Web プログラミング（１）

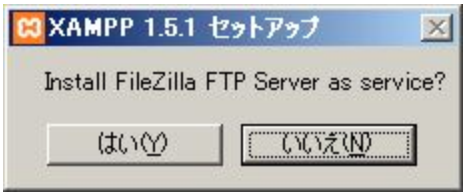
1. 講義の目標
   実際のアプリケーションの作成を通じて、プログラミング・データベース・ウェブの開発技術を身につける。
2. ツールのインストール
   開発ツール：XAMPP （Apache, MySQL, PHP, Perl　のパッケージ・GPL）

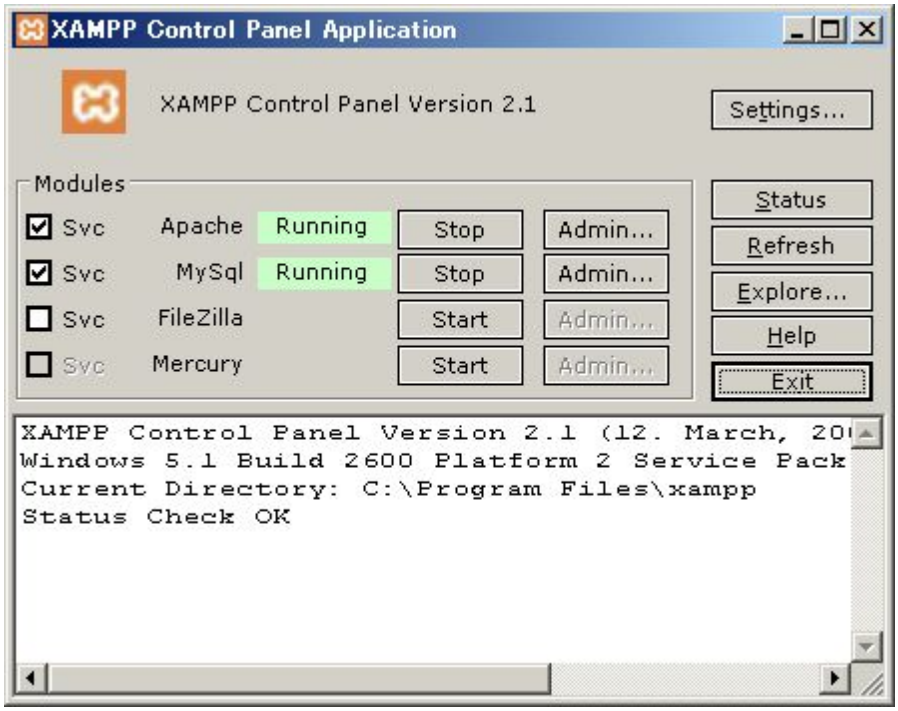
   ダウンロードとインストール：
   　　オリジナルサイト　http://www.apachefriends.org/
   　　学内ミラー　　http://iis.edu.tama.ac.jp/webpg/

