_pkey" for table "books"
CREATE TABLE
psql:myapp01_psql.sql:15: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "book_authors_pkey" for table "book_author
CREATE TABLE
psql:myapp01_psql.sql:21: NOTICE:  CREATE TABLE will create implicit sequence "authors_id_seq" for serial column "authors.id"
psql:myapp01_psql.sql:21: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "authors_pkey" for table "authors"
CREATE TABLE
INSERT 0 1
INSERT 0 1
INSERT 0 1
...
```

- Make sure the data loaded correctly:

```
$ psql -U catappuser -W catappdb
Password for user catappuser: <catalyst>
Welcome to psql 8.3.7, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

catappdb=> \dt
             List of relations
 Schema |     Name     | Type  |   Owner
--------+--------------+-------+------------
 public | authors      | table | catappuser
 public | book_authors | table | catappuser
 public | books        | table | catappuser
(3 rows)

catappdb=> select * from books;
 id |               title                | rating
----+------------------------------------+--------
  1 | CCSP SNRS Exam Certification Guide |      5
  2 | TCP/IP Illustrated, Volume 1       |      5
  3 | Internetworking with TCP/IP Vol.1  |      4
  4 | Perl Cookbook                      |      5
  5 | Designing with Web Standards       |      5
(5 rows)

catappdb=>
```

- After the steps where you:

```
edit lib/MyApp.pm

create lib/MyAppDB.pm

create lib/MyAppDB/Book.pm
```

```
    create lib/MyAppDB/Author.pm

    create lib/MyAppDB/BookAuthor.pm
```

- Generate the model using the Catalyst "_create.pl" script:

```
    $ rm lib/MyApp/Model/DB.pm   # Delete just in case already there
    $ script/myapp_create.pl model DB DBIC::Schema MyApp::Schema \
        create=static components=TimeStamp,PassphraseColumn \
        'dbi:Pg:dbname=catappdb' 'catappuser' 'catalyst' '{ AutoCommit => 1 }'
```

- Chapter 4: Basic CRUD

  Add Datetime Columns to Our Existing Books Table

```
    $ psql -U catappuser -W catappdb
    ...
    catappdb=> ALTER TABLE books ADD created TIMESTAMP NOT NULL DEFAULT now();
    ALTER TABLE
    catappdb=> ALTER TABLE books ADD updated TIMESTAMP;
    ALTER TABLE
    catappdb=> \q
```

  Re-generate the model using the Catalyst "_create.pl" script:

```
    $ script/myapp_create.pl model DB DBIC::Schema MyApp::Schema \
        create=static components=TimeStamp,PassphraseColumn \
        'dbi:Pg:dbname=catappdb' 'catappuser' 'catalyst' '{ AutoCommit => 1 }'
```

- Chapter 5: Authentication
  - Create the `.sql` file for the user/roles data:

    Open `myapp02_psql.sql` in your editor and enter:

```
    --
    -- Add users and roles tables, along with a many-to-many join table
    --

    CREATE TABLE users (
        id            SERIAL PRIMARY KEY,
        username      TEXT,
        password      TEXT,
        email_address TEXT,
        first_name    TEXT,
        last_name     TEXT,
        active        INTEGER
    );

    CREATE TABLE roles (
        id   SERIAL PRIMARY KEY,
        role TEXT
    );

    CREATE TABLE user_roles (
        user_id INTEGER REFERENCES users(id) ON DELETE CASCADE ON UPDATE CASCADE,
        role_id INTEGER REFERENCES roles(id) ON DELETE CASCADE ON UPDATE CASCADE,
        PRIMARY KEY (user_id, role_id)
    );

    --
    -- Load up some initial test data
    --
    INSERT INTO users (username, password, email_address, first_name, last_name, active)
        VALUES ('test01', 'mypass', 't01@na.com', 'Joe',  'Blow', 1);
    INSERT INTO users (username, password, email_address, first_name, last_name, active)
        VALUES ('test02', 'mypass', 't02@na.com', 'Jane', 'Doe',  1);
    INSERT INTO users (username, password, email_address, first_name, last_name, active)
        VALUES ('test03', 'mypass', 't03@na.com', 'No',   'Go',   0);
    INSERT INTO roles (role) VALUES ('user');
    INSERT INTO roles (role) VALUES ('admin');
    INSERT INTO user_roles VALUES (1, 1);
    INSERT INTO user_roles VALUES (1, 2);
    INSERT INTO user_roles VALUES (2, 1);
    INSERT INTO user_roles VALUES (3, 1);
```

  - Load the data:

