一、准备

在开始编写代码之前，我们需要思考一下：用户登陆模块，实现的是什么功能？很明显，是登陆功能，那么，登陆需要用户名和密码，我们在数据库的一张表中就应该准备好用户名和密码的字段，再思考一下，如果要实现自动登陆的功能，那么还需要什么？Cookie，是专门用于自动登陆的，所以，我们的数据表可能需要准备一个字段，专门用于储存客户端登陆所生成的cookie，这样，就能通过验证客户端和服务端的cookie是否相同来进行自动登陆了。基于以上思考，我们的数据表应该包含以下字段：

id(primarykey,auto\_increment),

username(varchar),

password(varchar(32)),

auth\_key(varchar(32)),

accessToken(varchar(32))

(这个暂不解释，后文解释).

     1、首先，建立一个数据库：myDatabase,

     2、然后建立一张数据表：user，增加上述字段。

**二、模型（Model）**

Yii框架采用MVC设计模式，所以Model是一个模块的核心所在，所以我们先完成对Model的编写

**1、**LoginForm.php

用户登陆模块，所提交的是username和password，所以我们要先建立一个Model，专门处理用户提交的数据，所以先新建一个LoginForm.php,以下为代码：

1. <?php
3. namespace app\modules\backend\models;
5. **use** Yii;
6. **use** yii\base\Model;
8. /\*\*
9. \* LoginForm is the model behind the login form.
10. \*/
11. **class** LoginForm **extends** Model
12. {
13. **public** $username;
14. **public** $password;
15. **public** $rememberMe = true;
17. **private** $\_user = false;

20. /\*\*
21. \* @return array the validation rules.
22. \*/
23. **public** **function** rules()<span style="white-space:pre">     </span>//①
24. {
25. **return** [
26. // username and password are both required
27. [['username', 'password'], 'required','message'=>""],
28. // rememberMe must be a boolean value
29. ['rememberMe', 'boolean'],
30. // password is validated by validatePassword()
31. ['password', 'validatePassword'],
32. ];
33. }
35. /\*\*
36. \* Validates the password.
37. \* This method serves as the inline validation for password.
38. \*
39. \* @param string $attribute the attribute currently being validated
40. \* @param array $params the additional name-value pairs given in the rule
41. \*/
42. **public** **function** validatePassword($attribute, $params)
43. {
44. **if** (!$this->hasErrors()) {
45. $user = $this->getUser();
47. **if** (!$user || !$user->validatePassword($this->password)) {
48. $this->addError($attribute, 'Incorrect username or password.');
49. }
50. }
51. }
53. /\*\*
54. \* Logs in a user using the provided username and password.
55. \* @return boolean whether the user is logged in successfully
56. \*/
57. **public** **function** login()
58. {
59. **if** ($this->validate()) {
60. **if**($this->rememberMe)
61. {
62. $this->\_user->generateAuthKey();//③
63. }
64. **return** Yii::$app->user->login($this->getUser(), $this->rememberMe ? 3600\*24\*30 : 0);
65. }
66. **return** false;
67. }
69. /\*\*
70. \* Finds user by [[username]]
71. \*
72. \* @return User|null
73. \*/
74. **public** **function** getUser()
75. {
76. **if** ($this->\_user === false) {
77. $this->\_user = User::findByUsername($this->username); //②
78. }
80. **return** $this->\_user;
81. }
82. }

该Model是根据basic模板自带的LoginForm修改而成，代码中大多有注释，这里关注以下代码：

①号代码处是rules规则，rules规则定义了填充过来的数据的规则，验证所填的数据是否为空，是否符合格式之类的，其中有一栏是password,对应的规则是validatePassword，会自动调用当前类的validatePassword()方法，注意与下文的User类对应的方法区分。

②号代码，调用了User类里面的findByUsername方法，这个User类下面会写到，主要是为了返回一个AR类实例，与当前LoginForm的数据进行比较。

③号代码，这里暂时不提，等讲到cookie登陆的时候再提。

2、User.php

（1）ActiveRecord 类

在完成LoginForm后，我们还缺少一些东西，从用户接受到数据了，那么还需要从数据库取出相应的数据来进行比较，所以我们接下来需要完成的是一个从数据库获取的数据的类——AR类，全称是ActiveRecord，活动记录类，方便用于查找数据，只要类名和数据表的表名相同，那么它就能从这个数据表中获取数据，比如说这样：

