Chapter 21

How to create secure web sites

Objectives

Applied

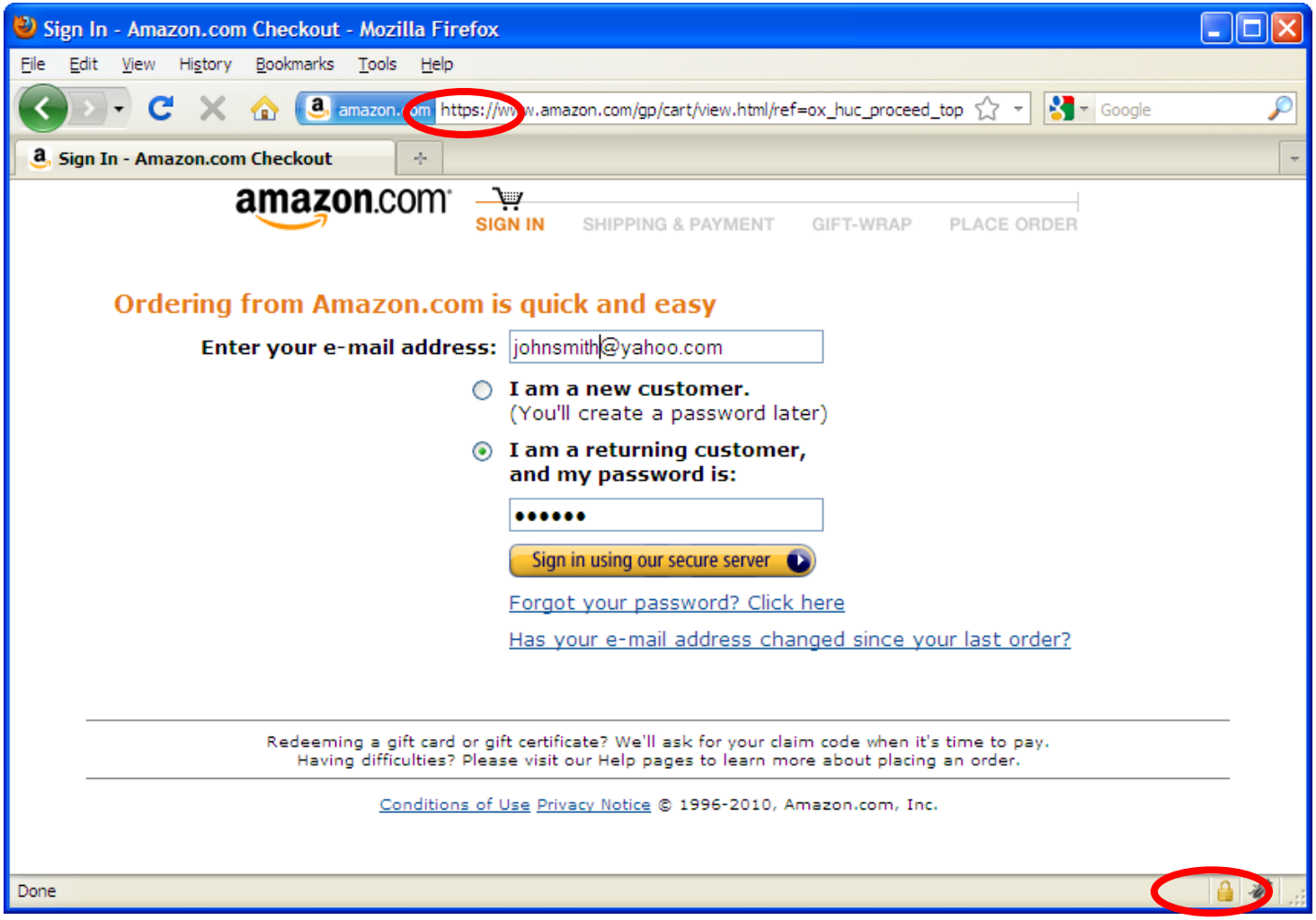
1. Use a secure connection and the Secure Sockets Layer (SSL) protocol for your web pages whenever that's needed.
2. Use form-based authentication for your web pages whenever that's needed.
3. Use PHP to encrypt and decrypt data whenever that's needed.

