Implementing single table inheritance

Relational databases do not support inheritance. If we need to store inheritance in the database, we should  
somehow support it through code. This code should be efficient, so that it should generate as few JOINs  
as possible. A common solution to this problem was described by Martin Fowler and is named single  
table inheritance.

When we use this pattern, we store all the class tree data in a single table and use the type field to  
determine a model for each row.

As an example, we will implement the single table inheritance for the following class tree:

Car

|- SportCar  
|- FamilyCar

Getting ready

1. Create a new application using the Composer package manager, as described in the official guide at  
   <http://www.yiiframework.com/doc-2.0/guide-start-installation.html>.
2. Create and set up a database. Add the following table:

DROP TABLE IF EXISTS 'car';

CREATE TABLE 'car' (

'id' int(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

'name' varchar(255) NOT NULL,

'type' varchar(100) NOT NULL,

PRIMARY KEY ('id')

);

INSERT INTO 'car' ('name', 'type')

VALUES ('Ford Focus', 'family'),

('Opel Astra', 'family'),

('Kia Ceed', 'family'),

('Porsche Boxster', 'sport'),

('Ferrari 550', 'sport');

1. Use Gii to create a Car model for the car table and generate ActiveQuery for the Car model.

How to do it...

1. Add the following method and property to models/CarQuery .php:

/\*\*

* @var  
  \*/

public $type;

/\*\*

* @param \yii\db\QueryBuilder $builder

\*

* @return \yii\db\Query  
  \*/

public function prepare($builder)

{

if ($this->type !== null) {

$this->andWhere(['type' => $this->type]);

}

return parent::prepare($builder);

}

1. Create models/sportcar.php as follows:

<?php

namespace app\models;  
use Yii;

/\*\*

* Class SportCar
* @package app\models  
  \*/

class SportCar extends Car  
{

const TYPE = 'sport';

/\*\*

* @return CarQuery  
  \*/

public static function find()

{

return new CarQuery(get\_called\_class(), ['where' => ['type' =>  
self : :TYPE]]);

}

/\*\*

* @param bool $insert

\*

* @return bool  
  \*/

public function beforeSave($insert)

{

$this->type = self::TYPE;

return parent::beforeSave($insert);

}

}

1. Create models/FamilyCar.php as follows:

<?php

namespace app\models;  
use Yii;

/\*\*

* Class FamilyCar
* @package app\models  
  \*/

class FamilyCar extends Car  
{

const TYPE = 'family';

/\*\*

* @return CarQuery

\*/

public static function find()

{

return new CarQuery(get\_called\_class(), ['where' => ['type' =>  
self::TYPE]]);

}

/\*\*

* @param bool $insert

\*

* @return bool  
  \*/

public function beforeSave($insert)

{

$this->type = self::TYPE;

return parent::beforeSave($insert);

}

}

1. Add the following method to models/Car .php:

/\*\*

* @param array $row

\*

\* @return Car|FamilyCar|SportCar  
\*/

public static function instantiate($row)

{

switch ($row['type']) {  
case SportCar::TYPE:

return new SportCar();  
case FamilyCar::TYPE:

return new FamilyCar();  
default:

return new self;

}

}

1. Add TestController with the following code:

<?php

namespace app\controllers;

use app\models\Car;  
use app\models\FamilyCar;  
use Yii;

use yii\helpers\Html;  
use yii\web\Controller;

/\*\*

* Class TestController
* @package app\controllers  
  \*/

class TestController extends Controller  
{

public function actionIndex()

{

echo Html::tag('h1', 'All cars');

$cars = Car::find()->all();  
foreach ($cars as $car) {

// Each car can be of class Car, SportCar or FamilyCar  
echo get\_class($car).' '.$car->name."<br />";

}

echo Html::tag('h1', 'Family cars');

$familyCars = FamilyCar::find()->all();  
foreach($familyCars as $car)

