Лабораторна робота №5

Тема:Створення власної соціальної мережі
Мета:Навчитись використовувати хешування, працювати з файлами та криптографією, безпекою, аутентифікацією користувачів.

Загальні теоретичні відомості

Аутентифікація

Аутентифікація - це процес перевірки автентичності користувача. Зазвичай використовується ідентифікатор (наприклад, username або адресу електронної пошти) і секретний токен (наприклад, пароль або ключ доступу), щоб судити про те, що користувач саме той, за кого себе видає. Аутентифікація є основною функцією форми входу.
Yii надає фреймворк авторизації з різними компонентами, що забезпечують процес входу. Для використання цього фреймворка вам потрібно виконати наступне:
Налаштувати компонент додатка user;
Створити клас, який реалізує інтерфейс yii \ web \ IdentityInterface.
Настройкаyii \ web \ User
Компонент user управляє статусом аутентифікації користувача. Він вимагає, щоб ви вказали identity class, який буде містити поточну логіку аутентифікації. У наступній конфігурації програми, identity class для user заданий як app \ models \ User, реалізація якого буде пояснена в наступному розділі:

```
return [
   'Components' => [
      'User' => [
         'IdentityClass' => 'app \ models \ User',
      ],
   ],
];
```

Реалізаціяyii \ web \ IdentityInterface

identity class повинен реалізовувати yii \ web \ IdentityInterface, який містить наступні методи:
findIdentity (): Цей метод знаходить екземпляр identity class, використовуючи ID користувача. Цей метод використовується, коли необхідно підтримувати стан аутентифікації через сесії.
findIdentityByAccessToken (): Цей метод знаходить екземпляр identity class, використовуючи токен доступу. Метод використовується, коли потрібно аутентифицировать користувача тільки по секретному токені (наприклад в RESTful додатках, які не зберігають стан між запитами).
getId (): Цей метод повертає ID користувача, представленого даними екземпляром identity.

getAuthKey (): Цей метод повертає ключ, який використовується для заснованої на cookie аутентифікації. Ключ зберігається в аутентификационной cookie і пізніше порівнюється з версією, що знаходиться на сервері, щоб упевнитися, що аутентифікаційних cookie вірна.
validateAuthKey (): Цей метод реалізує логіку перевірки ключа для заснованої на cookie аутентифікації.
Якщо якийсь із методів не потрібно, то можна реалізувати його з порожнім тілом. Для прикладу, якщо у вас RESTful програму без зберігає стан між запитами, ви можете реалізувати тільки findIdentityByAccessToken () і getId (), тоді як інші методи залишити порожніми.
У наступному прикладі, identity class реалізований як клас Active Record, пов'язаний з таблицею user.

```
<? Php

use yii \ db \ ActiveRecord;
use yii \ web \ IdentityInterface;

class User extends ActiveRecord implements IdentityInterface
{
   public static function tableName ()
   {
      return 'user';
   }

   / **
    * Finds an identity by the given ID.
    *
    * @param string | int $ id the ID to be looked for
    * @return IdentityInterface | null the identity object that matches the given ID.
    * /
   public static function findIdentity ($ id)
   {
      return static :: findOne ($ id);
   }

   / **
    * Finds an identity by the given token.
    *
    * @param string $ token the token to be looked for
    * @return IdentityInterface | null the identity object that matches the given token.
    * /
   public static function findIdentityByAccessToken ($ token, $ type = null)
   {
      return static :: findOne ([ 'access_token' => $ token]);
   }

   / **
    * @return int | string current user ID
    * /
   public function getId ()
   {
      return $ this-> id;
   }

   / **
```

```
    * @return string current user auth key
    * /
   public function getAuthKey ()
   {
      return $ this-> auth_key;
   }

   / **
    * @param string $ authKey
    * @return bool if auth key is valid for current user
    * /
   public function validateAuthKey ($ authKey)
   {
      return $ this-> getAuthKey () === $ authKey;
   }
}
```

Як пояснювалося раніше, вам потрібно реалізувати тільки getAuthKey () і validateAuthKey (), якщо ваше додаток використовує тільки аутентифікацію засновану на cookie. В цьому випадку ви можете використовувати наступний код для генерації ключа аутентифікації для кожного користувача і зберігання його в таблиці user:

```
class User extends ActiveRecord implements IdentityInterface
{
   ......

   public function beforeSave ($ insert)
   {
      if (parent :: beforeSave ($ insert)) {
         if ($ this-> isNewRecord) {
            $ This-> auth_key = \ Yii :: $ app-> security-> generateRandomString ();
         }
         return true;
      }
      return false;
   }
}
```

Note: Не плутайте identity клас User з класом yii \ web \ User. Перший є класом, які реалізують логіку аутентифікації користувача. Він часто реалізується як клас Active Record, пов'язаний з деяким постійним сховищем, де лежить інформація про користувачів. Другий - це клас компонента додатка, який відповідає за управління станом аутентифікації користувача.

