Role Based Access Control

Using PHP Sessions

Session

- Developed in PHP to store client data on the web server, but keep a single session ID on the client machine (cookie)
- The session ID : identifies the user uniquely for the duration of the user's visit to your site
- PHP sessions are automatically set as a cookie in the user's browser that contains the session ID
- Your pages can check the session ID and do something with it.

Working with Session IDs

- To look for an existing session ID or create a new one:
  - session_start();
- Set a variable with session data:
  - $_SESSION[‘password’] = ‘iamawesome’;
- Remove a variable from the current session:
  - Unset($_SESSION[‘password’]);
- End the session and delete all registered variables
  - $_SESSION = array();
  - Session_destroy();

Access Control

- Require username and password authentication for some or all of your site
  - Require a valid login for relevant (protected) pages
  - Use PHP to prompt the user and check the login credentials in their session as appropriate

User Data Where?

- User authentication requires user data to be saved on the server
  - User name, password, email, etc.
- Use PHP to prompt the user and check the login credentials on the server to set variables.
  - At each protected page, check variables to confirm that the user is authorized

Roles

- Some pages may be protected from certain groups of users, but not others
- Assign a role to a user's login credentials. If the role allows access, the user can access the protected page for that role.
- Allows varying levels of protection
  - Site administrators
  - Content editors

The Controller

- The controller can have gatekeeper PHP code to only allow actions if the user is logged in. Place this gatekeeper code at the beginning of the controller to lock down the controller.
  - Artist's index.php restricts artists
  - Album's index.php restricts albums
- This example uses a function to check session:

```
if (!userIsLoggedIn())
{
 include '../login.html.php'; //go to login page
exit();
}
```

userIsLoggedIn()

- Create this function in a new file that must be included before the function can be used:
  - access.inc.php
- This function checks if the user is logged in

function userIsLoggedIn()

{

if (isset($_POST[‘action’]) and $_POST[‘action’] == ‘login’)

{

1. check for $_POST[‘email’] and $_POST[‘email’] from login form

2. Scramble password with md5

3. Query the database and match submitted data to database data (use another custom function so you can reuse this)

4. Login the user with a session variable. This can be checked again later once it is set because it is saved for the entire session. Example:

$_SESSION[‘loggedIn’] = TRUE

Note: You can store the login and password info instead in session variables so you can re-check them against the database during the session for added security.

The Controller

- The controller can have gatekeeper PHP code to only allow actions if the user is assigned to the appropriate role. Place this gatekeeper code at the beginning of the controller to lock down the controller.
  - Artist's index.php restricts artists
  - Album's index.php restricts albums
- This example uses a function to check role:

```
if (!userHasRole('Site Administrator'))
{
 $error = "Only Site Administrators Allowed";
Include '../accessdenied.html.php';
exit();
}
```

userHasRole($role)

- Query database for user roles
- If user is not assigned to the appropriate role, produce error and return FALSE, otherwise return TRUE

New Pages for Access Control

- login.html.php //login form
- accessdenied.html.php //error message
- access.inc.php //contains authorizing functions

User and Role Data

- User data: primary key, login, password, email, etc
- Role data: primary key, description. Examples:
  - ‘Content Editor’
  - ‘Account Administrator’
  - ‘Site Administrator’

Many-To-Many Relationship

- Users can have many roles.
- Roles will have many users
- Create transition table: userrole
  - userid + roleid = composite primary key

userHasRole($role)

- Query database for user roles
- If user is not assigned to the appropriate role, produce error and return FALSE, otherwise return TRUE
- Three tables: user, role, userrole:

```
$sql = “SELECT COUNT(*) FROM user
INNER JOIN userrole ON user.user_id = userrole.user_id
INNER JOIN role ON userrole.role_id = role.role_id
WHERE user.email = :email AND role.role_id = :role_id;
```

userHasRole($role)

$s = $pdo->prepare($sql)…then bind and execute

$row = $s->fetch(); //fetch results of sql

If($row[0] > 0) //if result set is empty then no match

{ return TRUE;} //user has role

else

{return FALSE;} //user doesn’t have role