Objectives (continued)

Knowledge

1. Describe the use of the SSL protocol for getting a secure connection and providing for authentication, including the use of a digital secure certificate, SSL strength, and the $_SERVER array.
2. Distinguish between form-based authentication and basic authentication.
3. Describe the use of PHP for encrypting the data that's stored in a database and for decrypting the data after it's retrieved from the database.

A request made with a secure connection



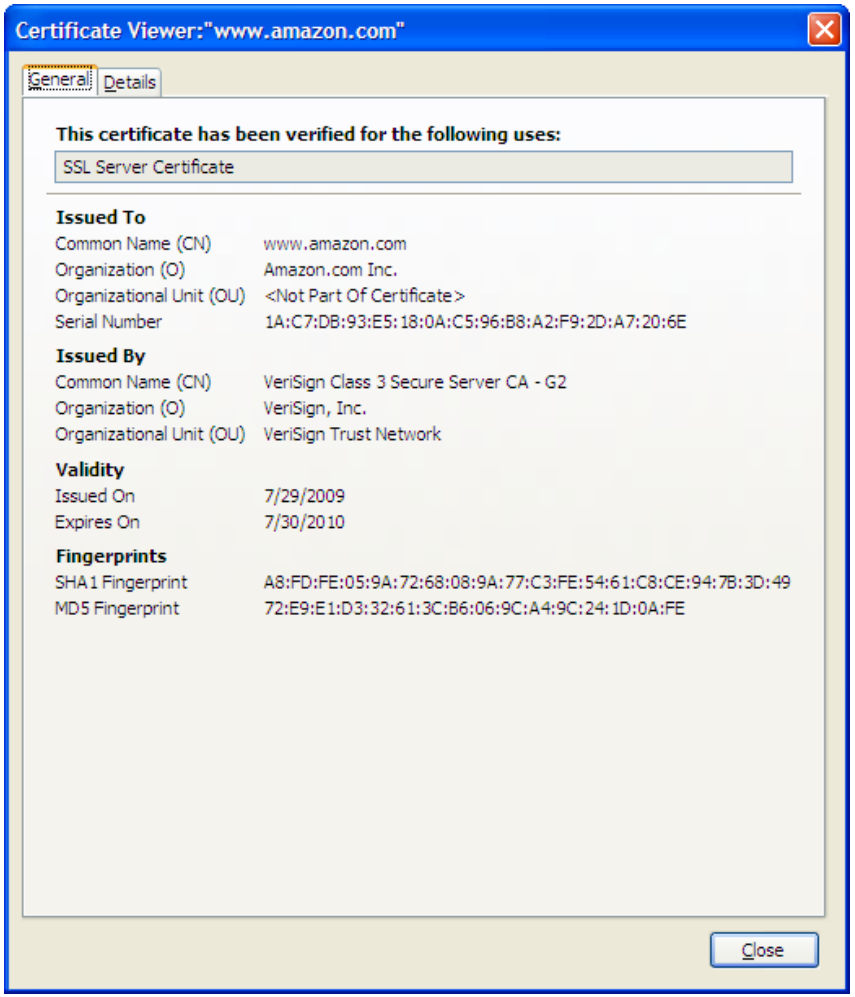
Terms

- Secure Sockets Layer (SSL)
  - An older Internet protocol that allows for data transmission between server and client through a secure connection
- Transport Layer Security (TLS)
  - A newer protocol for transferring data via a secure connection.
  - Often referred to as SSL
- Secure connection
  - The browser encrypts data being sent to the server and the server then decrypts it
  - The server encrypts data being sent to the browser and the browser then decrypts it

Types of digital secure certificates

- Server certificate
- Client certicate

A digital secure certificate



How authentication works

- *Authentication* is the process of determining whether a server or client is who and what it claims to be.
- When a browser makes an initial attempt to communicate with a server over a secure connection, the server authenticates itself by providing a *digital secure certificate*.
- If the digital secure certificate is registered with the browser, the browsers won't display the certificate by default. However, the user still has the option to view the certificate.
- In some rare cases, the server may request that a client authenticate itself by presenting its own digital secure certificate.