1. <?php
2. namespace app\modules\backend\models;
3. **use** yii\db\ActiveRecord;
5. **class** User **extends** ActiveRecord{       } ?>

此外，还能自己添加返回的表名，只要在这个类中重写以下方法：

1. **public** **static** **function** tableName(){
2. **return** 'user';
3. }

（2）IdentityInterface 接口

一般来说，从数据库查找数据，只需要继承AR类即可，但是，我们这个是用户登录模型，核心是验证，所以自然需要实现核心的验证功能，就像LoginForm模型提到的validatePassword一样，实际的验证逻辑是在当前的User模型完成的。一般来说，实现IdentityInterface接口，需要实现以下方法：

1. **public** **static** **function** findIdentity($id);  //①
3. **public** **static** **function** findIdentityByAccessToken($token, $type = null);   //②
5. **public** **function** getId();    //③
7. **public** **function** getAuthKey();   //④
9. **public** **function** validateAuthKey($authKey);    //⑤

①findIdentity:是根据id查找数据表对应的数据

②findIdentityByAccessToken是根据AccessToken（上文提到的）查找对应的数据，而AccessToken我们在数据表也有这个字段，那么它到底有什么用呢？其实AccessToken在我们当前的用户登陆模型中用处并不大，它是专门用于Resetful登陆验证用到的，具体可自行百度，这里不展开说明。

③getId:返回当前AR类所对应的id

④getAuthKey:返回当前AR类所对应的auth\_key

⑤validateAuthKey:这个方法比较重要，是我们后面要讲到的cookie登陆验证的核心所在。

好了，既然知道了这五个方法的用处，那么我们在我们的User.php实现接口，然后重写以上方法，完整的User.php的代码如下：

1. <?php
2. namespace app\modules\backend\models;
3. **use** yii\db\ActiveRecord;
5. **class** User **extends** ActiveRecord **implements** \yii\web\IdentityInterface
6. {
8. **public** **static** **function** tableName(){
9. **return** 'user';
10. }
12. **public** **static** **function** findIdentity($id){
13. **return** **static**::findOne($id);
14. }
16. **public** **static** **function** findIdentityByAccessToken($token,$type=null){
17. **return** **static**::findOne(['accessToken'=>$token]);
18. }
20. **public** **static** **function** findByUsername($username){     //①
21. **return** **static**::findOne(['username'=>$username]);
22. }
24. **public** **function** getId(){
25. **return** $this->id;
26. }
28. **public** **function** getAuthkey(){
29. **return** $this->auth\_key;
30. }
32. **public** **function** validateAuthKey($authKey){
33. **return** $this->auth\_key === $authKey;
34. }
36. **public** **function** validatePassword($password){          //②
37. **return** $this->password === md5($password);
38. }
40. <span style="white-space:pre"> </span> /\*\*
41. <span style="white-space:pre">    </span> \* Generates "remember me" authentication key
42. <span style="white-space:pre">    </span> \*/
43. **public** **function** generateAuthKey()                    //③
44. {
45. <span style="white-space:pre">     </span>$this->auth\_key = \Yii::$app->security->generateRandomString();
46. <span style="white-space:pre">     </span>$this->save();
47. }
49. }
50. ?>

这里分析其中的三个方法：

①findByUsername():在LoginForm的代码中，引用了这个方法，目的是根据用户提交的username返回一个在数据表与username相同的数据项，即AR实例。

②validatePassword()：这里对用户提交的密码以及当前AR类的密码进行比较。

③generateAuthKey()：生成随机的auth\_key，用于cookie登陆。

到此，我们完成了Model的编写，一共写了两个Model类：LoginForm和User，一个用于接收用户提交的数据，一个用于获取数据库的数据，接下来我们编写Controller.