```
    $ psql -U catappuser -W catappdb -f myapp02_psql.sql
    Password for user catappuser: <catalyst>
    psql:myapp02_psql.sql:13: NOTICE:  CREATE TABLE will create implicit sequence "users_id_seq" for serial column "users.id"
    psql:myapp02_psql.sql:13: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "users_pkey" for table "users"
    CREATE TABLE
    psql:myapp02_psql.sql:18: NOTICE:  CREATE TABLE will create implicit sequence "roles_id_seq" for serial column "roles.id"
    psql:myapp02_psql.sql:18: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "roles_pkey" for table "roles"
    CREATE TABLE
    psql:myapp02_psql.sql:24: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "user_roles_pkey" for table "user_roles"
    CREATE TABLE
```

```
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
```

Confirm with:

```
$ psql -U catappuser -W catappdb -c "select * from users"
Password for user catappuser: <catalyst>
 id | username | password | email_address | first_name | last_name | active
----+----------+----------+---------------+------------+-----------+--------
  1 | test01   | mypass   | t01@na.com    | Joe        | Blow      |      1
  2 | test02   | mypass   | t02@na.com    | Jane       | Doe       |      1
  3 | test03   | mypass   | t03@na.com    | No         | Go        |      0
(3 rows)
```

- Modify `set_hashed_passwords.pl` to match the following (the only difference is the `connect` line):

```
#!/usr/bin/perl

use strict;
use warnings;

use MyApp::Schema;

my $schema = MyApp::Schema->connect('dbi:Pg:dbname=catappdb', 'catappuser', 'catalyst');

my @users = $schema->resultset('Users')->all;

foreach my $user (@users) {
    $user->password('mypass');
    $user->update;
}
```

Run the `set_hashed_passwords.pl` as per the "normal" flow of the tutorial:

```
$ perl -Ilib set_hashed_passwords.pl
```

You can verify that it worked with this command:

```
$ psql -U catappuser -W catappdb -c "select * from users"
```

MySQL

Use the following steps to adapt the tutorial to MySQL. Thanks to Jim Howard for the help and Zsolt Zemancsik for the up to date fixes.

- Chapter 3: Catalyst Basics
  - Install the required software:
    - The MySQL database server and client utility.
    - The Perl `DBD::MySQL` module

For CentOS users (see Catalyst::Manual::Installation::CentOS4), you can use the following commands to install the software and start the MySQL daemon:

```
yum -y install mysql mysql-server
service mysqld start
```

For Debian users you can use the following commands to install the software and start the MySQL daemon:

```
apt-get install mysql-client mysql-server
/etc/init.d/mysql start
```

NOTE: The tutorial is based on Foreign Keys in database which is supported by InnoDB. Only MySQL 5.0 and above supports InnoDB storage Engine so you need to have InnoDB support in you MySQL. You can simply figure out that your install supports it or not:

```
# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql> SHOW VARIABLES LIKE 'have_innodb';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
```

```
| have_innodb   | YES   |
+---------------+-------+
1 row in set (0.01 sec)

mysql> exit
Bye
```

If the Value is "YES" you can use your setup (Debian based mysql supports it by default). Else, you need to configure your my.cnf or start your MySQL daemon without --skip-innodb option.

- Create the database and set the permissions:

```
# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE `myapp`;
Query OK, 1 row affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON myapp.* TO 'tutorial'@'localhost' IDENTIFIED BY 'yourpassword';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
```

- Create the `.sql` file and load the data:
  - Open the `myapp01_mysql.sql` in your editor and enter:

```
--
-- Create a very simple database to hold book and author information
--
CREATE TABLE IF NOT EXISTS `books` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` text CHARACTER SET utf8,
  `rating` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
-- 'book_authors' is a many-to-many join table between books & authors
CREATE TABLE IF NOT EXISTS `book_authors` (
  `book_id` int(11) NOT NULL DEFAULT '0',
  `author_id` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`book_id`,`author_id`),
  KEY `author_id` (`author_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE IF NOT EXISTS `authors` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `first_name` text CHARACTER SET utf8,
  `last_name` text CHARACTER SET utf8,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
---
--- Load some sample data
---
INSERT INTO `books` (`id`, `title`, `rating`) VALUES
(1, 'CCSP SNRS Exam Certification Guide', 5),
(2, 'TCP/IP Illustrated, Volume 1', 5),
(3, 'Internetworking with TCP/IP Vol.1', 4),
(4, 'Perl Cookbook', 5),
(5, 'Designing with Web Standards', 5);

INSERT INTO `book_authors` (`book_id`, `author_id`) VALUES
(1, 1),
(1, 2),
(1, 3),
(2, 4),
(3, 5),
(4, 6),
(4, 7),
(5, 8);

INSERT INTO `authors` (`id`, `first_name`, `last_name`) VALUES
(1, 'Greg', 'Bastien'),
(2, 'Sara', 'Nasseh'),
(3, 'Christian', 'Degu'),
(4, 'Richard', 'Stevens'),
(5, 'Douglas', 'Comer'),
(6, 'Tom', 'Christiansen'),
(7, 'Nathan', 'Torkington'),
(8, 'Jeffrey', 'Zeldman');