Authorities that issue digital secure certificates

www.verisign.com

www.thawte.com

www.geotrust.com

www.instantssl.com

www.entrust.com

SSL strengths

Refers to the length of the generated key that is created during encryption

Stronger security costs more

40-bit

56-bit

128-bit *typical SSL strength for collecting personal information

256-bit

The process

- The person/company desiring a digital secure certificate provides necessary information to a registration authority (RA)
- The RA verifies the information and approves the request.
- The certificate authority (CA) issues the secure certificate
- The certificate is then sent to the web host for installation

URLs for secure connections on a local system

Test if secure connections are configured correctly

```
https://localhost/
```

Request a secure connection

```
https://localhost/book_apps/ch21_ssl/
```

Return to a regular connection

```
http://localhost/book_apps/ch21_ssl/
```

URLs for secure connections over the Internet

Request a secure connection

```
https://www.murach.com/
```

Return to a regular connection

```
http://www.murach.com/
```

The $_SERVER array

Index	Description
HTTPS	Returns a non-empty value if the current request is using HTTPS.
HTTP_HOST	Returns the host for the current request.
REQUEST_URI	Returns the URI (Uniform Resource Identifier) for the current request.

A utility file that redirects to a secure connection

```
<?php
    // make sure the page uses a secure connection
    if (!isset($_SERVER['HTTPS'])) {
        $url = 'https://' .
               $_SERVER['HTTP_HOST'] .
               $_SERVER['REQUEST_URI'];
        header("Location: " . $url);
        exit();
    }
?>
```

Recall from Chapter 5: A PHP function for redirecting a request

```
header($header) //send an HTTP header to the browser
```

The header function

```
header('Location: .');          // the current directory
header('Location: ..');         // up one directory
header('Location: ./admin');    // down one directory
header('Location: error.php');
header('Location: http://www.murach.com/');
```

Form-based authentication

- Allows the developer to code a login form that gets the username and password.
- Allows the developer to only request the username and password once per session.
- By default, it doesn't encrypt the username and password before sending them to the server.

Basic authentication

- Causes the browser to display a dialog box that gets the username and password.
- Requires the browser to send the username and password for every protected page.
- By default, it doesn't encrypt the username and password before sending them to the server.

Digest authentication

- Causes the browser to display a dialog box that gets the username and password.
- Encrypts the username and password before sending them to the server.

Which to use?

- Form-based and basic authentication don’t encrypt information, so they are typically used over a secure connection.
- Although digest authentication encrypts the information, it is not as secure as using a secure connection.

A function that encrypts a string

sha1($string[, $bin])

- **Uses the Secure Hash Algorithm 1 to calculate the hash value for the specified string**
- **Returns a 40-character hexadecimal value**
- **The optional second parameter can be set to TRUE to get raw binary data with a length of 20**

A script that creates a table for usernames and passwords

```
CREATE TABLE administrators (
  adminID             INT              NOT NULL
                                       AUTO_INCREMENT,
  emailAddress        VARCHAR(255)     NOT NULL,
  password            VARCHAR(60)      NOT NULL,
  firstName           VARCHAR(60),
  lastName            VARCHAR(60),
  PRIMARY KEY (adminID)
);

INSERT INTO administrators
   (adminID, emailAddress, password)
VALUES
   (1, 'joelmurach@yahoo.com',
       '446b9db5b3fdd38be64e3a4bde284196f77000df'),
   (2, 'ray@harris.net',
       'ba7294056da6cfb82cabe5f85a31eed548979611'),
   (3, 'mike@murach.com',
       '3f2975c819cefc686282456aeae3a137bf896ee8');
```