   xampp-win32-1.5.1-installer.exe　をダウンロードして実行

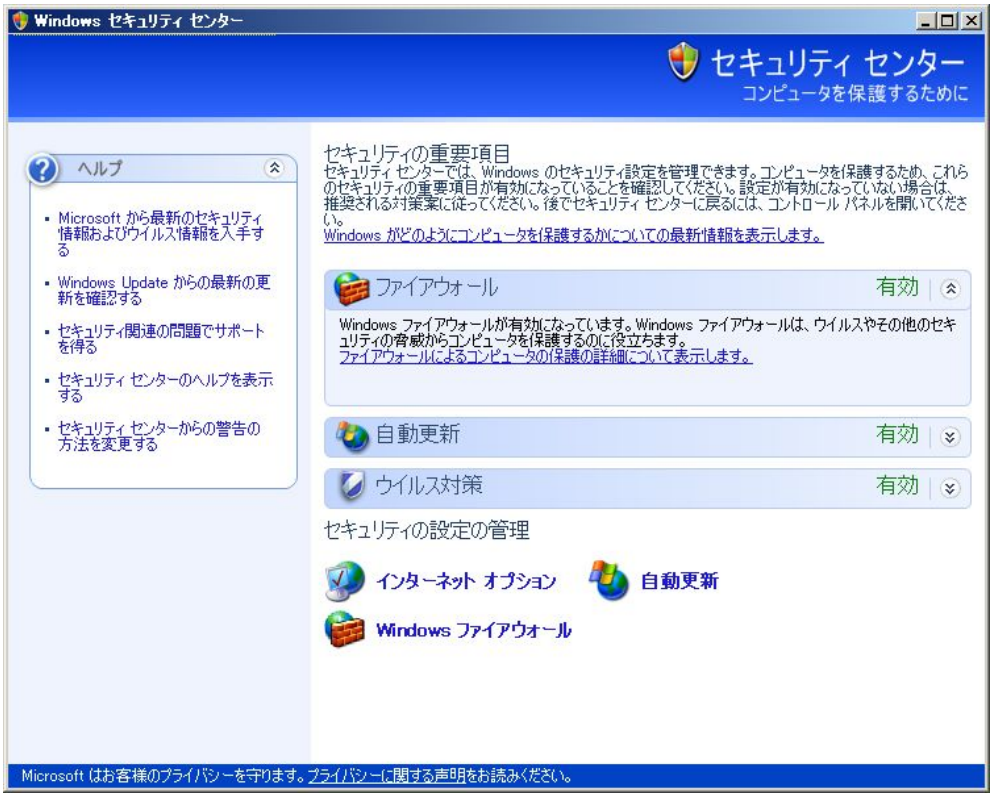
Apache, MySQL を service としてインストールすると、login していなくても起動する

FTP サーバは今回は不要（外からファイルを転送することはない）



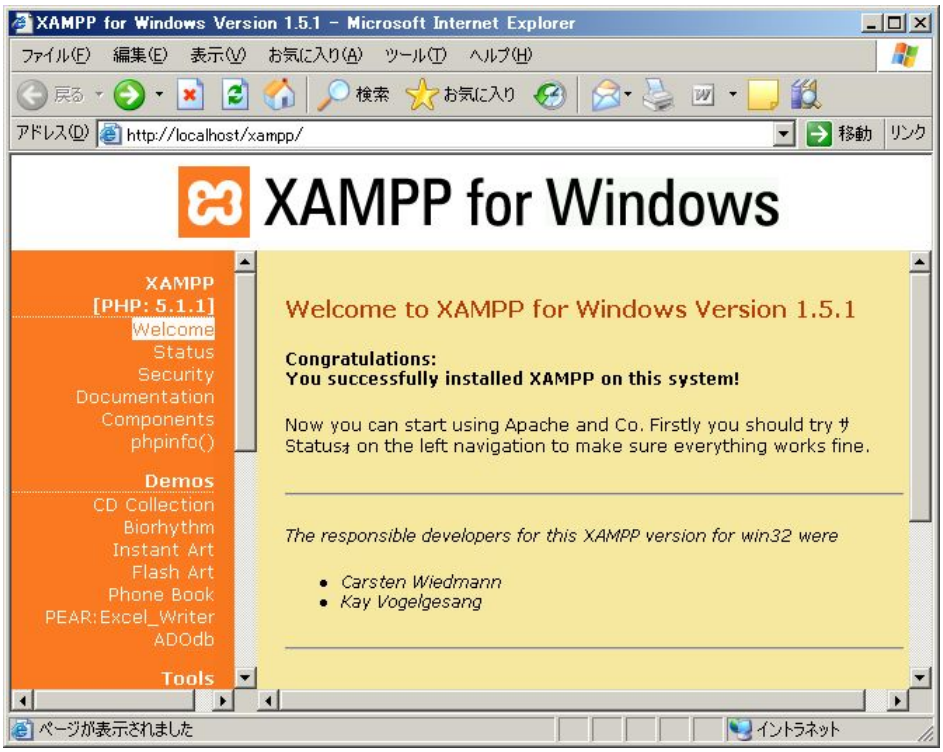


3. ファイアウォール設定
   ファイアウォールでブロックされた場合は：ポート 80, 443, 3306 を開く
   1. ブロック時に警告が出る設定の場合は、ブロック対象が xampp/apache/bin/apache と xampp/mysql/bin/mysql であること、ポート番号が一致することを確認して許可
   2. 警告が出ないが動作しない場合は、セキュリティセンターで許可
      スタート → 設定 → コントロールパネル → セキュリティセンター → セキュリティの設定の管理：Windows ファイアウォール