三、控制器（Controller）

控制器，主要是用于数据的提交，把用户提交的数据填充到相应的模型（Model）中，然后根据模型返回的信息进一步渲染视图（View），或者执行其他逻辑。

     这里，把控制器命名为LoginController.php，以下是完整的实现代码：

1. <?php
3. namespace app\controllers;
5. **use** Yii;
6. **use** yii\filters\AccessControl;
7. **use** yii\web\Controller;
8. **use** yii\filters\VerbFilter;
9. **use** app\models\LoginForm;
10. **use** app\models\ContactForm;
12. **class** SiteController **extends** Controller
13. {
14. **public** **function** actionIndex()
15. {
16. **return** $this->render('index');
17. }
19. **public** **function** actionLogin()
20. {
21. **if** (!\Yii::$app->user->isGuest) {     //①
22. **return** $this->goHome();
23. }
25. $model = **new** LoginForm();             //②
26. **if** ($model->load(Yii::$app->request->post()) && $model->login()) {      //③
27. **return** $this->goBack();          //④
28. }
29. **return** $this->render('login', [      //⑤
30. 'model' => $model,
31. ]);
32. }
34. **public** **function** actionLogout()
35. {
36. Yii::$app->user->logout();
38. **return** $this->goHome();
39. }
40. }

关注其中的actionLogin()方法：

①首先从\Yii::$app->user->isGuest中判断，当前是否是游客模式，即未登陆状态，如果用户已经登陆，会在user类中储存当前登陆用户的信息。

②如果当前是游客，会先实例化一个LoginForm模型

③这行代码是整个login方法的核心所在，首先：$model->load(Yii::$app->request->post())把post过来的数据填充进$model，即LoginForm模型，如果返回true，则填充成功。接着：$model->login()：执行LoginForm类里面的login()方法，可以从login()方法里面看到，将会执行一系列的验证。

关于Yii框架到底是怎样进行用户登陆的，底层是怎样实现的，我们在下一篇文章详谈，这里先说明实现方法。

**四、视图（View）**

在实现了model和controller，接下来是视图部分，由于用户需要输入数据，所以我们要提供一个表单，在Yii2中，提供了ActiveForm快速生成表单，代码如下：

1. <?php
3. /\* @var $this yii\web\View \*/
4. /\* @var $form yii\bootstrap\ActiveForm \*/
5. /\* @var $model app\models\LoginForm \*/
7. **use** yii\helpers\Html;
8. **use** yii\bootstrap\ActiveForm;
10. $this->title = 'Login';
11. $this->params['breadcrumbs'][] = $this->title;
12. ?>
13. <div **class**="site-login">
14. <h1><?= Html::encode($this->title) ?></h1>
16. <p>Please fill out the following fields to login:</p>
18. <?php $form = ActiveForm::begin([
19. 'id' => 'login-form',
20. 'options' => ['class' => 'form-horizontal'],
21. 'fieldConfig' => [
22. 'template' => "{label}\n<div class=\"col-lg-3\">{input}</div>\n<div class=\"col-lg-8\">{error}</div>",
23. 'labelOptions' => ['class' => 'col-lg-1 control-label'],
24. ],
25. ]); ?>
27. <?= $form->field($model, 'username')->textInput(['autofocus' => true]) ?>
29. <?= $form->field($model, 'password')->passwordInput() ?>
31. <?= $form->field($model, 'rememberMe')->checkbox([
32. 'template' => "<div class=\"col-lg-offset-1 col-lg-3\">{input} {label}</div>\n<div class=\"col-lg-8\">{error}</div>",
33. ]) ?>
35. <div **class**="form-group">
36. <div **class**="col-lg-offset-1 col-lg-11">
37. <?= Html::submitButton('Login', ['class' => 'btn btn-primary', 'name' => 'login-button']) ?>
38. </div>
39. </div>
41. <?php ActiveForm::end(); ?>
43. <div **class**="col-lg-offset-1" style="color:#999;">
44. You may login with <strong>admin/admin</strong> **or** <strong>demo/demo</strong>.<br>
45. To modify the username/password, please check out the code <code>app\models\User::$users</code>.
46. </div>
47. </div>

$form=ActiveForm::begin() :创建一个Form表单

$form=field()->textInput()   :创建一个文本输入框

$form=field()->checkbox() :创建一个checkbox

Html::submitButton():          创建一个登陆按钮

ActiveForm::end()    :   结束表单

以上，就是创建一个用户登陆模块的全流程，这里对用户登陆的细节和怎样实现cookie自动登陆见下一小节

**五、详细登录流程**

上节中，讲述了yii2应用用户登陆的基本方法，但是这些方法到底是怎样实现登陆的呢？底层的原理到底是什么？在这节将从Yii的源码角度分析登陆的基本原理以及cookie自动登陆的原理，通过源码的分析，对Yii的理解也会更上一层楼。