PHP for storing and validating passwords

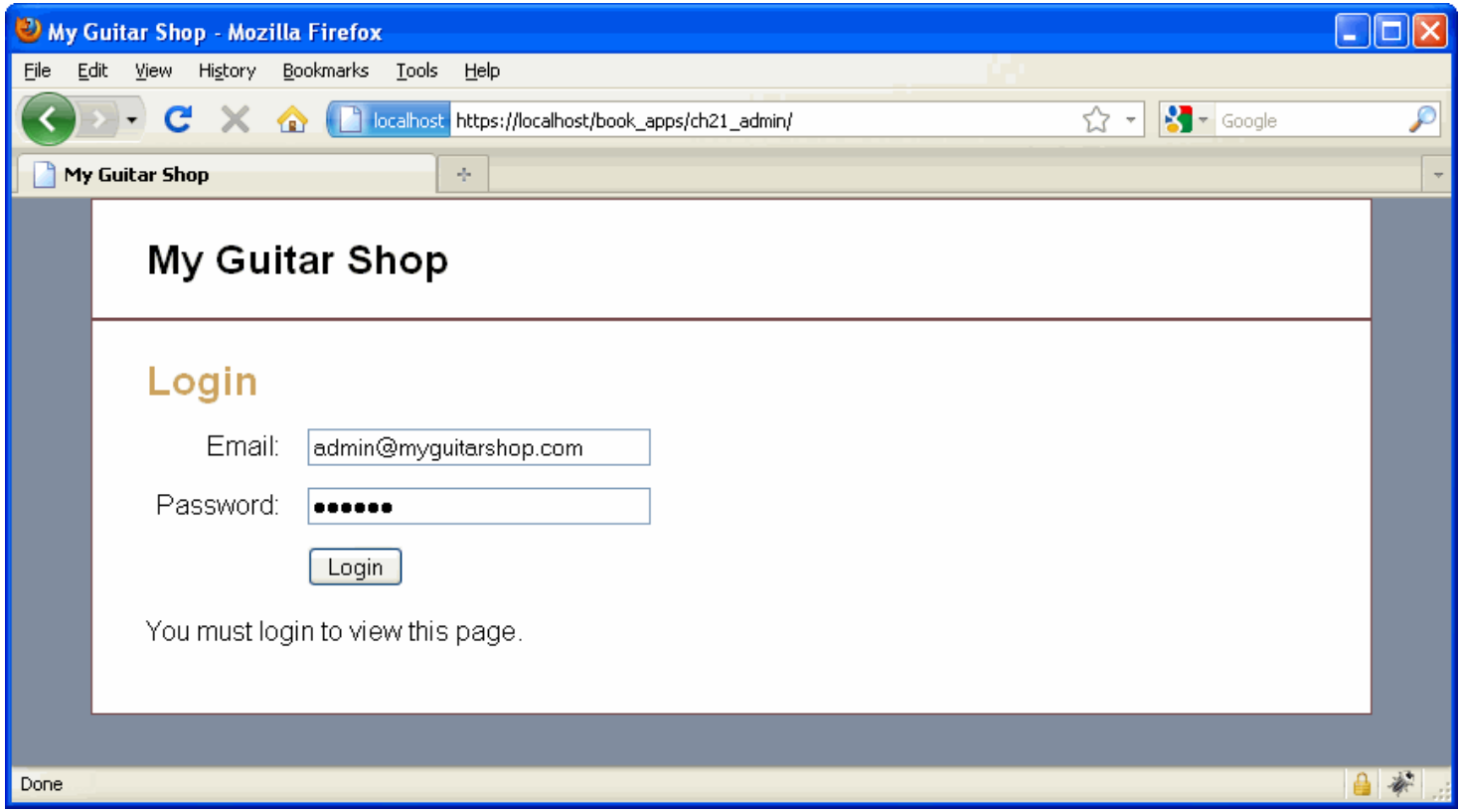
```
<?php
function add_admin($email, $password) {
    global $db;
    $password = sha1($email . $password);
    $query = 'INSERT INTO administrators
                        (emailAddress, password)
              VALUES
                        (:email, :password)';
    $statement = $db->prepare($query);
    $statement->bindValue(':email', $email);
    $statement->bindValue(':password', $password);
    $statement->execute();
    $statement->closeCursor();
}
```