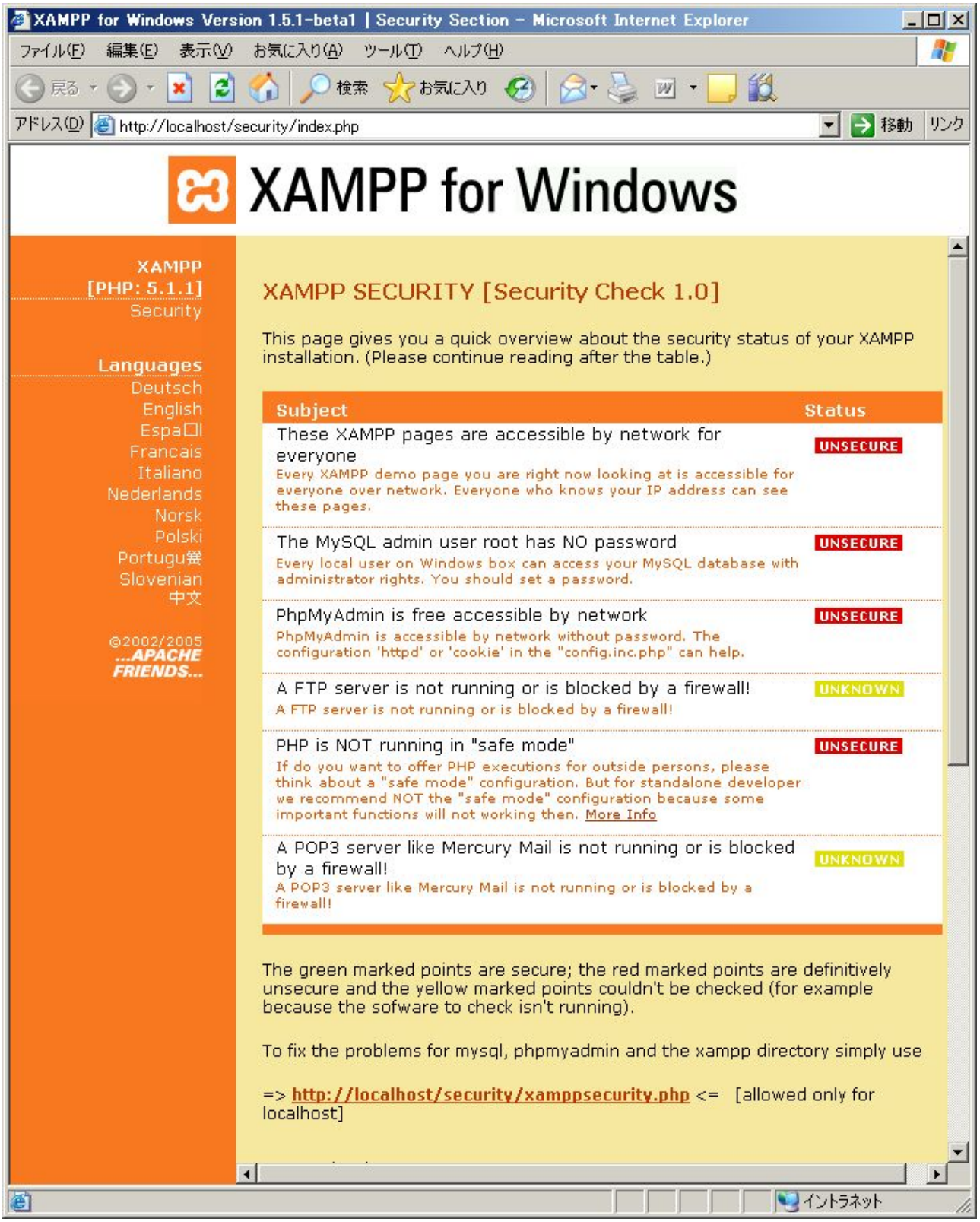
→　例外　→　ポートの追加　→ 80, 443, 3306 を追加

4. 動作テスト
XAMPP コントロールパネルから、apache の Admin... を選択