Використання yii \ web \ User

В основному клас yii \ web \ User використовують як компонент додатка user. Можна отримати identity поточного користувача, використовуючи вислів Yii :: $ app-> user-> identity. Воно поверне примірник identity class, який представляє поточного аутентифицированного користувача, або null, якщо поточний користувач не аутентифікований (наприклад, гість). Наступний код показує, як отримати іншу пов'язану з аутентифікацією інформацію з yii \ web \ User:

```
// `identity` поточного користувача. `Null`, якщо користувач не аутентифікований.
$ Identity = Yii :: $ app-> user-> identity;

// ID поточного користувача. `Null`, якщо користувач не аутентифікований.
$ Id = Yii :: $ app-> user-> id;

// перевірка на те, що поточний користувач гість (НЕ аутентифікований)
$ IsGuest = Yii :: $ app-> user-> isGuest;
```

Для залогініванія користувача ви можете використовувати наступний код:

```
// знайти identity із зазначеним username.
// зауваження: також ви можете перевірити і пароль, якщо це потрібно
$ Identity = User :: findOne ([ 'username' => $ username]);

// логін користувача
Yii :: $ app-> user-> login ($ identity);
```

Метод yii \ web \ User :: login () встановлює identity поточного користувача в yii \ web \ User. Якщо сесії включені, то identity буде зберігатися в сесії, так що стан статусу аутентифікації буде підтримуватися на всьому протязі сесії. Якщо включений вхід, заснований на cookie (так званий "запам'ятай мене" вхід), то identity також буде збережена в cookie так, щоб статус аутентифікації користувача міг бути відновлений протягом усього часу життя cookie.

Для включення входу, заснованого на cookie, вам потрібно встановити yii \ web \ User :: enableAutoLogin в true в конфігурації програми. Ви також можете налаштувати час життя, передавши його при виклику методу yii \ web \ User :: login ().

Для виходу користувача, просто викличте

Yii :: $ app-> user-> logout ();

Зверніть увагу: вихід користувача має сенс тільки якщо сесії включені. Метод скидає статус аутентифікації відразу і з пам'яті і з сесії. І за замовчуванням, будуть також знищені всі сесійні дані користувача. Якщо ви хочете зберегти сесійні дані, ви повинні замість цього викликати Yii :: $ app-> user-> logout (false).

події аутентифікації
Клас yii \ web \ User викликає кілька подій під час процесів входу і виходу.

EVENT_BEFORE_LOGIN: викликається перед викликом yii \ web \ User :: login (). Якщо обробник встановлює властивість isValid об'єкта в false, процес входу буде перерваний.
EVENT_AFTER_LOGIN: викликається після успішного входу.

EVENT_BEFORE_LOGOUT: викликається перед викликом yii \ web \ User :: logout (). Якщо обробник встановлює властивість isValid об'єкта в false, процес виходу буде перерваний.
EVENT_AFTER_LOGOUT: викликається після успішного виходу.
Ви можете використовувати ці події для реалізації функції аудиту входу, збору статистики онлайн користувачів. Наприклад, в обробнику для EVENT_AFTER_LOGIN ви можете зробити запис про час і IP адресу входу в таблицю user.

Робота з паролями
Багато розробники знають, що зберігати пароль відкритим текстом не можна, але багато хто до цих пір вважають безпечним використання для хешування паролів md5 або sha1. Раніше згаданих алгоритмів було досить, але сучасне обладнання дозволяє підібрати ці хеші дуже швидко, методом простого перебору.
Для того, щоб забезпечити підвищену безпеку паролів ваших користувачів навіть в гіршому випадку (ваше додаток зламано), потрібно використовувати алгоритм шифрування, стійкий до атаки перебором. Кращий варіант в поточний момент bcrypt. У PHP ви можете використовувати хеш-кодування bcrypt через функцію crypt. Yii забезпечує дві допоміжні функції, які спрощують використання функції crypt для генерації і перевірки пароля.
Коли користувач задає пароль (наприклад під час реєстрації), пароль повинен бути захешірован:
$ Hash = Yii :: $ app-> getSecurity () -> generatePasswordHash ($ password);
Хеш можна пов'язати з відповідним атрибутом моделі, так щоб він зберігався в базі для подальшого використання.
Коли користувач спробує увійти, висланий пароль повинен бути хешировать і сравнён з раніше збереженим хешем:

```
if (Yii :: $ app-> getSecurity () -> validatePassword ($ password, $ hash)) {
   // все добре, користувач може увійти
} Else {
   // неправильний пароль
}
```

Фільтрація введення
Фільтрація введення означає, що вхідні дані ніколи не повинні вважатися безпечними і ви завжди повинні перевіряти, чи є отримані дані допустимими. Наприклад, якщо ми знаємо, що сортування може бути здійснена тільки за трьома полями title, created_at і status, і поле може передаватися через введення користувачем, краще перевірити значення там, де ми його отримали:

```
$ SortBy = $ _GET [ 'sort'];
if (! in_array ($ sortBy, [ 'title', 'created_at', 'status'])) {
    throw new Exception ( 'Invalid sort value.');
```

}
В Yii, ви, швидше за все, будете використовувати валідацію форм, щоб робити такі перевірки.