一、第一次正常登陆

     1、在LoginForm.php中，我们曾经调用了这个方法：

1. Yii::$app->user->login($this->getUser(), $this->rememberMe ? 3600\*24\*30 : 0);

由以上可知，调用了user组件的login()方法，把两个参数传递进入，分别是User类实例以及记住时间(Cookie验证会用到，如果传递0，则不启用Cookie验证)。

     2、进入yii\web\user类中，找到login()方法如下所示：

1. **public** **function** login(IdentityInterface $identity, $duration = 0)
2. {
3. **if** ($this->beforeLogin($identity, false, $duration)) {
4. $this->switchIdentity($identity, $duration);     //①
5. $id = $identity->getId();
6. $ip = Yii::$app->getRequest()->getUserIP();
7. **if** ($this->enableSession) {
8. $log = "User '$id' logged in from $ip with duration $duration.";
9. } **else** {
10. $log = "User '$id' logged in from $ip. Session not enabled.";
11. }
12. Yii::info($log, **\_\_METHOD\_\_**);
13. $this->afterLogin($identity, false, $duration);
14. }
16. **return** !$this->getIsGuest();
17. }

 这里关注①号处代码：$this->switchIdentity($identity,$duration).这里调用了当前类的switchIdentity方法，把接受到的两个参数同时传递进去，我们往下看：

3、switchIdentity($identity,$duration)方法如下：

1. **public** **function** switchIdentity($identity, $duration = 0)
2. {
3. <span style="white-space:pre">    </span>...
4. <span style="white-space:pre">    </span>...
5. **if** ($identity) {
6. ...
7. **if** ($duration > 0 && $this->enableAutoLogin) {<span style="white-space:pre">    </span>//①
8. $this->sendIdentityCookie($identity, $duration); //②
9. }
10. } **elseif** ($this->enableAutoLogin) {
11. Yii::$app->getResponse()->getCookies()->remove(**new** Cookie($this->identityCookie));
12. }
13. }

由于代码过长，笔者做了适当精简，只讨论与登陆和cookie联系最为密切的部分，由①处代码，首先会对duration进行判断，只有大于0的情况下才会进行cookie验证，然后再判断了enableAutoLogin的值，这个值也是cookie验证的关键所在，只有为true的时候才会储存cookie，该值在config/main.php中注册user组件的时候进行初始化，代码如下：

1. 'components' => [
2. 'user' => [
3. 'identityClass' => 'app\modules\backend\models\User',
4. 'enableAutoLogin' => true,
5. ],]

在判断都为真的时候，即进行cookie储存和登陆的时候，进入②号代码，可以看到，调用了sendIdentityCookie()方法。

4、sendIdentityCookie($identity,$duration):

1. **protected** **function** sendIdentityCookie($identity, $duration)
2. {
3. $cookie = **new** Cookie($this->identityCookie);
4. //cookie的value值是json数据
5. $cookie->value = json\_encode([
6. $identity->getId(),
7. $identity->getAuthKey(),
8. $duration,
9. ], JSON\_UNESCAPED\_SLASHES | JSON\_UNESCAPED\_UNICODE);
10. //设置cookie的有效时间
11. $cookie->expire = time() + $duration;
12. Yii::$app->getResponse()->getCookies()->add($cookie);  //①
13. }

在生成一个cookie的实例以及对cookie进行初始化后，接下来来到了重点部分，即①号代码处，首先getResponse()得到response组件，然后调用response的getCookies方法，返回一个CookieCollection实例，该实例保存了response组件所生成的所有cookie，显然，最后一个add()方法是把当前设置好的cookie保存进CookieCollection实例中。  
到目前为止，yii已经完成了储存cookie的操作，但是，有一点要注意的，利用yii框架的这种方式储存cookie，要记住一点，就是在Controller那里，通过验证后，应该使用redirect()来进行页面跳转，而不应该直接render()渲染布局，否则，当前登陆完成后，cookie将暂时得不到保存，(详细可通过查看浏览的cookie缓存来查看)如果cookie得不到立即的储存，有可能对后续用户的登陆造成未知困扰。

为什么会出现这样的情况呢？

我们来看看response组件的redirect()方法，(一般通过Yii::$app->response->redirect()来跳转url)：

1. **public** **function** redirect($url, $statusCode = 302, $checkAjax = true)
2. {
3. ...
4. ...
6. **return** $this;
7. }