{

// Each car should be FamilyCar

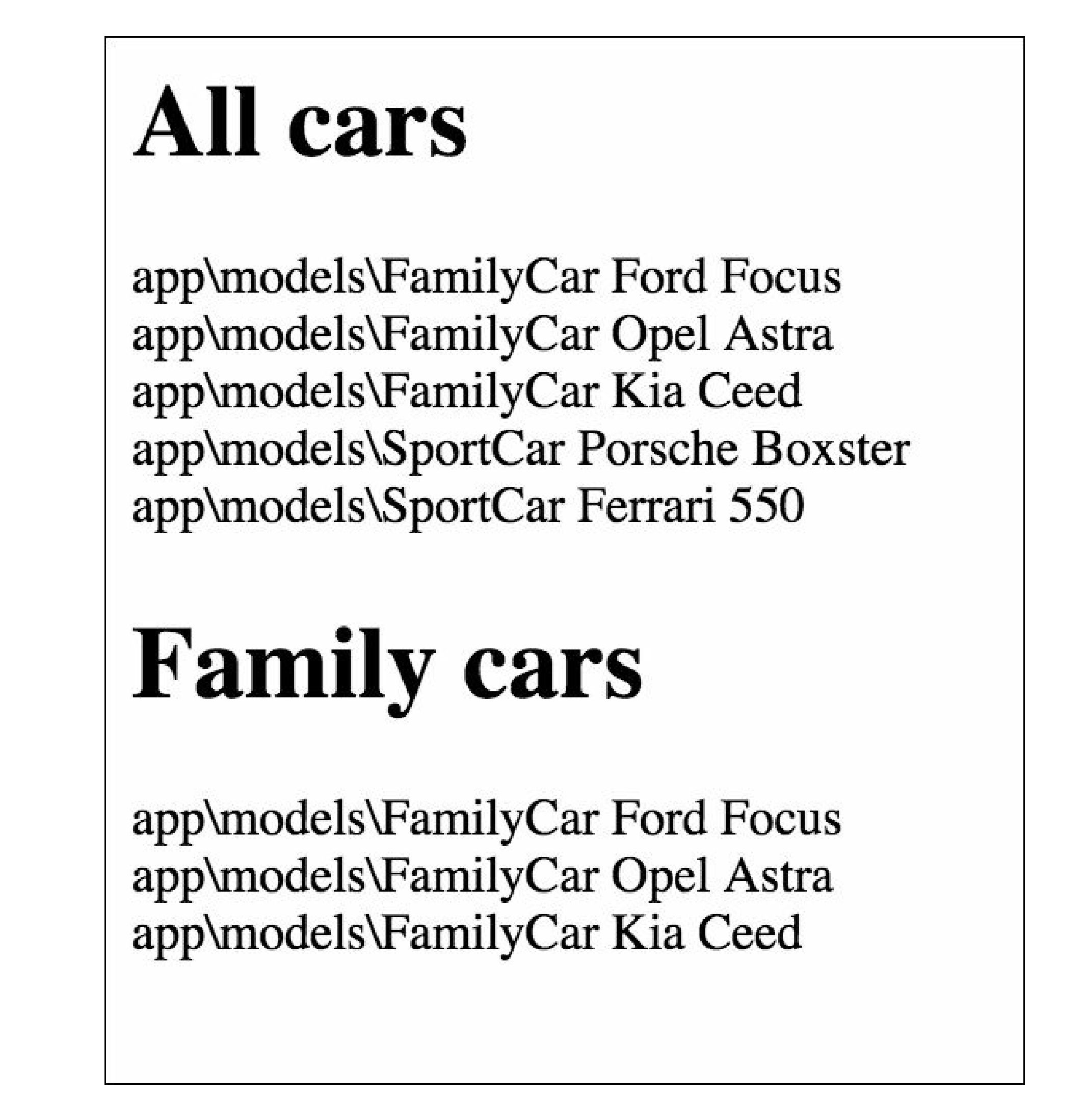
echo get\_class($car).' '.$car->name."<br />";

}

}

}

6. Run test/index and you should get the output shown in the following screenshot:



How it works...

The base model Car is a typically-used Yii AR model except that it has two added methods. The

tableName method explicitly declares the table name to be used for the model. For the Car model alone,  
this does not make sense, but for child models, it will return the same car table, which is just what we want  
—a single table for the entire class tree. The instantiate method is used by AR internally to create a model  
instance from the raw data when we call methods such as Car: : :find()->all(). We use a switch  
statement to create different classes based on the type attribute and use the same class if the attribute  
value is either not specified or points to the non-existing class.

The Sportcar and FamilyCar models simply set the default AR scope, so when we search for models  
with the SportCar: : model()-> methods, we will get the SportCar model only.

See also

Use the following references to learn more about the single table inheritance pattern and Yii Active  
Record implementation:

* <http://martinfowler.com/eaaCatalog/singleTableInheritance.html>
* <https://blog.liip.ch/archive/2012/03/27/table-inheritance-with-doctrine.html>
* <http://www.yiiframework.com/doc/api/CActiveRecord/>