ALTER TABLE `book_authors`
ADD CONSTRAINT `book_author_ibfk_2` FOREIGN KEY (`author_id`) REFERENCES `authors` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT `book_author_ibfk_1` FOREIGN KEY (`book_id`) REFERENCES `books` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

  - Load the data:

```
mysql -u tutorial -p myapp < myapp01_mysql.sql
```

- Make sure the data loaded correctly:

```
$ mysql -u tutorial -p myapp
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show tables;
+----------------+
| Tables_in_myapp |
+----------------+
| authors         |
| book_authors    |
| books           |
+----------------+
3 rows in set (0.00 sec)

mysql> select * from books;
+----+-----------------------------------+--------+
| id | title                             | rating |
+----+-----------------------------------+--------+
|  1 | CCSP SNRS Exam Certification Guide |      5 |
|  2 | TCP/IP Illustrated, Volume 1       |      5 |
|  3 | Internetworking with TCP/IP Vol.1  |      4 |
|  4 | Perl Cookbook                      |      5 |
|  5 | Designing with Web Standards       |      5 |
+----+-----------------------------------+--------+
5 rows in set (0.00 sec)

mysql>
```

- Update the model:
  - Delete the existing model:

```
rm lib/MyApp/Model/MyAppDB.pm
```

  - Regenerate the model using the Catalyst "_create.pl" script:

```
script/myapp_create.pl model DB DBIC::Schema MyApp::Schema create=static \
    dbi:mysql:myapp 'tutorial' 'yourpassword' '{ AutoCommit => 1 }'
```

- Chapter 5: Authentication
  - Create the `.sql` file for the user/roles data:

    Open `myapp02_mysql.sql` in your editor and enter:

```
--
-- Add users and roles tables, along with a many-to-many join table
--
CREATE TABLE IF NOT EXISTS `roles` (
  `id` int(11) NOT NULL,
  `role` text CHARACTER SET utf8,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE IF NOT EXISTS `users` (
  `id` int(11) NOT NULL,
  `username` text CHARACTER SET utf8,
  `password` text CHARACTER SET utf8,
  `email_address` text CHARACTER SET utf8,
  `first_name` text CHARACTER SET utf8,
  `last_name` text CHARACTER SET utf8,
  `active` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE IF NOT EXISTS `user_roles` (
  `user_id` int(11) NOT NULL DEFAULT '0',
  `role_id` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`user_id`,`role_id`),
  KEY `role_id` (`role_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
--
-- Load up some initial test data
--
INSERT INTO `roles` (`id`, `role`) VALUES
(1, 'user'),
(2, 'admin');

INSERT INTO `users` (`id`, `username`, `password`, `email_address`, `first_name`, `last_name`, `active`) VALUES
(1, 'test01', 'mypass', 't01@na.com', 'Joe', 'Blow', 1),
(2, 'test02', 'mypass', 't02@na.com', 'Jane', 'Doe', 1),
(3, 'test03', 'mypass', 't03@na.com', 'No', 'Go', 0);

INSERT INTO `user_roles` (`user_id`, `role_id`) VALUES
```

```
    (1, 1),
    (2, 1),
    (3, 1),
    (1, 2);

    ALTER TABLE `user_roles
    ADD CONSTRAINT `user_role_ibfk_2` FOREIGN KEY (`role_id`) REFERENCES `roles` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
    ADD CONSTRAINT `user_role_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

- Load the user/roles data:

```
    mysql -u tutorial -p myapp < myapp02_mysql.sql
```

- Update the model:
  - Regenerate the model using the Catalyst "_create.pl" script:

```
    script/myapp_create.pl model DB DBIC::Schema MyApp::Schema create=static \
        components=TimeStamp,PassphraseColumn dbi:mysql:myapp 'tutorial' 'yourpassword' '{ AutoCommit => 1 }'
```

- Create the `.sql` file for the hashed password data:

  Open `myapp03_mysql.sql` in your editor and enter:

```
    --
    -- Convert passwords to SHA-1 hashes
    --
    UPDATE users SET password = '{SSHA}esgz64CpHMo8pMfgIIszP13ft23z/zio04aCwNdm0wc6MDeloMUH4g==' WHERE id = 1;
    UPDATE users SET password = '{SSHA}FpGhpCJus+Ea9ne4ww8404HH+hJKW/fW+bAv1v6FuRUy2G7I2aoTRQ==' WHERE id = 2;
    UPDATE users SET password = '{SSHA}ZyGlpiHls8qFBSbHr3r5t/iqcZE602XLMbkSVRRNl6rF8imv1abQVg==' WHERE id = 3;
```

- Load the user/roles data:

```
    mysql -u tutorial -p myapp < myapp03_mysql.sql
```

AUTHOR

Kennedy Clark, `hkclark@gmail.com`

Feel free to contact the author for any errors or suggestions, but the best way to report issues is via the CPAN RT Bug system at https://rt.cpan.org/Public/Dist/Display.html?Name=Catalyst-Manual.