Storing and validating passwords (continued)

```
function is_valid_admin_login($email, $password) {
    global $db;
    $password = sha1($email . $password);
    $query = 'SELECT adminID
              FROM administrators
              WHERE emailAddress = :email
                AND password = :password';
    $statement = $db->prepare($query);
    $statement->bindValue(':email', $email);
    $statement->bindValue(':password', $password);
    $statement->execute();
    $valid = ($statement->rowCount() == 1);
    $statement->closeCursor();
    return $valid;
}
?>
```

A login form for form-based authentication

- Uses HTML form text boxes for email and password
- Using a secure connection, the username and password are sent to the server



The controller for the protected pages

```
<?php
// Start session management and include necessary functions
session_start();
require_once('model/database.php');
require_once('model/admin_db.php');

// Get the action to perform
if (isset($_POST['action'])) {
    $action = $_POST['action'];
} else if (isset($_GET['action'])) {
    $action = $_GET['action'];
} else {
    $action = 'show_admin_menu';
}

// If the user isn't logged in, force the user to login
if (!isset($_SESSION['is_valid_admin'])) {
    $action = 'login';
}
```

The controller for the protected pages (cont.)

```
// Perform the specified action
switch($action) {
    case 'login':
        $email = $_POST['email'];
        $password = $_POST['password'];
        if (is_valid_admin_login($email, $password)) {
            $_SESSION['is_valid_admin'] = true;
            include('view/admin_menu.php');
        } else {
            $login_message =
                'You must login to view this page.';
            include('view/login.php');
        }
        break;
```

The controller for the protected pages (cont.)

```
    case 'show_admin_menu':
        include('view/admin_menu.php');
        break;
    case 'show_product_manager':
        include('view/product_manager.php');
        break;
    case 'show_order_manager':
        include('view/order_manager.php');
        break;
    case 'logout':
        $_SESSION = array();   // Clear all session data
        session_destroy();     // Clean up the session ID
        $login_message = 'You have been logged out.';
        include('view/login.php');
        break;
}
?>
```

A utility file that forces a valid admin user

```
<?php
    // make sure user is a valid administrator
    if (!isset($_SESSION['is_valid_admin'])) {
        header("Location: ." );
    }
?>
```

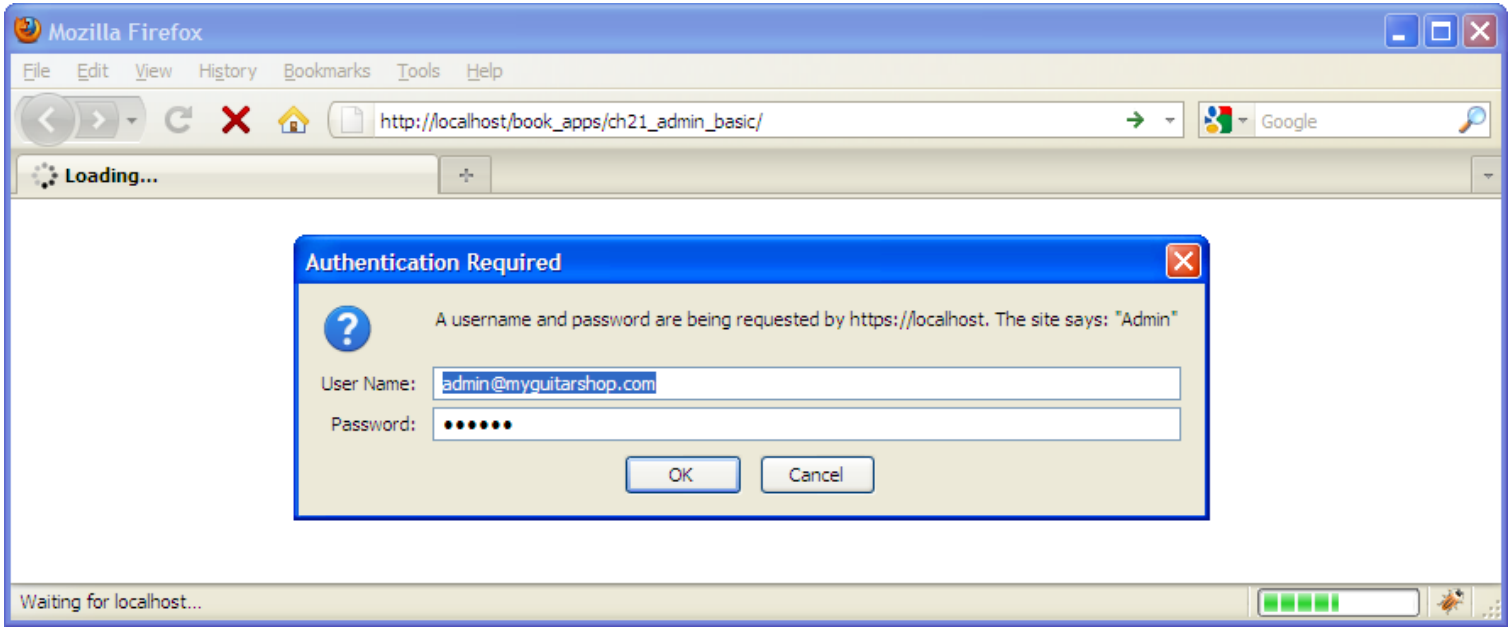
Code at the top of the login page

```
<?php
    // require a secure connection
    require_once('util/secure_conn.php');
?>
```