Екранування виведення
Екранування виведення означає, що дані в залежності від контексту повинні екрануватися, наприклад в контексті HTML ви повинні екранувати <,> і схожі спеціальні символи. В контексті JavaScript або SQL буде інший набір символів. Так як ручне екранування значною кількістю помилок, Yii надає різні утиліти для екранування в різних контекстах.

Як уникнути SQL-ін'єкцій
SQL-ін'єкції відбуваються, коли текст запиту формується склеюванням НЕ екранованих рядків, як показано нижче:

```
$ Username = $ _GET [ 'username'];
$ Sql = "SELECT * FROM user WHERE username = '$ username'";
```

Замість того, щоб підставляти коректне ім'я користувача, зловмисник може передати вам в додаток щось на зразок '; DROP TABLE user; -. В результаті SQL буде наступний:

```
SELECT * FROM user WHERE username = ''; DROP TABLE user; - '
```

Це валідний запит, який спочатку буде шукати користувачів з порожнім ім'ям, а потім видалить таблицю user. Швидше за все буде зламано додаток і будуть втрачені дані (ви адже робите регулярне резервне копіювання?).
Більшість запитів до бази даних в Yii відбувається через Active Record, який правильно використовує підготовлені запити PDO всередині. При використанні підготовлених запитів неможливо маніпулювати запитом як це показано вище.
Проте, іноді потрібні сирі запити або будівник запитів. В цьому випадку ви повинні використовувати безпечні способи передачі даних. Якщо дані використовуються для порівняння зі значенням стовпців краще використовувати підготовлені запити:

```
// query builder
$ UserIDs = (new Query ())
    -> select ( 'id')
    -> from ( 'user')
    -> where ( 'status =: status', [ ': status' => $ status])
    -> all ();

// DAO
$ UserIDs = $ connection
    -> createCommand ( 'SELECT id FROM user where status =: status')
```

```
   -> bindValues ([ ': status' => $ status])
   -> queryColumn ();
```

Якщо дані використовуються в якості імен стовпців або таблиць, то кращий шлях - це дозволити тільки зумовлений набір значень:

```
function actionList ($ orderBy = null)
{
   if (! in_array ($ orderBy, [ 'name', 'status'])) {
      throw new BadRequestHttpException ( 'Only name and status are allowed to order by.')
   }

   // ...
}
```

Якщо це неможливо, то імена стовпців і таблиць повинні екрануватися. Yii використовує спеціальний синтаксис для екранування для всіх підтримуваних баз даних:

```
$ Sql = "SELECT COUNT ([[$ column]]) FROM {{table}}";
$ RowCount = $ connection-> createCommand ($ sql) -> queryScalar ();
```

Ви можете отримати детальну інформацію про синтаксис в Екранування імен таблиць і стовпців.

Як уникнути XSS

XSS або крос-сайтінговий скриптинг стає можливий, коли НЕ екранований вихідний HTML потрапляє в браузер. Наприклад, якщо користувач повинен ввести своє ім'я, але замість Alexander він вводить <script> alert ('Hello!'); </ Script>, то всі сторінки, які його виводять без екранування, будуть виконувати JavaScript alert ('Hello!') ;, і в результаті буде виводитися вікно повідомлення в браузері. Залежно від сайту, замість невинних скриптів з висновком спливаючого hello, зловмисниками можуть бути відправлені скрипти, що викрадають особисті дані користувачів сайту, або виконують операції від їх імені.

В Yii уникнути XSS легко. На місці виведення тексту необхідно вибрати один з двох варіантів:

Ви хочете вивести дані у вигляді звичайного тексту.

Ви хочете вивести дані в вигляді HTML.

Якщо вам потрібно вивести простий текст, то екранувати краще наступним чином:

```
<? = \ Yii \ helpers \ Html :: encode ($ username)?>
```

Якщо потрібно вивести HTML, вам краще скористатися HtmlPurifier:

<? = \ Yii \ helpers \ HtmlPurifier :: process ($ description)?>
Зверніть увагу, що обробка за допомогою HtmlPurifier досить важка, так що непогано було б задуматися про кешування.