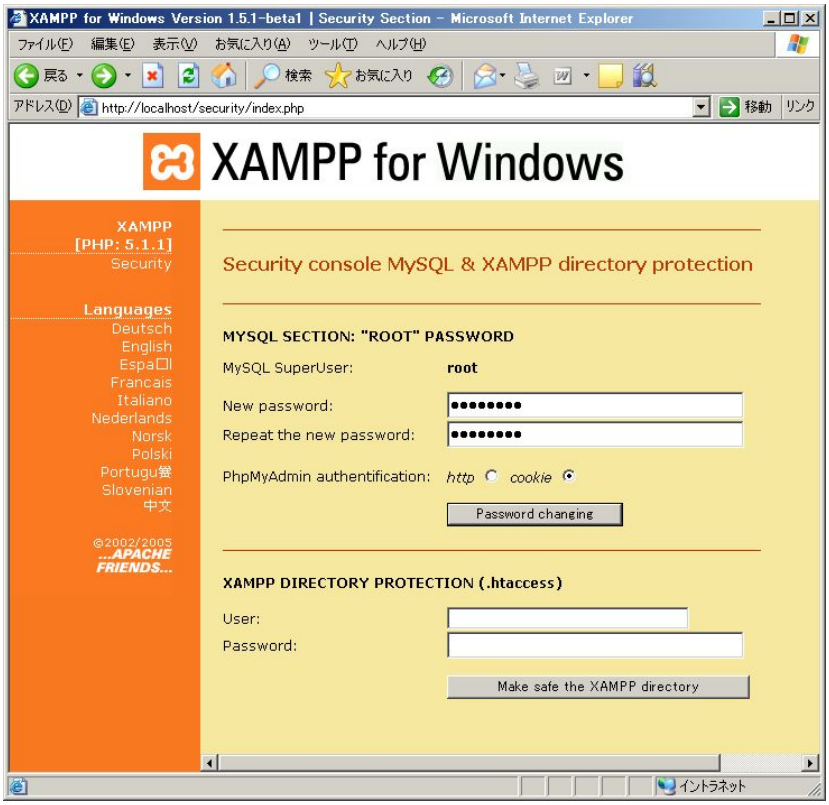


5. セキュリティ設定

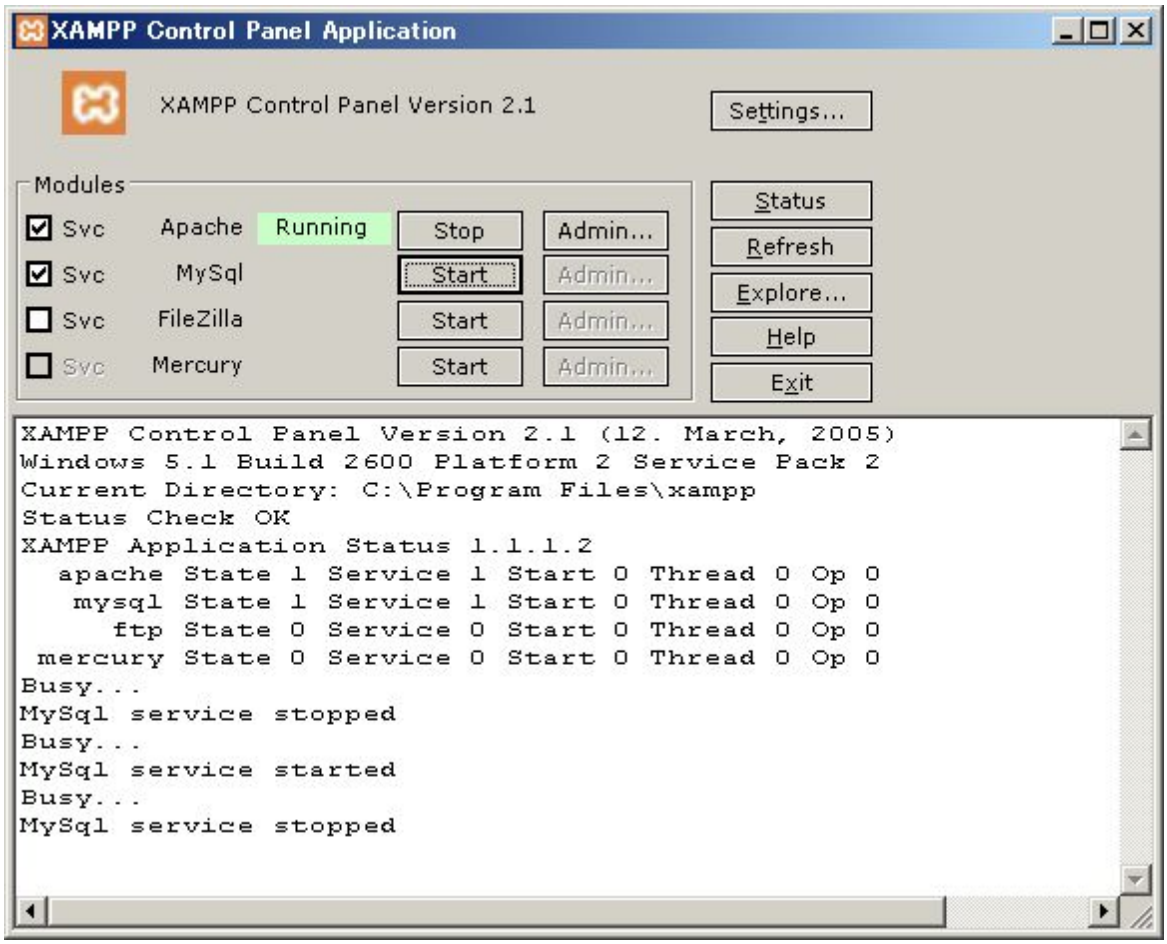
スタートページ左の Security を開く



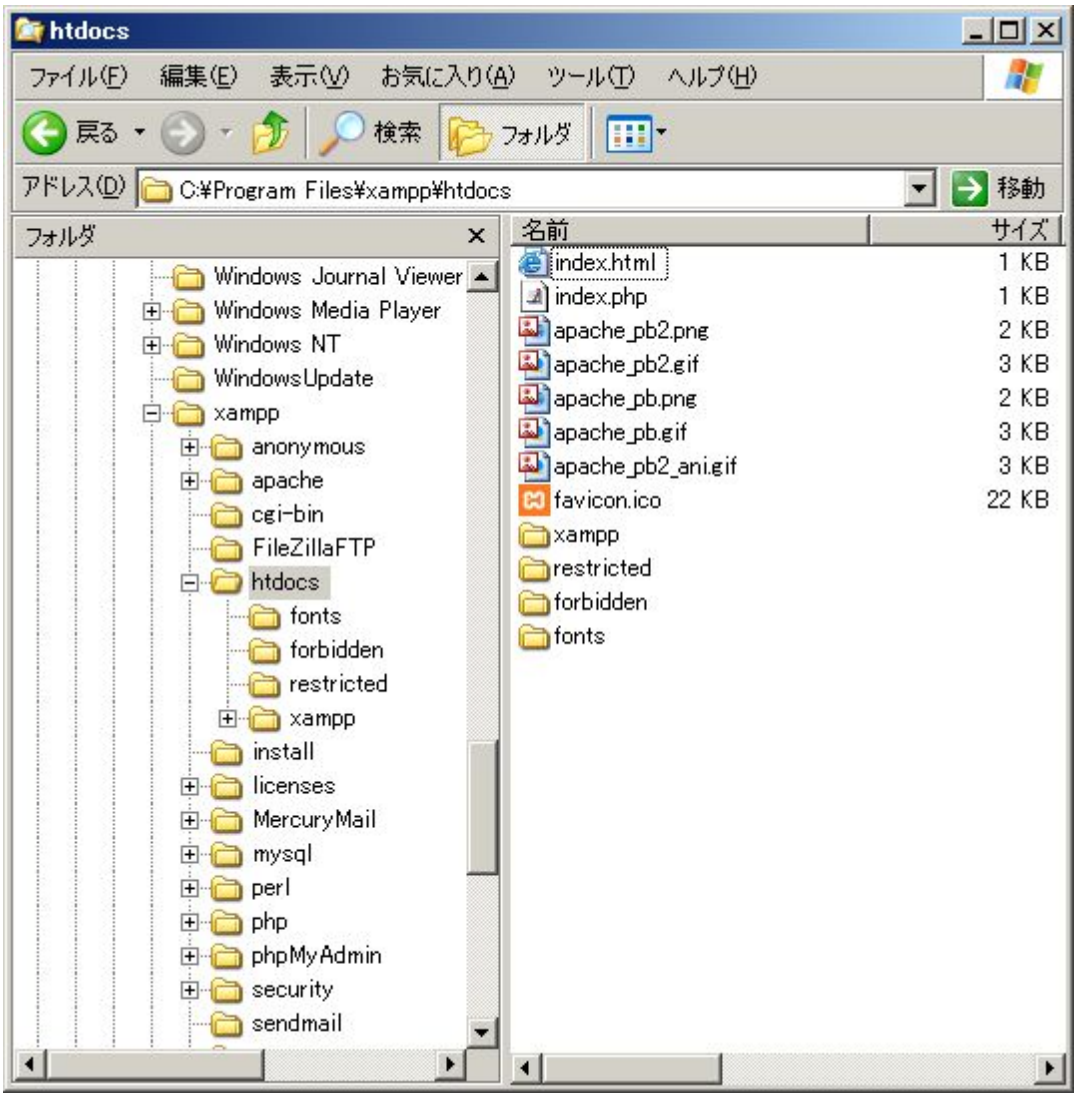
UNSECUREと表示されているところがセキュリティ上問題のある点。（正しい英語はINSECURE）ただし、PHP の “safe mode” はプログラム開発時は設定しないほうがよい。画面下の xamppsecurity.php のリンクをたどることで、上3つのセキュリティが改善する。