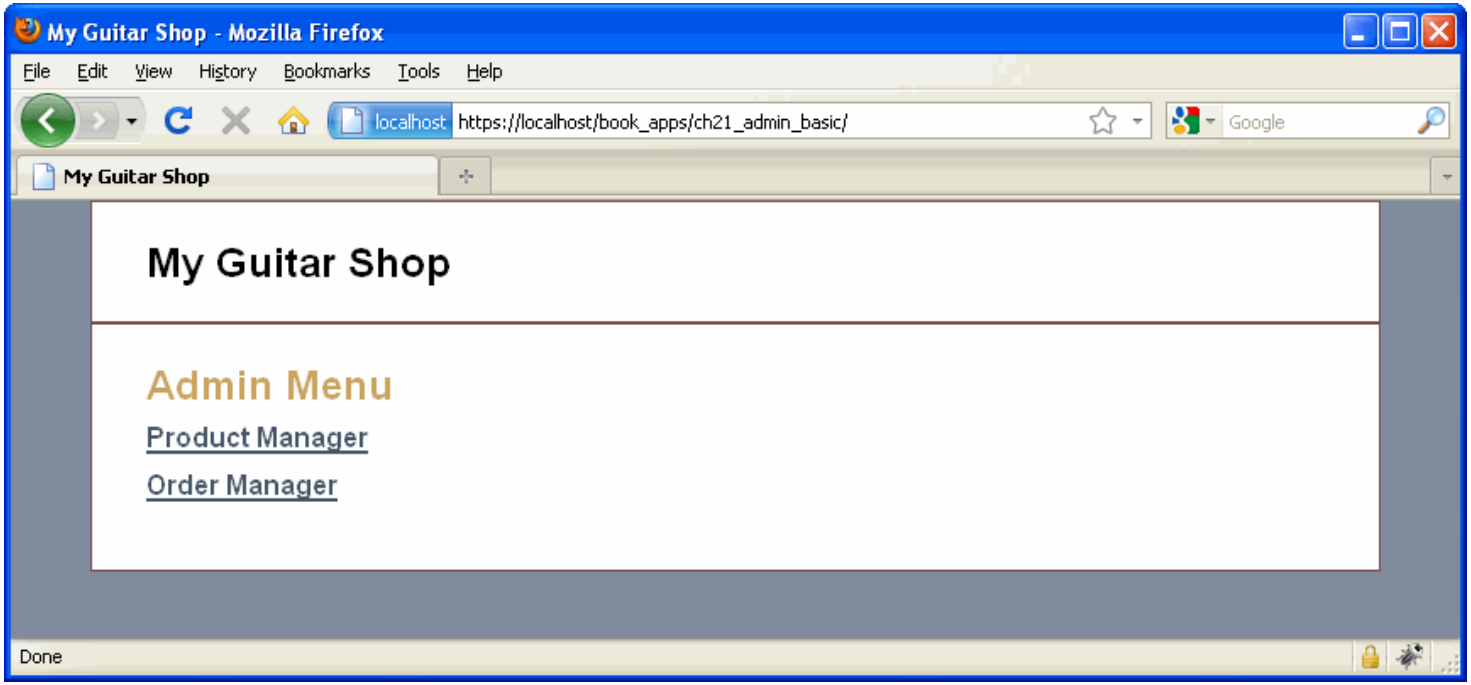
Code the top of the other protected pages

```
<?php
    // require a secure connection
    require_once('util/secure_conn.php');
    // require a valid admin user
    require_once('util/valid_admin.php');
?>
```