关键在于最后的return $this;这句表示把当前类作为整个反应返回给客户端，换句话说：即当前保存的所有cookie，以及各种其他属性，在调用了redirect后全部返回。所以，我们在控制器需要使用redirect()，而不是直接使用render()方法。  
综上所述，在完成了user组件的login()方法后，用户的个人信息便保存于user组件中，直到用户关闭浏览器或者退出登录。

二、利用Cookie登陆

在用户关闭浏览器的时候（非退出），再次访问登陆页面，会发现页面已经自动跳转到主页，也即是说完成了自动登陆的功能（前提是点击了Remember Me），我们回顾一下logincontroller里面关于登陆的逻辑：

1. **if** (!\Yii::$app->user->isGuest) {
2. **return** $this->goHome();
3. }

可以看出，当访问登陆页面的时候，会先执行判断，判断当前用户是否是游客，若不是，则直接跳转到主页。所以关于自动登陆的逻辑便隐藏在：Yii::$app -> user ->isGuest 中，我们查看user组件相关代码：

1. **public** **function** getIsGuest()
2. {
3. **return** $this->getIdentity() === null;
4. }

在以上方法中，调用了getIdentity()方法：

1. **public** **function** getIdentity($autoRenew = true)
2. {
3. **if** ($this->\_identity === false) {
4. **if** ($this->enableSession && $autoRenew) {
5. $this->\_identity = null;
6. $this->renewAuthStatus();
7. } **else** {
8. **return** null;
9. }
10. }
12. **return** $this->\_identity;
13. }

由于user组件默认是开始session的，所以enableSession应该为true，所以会执行renewAuthStatus()函数：

**[php]** [view plain](http://blog.csdn.net/a553181867/article/details/50994388) [copy](http://blog.csdn.net/a553181867/article/details/50994388)

[print?](http://blog.csdn.net/a553181867/article/details/50994388)

1. **protected** **function** renewAuthStatus()
2. {
3. ...
4. **if** ($this->enableAutoLogin) {
5. **if** ($this->getIsGuest()) {
6. $this->loginByCookie();
7. } **elseif** ($this->autoRenewCookie) {
8. $this->renewIdentityCookie();
9. }
10. }
11. }

结合之前的enableAutoLogin为true以及当前处于未登录状态，所以getIsGuest()返回真，所以最后会执行loginByCookie()方法，这也是核心所在：

1. **protected** **function** loginByCookie()
2. {<span style="white-space:pre">   </span>//从客户端读取cookie
3. $value = Yii::$app->getRequest()->getCookies()->getValue($this->identityCookie['name']);
4. **if** ($value === null) {
5. **return**;
6. }
7. <span style="white-space:pre">    </span>//由于之前储存cookie是用json格式储存，所以现在需要先解析
8. $data = json\_decode($value, true);
9. **if** (count($data) !== 3 || !isset($data[0], $data[1], $data[2])) {
10. **return**;
11. }
13. list ($id, $authKey, $duration) = $data;
14. /\* @var $class IdentityInterface \*/
15. $class = $this->identityClass;<span style="white-space:pre">   </span>//读取当前的用户验证类类名，即实现了Identity接口的类
16. $identity = $class::findIdentity($id);<span style="white-space:pre">  </span>//调用该类的方法，从数据库查找数据
17. **if** ($identity === null) {
18. **return**;
19. } **elseif** (!$identity instanceof IdentityInterface) {
20. **throw** **new** InvalidValueException("$class::findIdentity() must return an object implementing IdentityInterface.");
21. }
22. <span style="white-space:pre">    </span>//如果数据库提供的auth\_key与从客户取得的auth\_key相同
23. **if** ($identity->validateAuthKey($authKey)) {
24. **if** ($this->beforeLogin($identity, true, $duration)) { //①
25. $this->switchIdentity($identity, $this->autoRenewCookie ? $duration : 0);
26. $ip = Yii::$app->getRequest()->getUserIP();
27. Yii::info("User '$id' logged in from $ip via cookie.", **\_\_METHOD\_\_**);
28. $this->afterLogin($identity, true, $duration);
29. }
30. } **else** {
31. Yii::warning("Invalid auth key attempted for user '$id': $authKey", **\_\_METHOD\_\_**);
32. }
33. }

由①可知，在auth\_key验证通过后，所执行的代码基本与首次执行的login()方法相同，即实现了重新登录的功能。

关于Yii2用户登陆功能的实现以及cookie自动登陆的原理已经全部讲述完毕。