MySQL の管理者 root パスワードの設定（任意） → [Password Changing]
XAMPP ディレクトリの認証ユーザ名とパスワード（任意） → [Make safe ...]
xampp コントロールパネルから、MySQL の停止と起動



6. テストページの作成
   xampp コントロールパネルから Explore をクリック　→ xampp/htdocs フォルダ内にファイル（やフォルダ）を作成　→　ブラウザから確認　（ファイル名を省略したときは、index.php, index.html, index.htm の順に検索される）
   また、スタート→プログラム→apachefriends→xampp→xampp httpdoc folder からも公開フォルダを開くことができる。



7. “test” フォルダを作成し、index.html を記述して http://localhost/test/ で表示できることを確認しよう。余裕のある人は、index.php を記述し、http://localhost/test/ で index.php が優先されること・PHPが実行できることを確認しよう。

   index.php の例

```
<?
        phpinfo( );
?>
```

Web プログラミング（2）

1. HTML の基本（http://www.w3.org/TR/html4/ などを参照）
HTML(Hyper Text Markup Language) では、タグによって文字に属性を与える。基本的な構造は以下のとおり。

```
<HTML>
    <HEAD>
        <TITLE>ページタイトル</TITLE>（などのページの情報）
    </HEAD>
    <BODY>
        （本文）
    </BODY>
</HTML>
```

<BODY> 中でよく使用されるタグ：

```
<H1><H2><H3><H4><H5><H6>        見出しレベル
<TABLE>                         表
<P>                             段落
<BR>                                改行（終了タグ無し）
<A HREF="url">                  ハイパーリンク
```

<TABLE> 中の構成：

```
<TABLE>
    <CAPTION>表のタイトル（省略可能）</CAPTION>
    <TR>一行分データ</TR>        この繰り返し
</TABLE>
```

<TR> 中の構成：

```
<TR>
    <TH>見出し</TH> または <TD>一項目</TD> の繰り返し
</TR>
```

例： table.html

```
<BODY>
    <H1>HTML テスト</H1>
    <H2>テーブルテスト</H2>
    <TABLE>
        <CAPTION>テスト</CAPTION>
        <TR><TH>学籍番号</TH><TH>氏名</TH></TR>
        <TR><TD>29988777</TD><TD>多摩太郎</TD></TR>
    </TABLE>
</BODY>
```