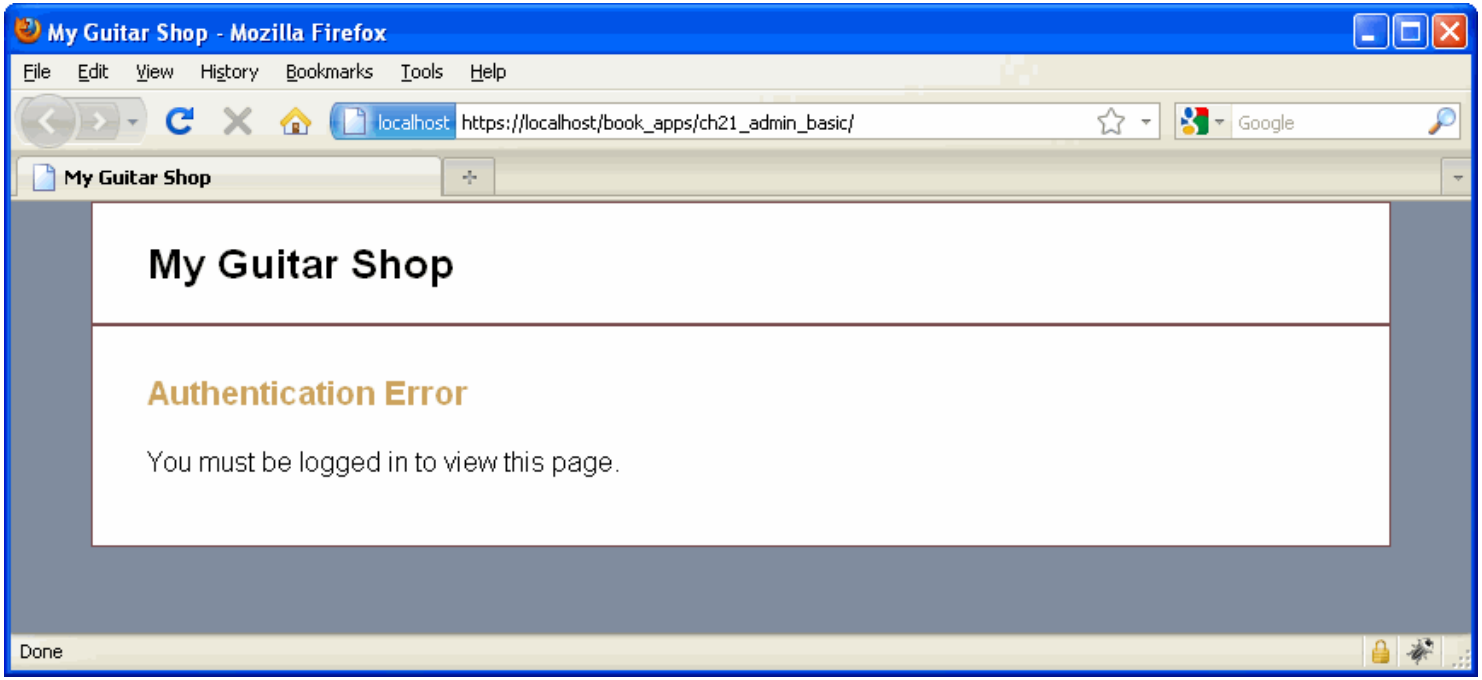
A login dialog box for basic authentication