Як уникнути CSRF
CSRF - це абревіатура для межсайтінговой підміни запитів. Ідея полягає в тому, що багато програм припускають, що запити, які надходять від браузера, відправляються самим користувачем. Це може бути неправдою.

Наприклад, сайт an.example.com має URL / logout, який, використовуючи простий GET, разлогінівает користувача. Поки це запит виконується самим користувачем - все в порядку, але в один прекрасний день зловмисники розміщують код <img src = "http://an.example.com/logout"> на форумі з великою відвідуваністю. Браузер не робить ніяких відмінностей між запитом зображення і запитом сторінки, так що коли користувач відкриє сторінку з таким тегом img, браузер відправить GET запит на вказану адресу, і користувач буде разлогінен.

Ось основна ідея. Можна сказати, що в разлогіруванії користувача немає нічого серйозного, але за допомогою цієї уразливості, можна виконувати небезпечні операції. Уявіть, що існує сторінка http://an.example.com/purse/transfer?to=anotherUser&amount=2000, звернення до якої за допомогою GET запиту, призводить до перерахування 2000 одиниць валюти з рахунку авторизованого користувача на рахунок користувача з логіном anotherUser. З огляду на, що браузер для завантаження контенту відправляє GET запити, можна подумати, що дозвіл на виконання такої операції тільки POST запитом на 100% убезпечить від проблем. На жаль, це не зовсім правда. Враховуйте, що замість тега, зловмисник може впровадити JavaScript код, який буде відправляти потрібні POST запити на цей URL.

Для того, щоб уникнути CSRF ви повинні завжди:
Дотримуйтесь специфікаціям HTTP, наприклад запит GET не повинен міняти стан додатки.
Тримайте захист CSRF в Yii включеною.
Як уникнути небажаного доступу до файлів
За замовчуванням, webroot сервера вказує на каталог web, де лежить index.php. У разі використання віртуального хостингу, це може бути недосяжно, в кінцевому підсумку весь код, конфіги і логи можуть виявитися в webroot сервера.

Якщо це так, то потрібно заборонити доступ до всього, крім директорії web. Якщо на вашому хостингу таке неможливо, розгляньте можливість зміни хостингу.

Як уникнути виведення інформації налагодження і інструментів в робочому режимі
У режимі налагодження, Yii відображає досить докладні помилки, які корисні під час розробки. Справа в тому, що докладні помилки зручні для нападника, так як можуть розкрити структуру бази даних, параметрів конфігурації і частини вашого коду. Ніколи не запускайте додатки в робочому режимі з YII_DEBUG встановленим в true в вашому index.php.
Ви ніколи не повинні включати Gii або Debug панель в робочому режимі. Це може бути використано для отримання інформації про структуру бази даних, коду і може дозволити замінити файли, які генеруються Gii автоматично.
Слід уникати включення в робочому режимі панелі налагодження, якщо тільки в цьому немає гострої необхідності. Вона розкриває весь додаток і деталі конфігурації. Якщо Вам все-таки потрібно запустити панель налагодження в робочому режимі, перевірте двічі, що доступ обмежений тільки вашими IP-адресами.

Безпечна конфігурація сервера
Мета цього розділу - виявити ризики, які необхідно враховувати при створенні конфігурації сервера для обслуговування веб-сайту на основі Yii. Крім перерахованих тут пунктів є й інші параметри, пов'язані з безпекою, які необхідно враховувати, тому не можете робити висновків розділ як завершений.

Як уникнути атаки типу Host-header
Класи типу yii \ web \ UrlManager і yii \ helpers \ Url можуть використовувати запитувана ім'я хоста] для генерації посилань. Якщо веб-сервер налаштований на обслуговування одного і того ж сайту незалежно від значення заголовка Host, ця інформація може бути ненадійною і може бути підроблена користувачем, які відправляють HTTP-запит. У таких ситуаціях Ви повинні або виправити конфігурацію свого веб-сервера, щоб обслуговувати сайт тільки для зазначених імен вузлів, або явно встановити або відфільтрувати значення, встановивши властивість hostInfo компонента додатка request.
Додаткові відомості про конфігурацію сервера дивіться в документації Вашого веб-сервера:

Завдання для виконання
1. Створити базу даних (Користувач, Список друзів, Галерея).
2. Реалізувати, реєстрацію, авторизацію користувачів.
3. Дозволити авторизованим користувачам публікувати власні фотографії(з описом, датою додавання).
4. Дозволити користувачам додавати друзів(список усіх можна фільтрувати)

5. Друзі можуть оцінювати фотографії та коментувати.
6. Реалізувати зручний інтерфейс.
7. Передбачити безпеку користувачів(Запобігання міжсайтового скриптинга, Запобігання підробці міжсайтових запитів, апобігання атаки через cookie
8.)