A protected page



The unauthorized page



The $_SERVER array for basic authentication

Index	Description
PHP_AUTH_USER	The username from the authentication dialog box or a NULL value if the dialog box hasn't been displayed.
PHP_AUTH_PW	The password from the authentication dialog box or a NULL value if the dialog box hasn't been displayed.

PHP that forces a valid admin user

```
<?php
require_once('model/database.php');
require_once('model/admin_db.php');

$email = $_SERVER['PHP_AUTH_USER'];
$password = $_SERVER['PHP_AUTH_PW'];
if (!is_valid_admin_login($email, $password)) {
    header('WWW-Authenticate: Basic realm="Admin"');
    header('HTTP/1.0 401 Unauthorized');
    include('unauthorized.php');
    exit();
}
?>
```

PHP at the top of each protected page

```
<?php
    // require a secure connection
    require_once('util/secure_conn.php');

    // require a valid admin user
    require_once('util/valid_admin.php');
?>
```

Encryption Vocabulary and Libraries

php includes a library called mcrypt which includes several functions and constants for encryption and decryption

Cipher: An algorithm used for encryption

One standard is the Rijndael cipher which operates on fixed-length blocks.

To control how the cipher operates on a block, a mode of operation is chosen.

An optional initialization vector (IV) allows the cipher to produce a unique stream.

Code that encrypts and decrypts data

```
$credit_card_no = '4111111111111111';

// Set up the variables
$cipher = MCRYPT_RIJNDAEL_128; //128-bit key size
$mode = MCRYPT_MODE_CBC; //CBC mode of operation

// use the sha1 function to generate a key
$key = sha1('secrectKey', true);
$ivs = mcrypt_get_iv_size($cipher, $mode); //get size of iv
$iv = mcrypt_create_iv($ivs); //create initialization vector

// Encrypt the data
$data = mcrypt_encrypt($cipher,
        $key, $credit_card_no, $mode, $iv);
$data = base64_encode($data);

// Decrypt the data
$data = base64_decode($data);
$credit_card_no  = mcrypt_decrypt($cipher,
                   $key, $data, $mode, $iv);
echo 'Decrypted data: ' . $credit_card_no . '<br />';
```

The Crypt class (crypt.php)

```
<?php
class Crypt {
    private $key;
    private $ivs;
    private $iv;
    private $cipher;
    private $mode;

    public function __construct() {
        $this->cipher = MCRYPT_RIJNDAEL_128;
        $this->mode = MCRYPT_MODE_CBC;
        $this->key = sha1('secrectKey', true);
        $this->ivs =
            mcrypt_get_iv_size($this->cipher,
                               $this->mode);
        $this->iv = mcrypt_create_iv($this->ivs);
    }
```

The Crypt class (continued)

```
    public function encrypt($data) {
        $data = mcrypt_encrypt($this->cipher,
                    $this->key, $data,
                    $this->mode, $this->iv);
        $data = base64_encode($data);
        return $data;
    }

    public function decrypt($data) {
        $data = base64_decode($data);
        $data = mcrypt_decrypt($this->cipher,
                    $this->key, $data,
                    $this->mode, $this->iv);
        return $data;
    }
}
?>
```

Code that uses the Crypt class

```
require 'crypt.php';

$credit_card_no = '4111111111111111';

// Create the Crypt object
$crypt = new Crypt();

// Use the Crypt object to encrypt the data
$data = $crypt->encrypt($credit_card_no);
echo 'Encrypted data: ' . $data . '<br />';

// Use the Crypt object to decrypt the data
$credit_card_no = $crypt->decrypt($data);
echo 'Decrypted data: ' . $credit_card_no . '<br />';
```