2. PHP の基本(http://www.php.net/manual/ja/ などを参照)

3. HTML 中に PHP スクリプトを埋め込む。

```
<?php
1. PHP スクリプト
?>
```

または

```
<SCRIPT LANGUAGE="php">
    PHP スクリプト
</SCRIPT>
```

一つのファイル中に何度も PHP スクリプト部分が出現してよい。スクリプト中では、大文字･小文字は区別する。

コメント：

```
// 行末までコメント
/*
    コメントブロック
*/
```

基本型：

integer, float, boolean, string
array, object, resource, NULL

変数名：

$ の後に英文字＋（任意長の英数文字の繰り返し）
（英文字にはアンダースコアを含む）
PHP では、変数宣言なしに変数を使用できる。変数がすでに宣言されているかどうかを調べるには、関数 isset（変数名） を用いる。定義済み変数を未定義にするには、命令 unset(変数名） を使用する。

```
if( ! isset($newvar) )
echo '$newvar は定義されていません';
```

文字列：

シングルクォーテーションで囲った文字列は、ほぼそのまま評価される。ダブルクォーテーションで囲った文字列は、C に類似したエスケープを行う。また、ダブルクォーテーション内の変数は展開される。文字列の接続には、ピリオドを使う。

```
$var = 'test';
echo ' $var \n';    // $var \n
echo " $var \n";   // test 改行
echo $var . '000';// test000
```

型の自動変換：

型は代入･参照時に自動的に変換される。

```
$x = '10';
$y = $x + 10;     // 20
```

配列：

PHPの配列は、「キー（インデックス）」と「値」のペアが保存される順序つきデータベースのようなものである（連想配列）。キーには整数のほかに文字列も指定できる。キーを指定しない場合は、自動的に指定される（最初は 0 から）。

```
$b['one'] = 'alpha';
$b['two'] = 'beta';
echo $b['one'];    // alpha

$a = array('alpha','beta','G' => 'gamma');
echo $a[0];        // alpha
echo $a['G'];      // gamma
```

制御構造：

if, switch, for, while, do-while

C とほぼ同様。switch の判定に文字列を使用できる。

foreach

配列の各要素の値とキーの組を取り出し操作する。キーは省略できる。

```
foreach( 配列名 as キー => 値 )
foreach( 配列名 as 値 )
```

例1：

```
<html>
<body>
<?php
for($i=1; $i<=9; $i++)
{
      for($j=1; $j<=9; $j++)
      {
            echo $i * $j;
            echo " ";
      }
      echo "<br>";
}
?>
</body>
</html>
```

例2：

```
<html>
<body>
<?php
$a = array(
	'apple' => 'red',
	'lemon' => 'yellow',
	'orange' => 'orange'
);
foreach($a as $v)
{
	echo $v;
	echo "<br>";
}
?>
</body>
</html>
```

例3：

```
<html>
<body>
<?php
$a = array(
	'apple' => 'red',
	'lemon' => 'yellow',
	'orange' => 'orange'
);
foreach($a as $k => $v)
{
	echo "$k is $v.";
	echo "<br>";
}
?>
</body>
</html>
```

演習：

例3の表示を <table> タグを用いて整形しなさい。スタイルシートを用いるか、<table border> によって、罫線を表示すること。

Web プログラミング（３）

1. HTML からの変数渡し

<FORM> 中に入力用コントロールエレメントを記述する：
<FORM ACTION="*uri*" METHOD="*method*">
…
</FORM>
uri： 入力データの送信先。相対指定するのが普通
method： GET または POST

<FORM> 中の入力コントロール
<INPUT TYPE=*type* NAME=*name* VALUE=*value*>
他に、<TEXTAREA> <SELECT> <BUTTON>
VALUE は省略可能
よく使われる type
text|password|radio|checkbox|hidden|submit|reset
ボタン以外のコントロールの name と value のペアが送信される

例： form.html

```
<BODY>
	<H1>FORM テスト</H1>
	<FORM ACTION="get.php" METHOD="GET">
	<TABLE>
	<TR><TH>学籍番号</TH><TD><INPUT TYPE="TEXT"
NAME="nr"></TD></TR>
	<TR><TH>　氏名　</TH><TD><INPUT TYPE="TEXT"
NAME="name"></TD></TR>
	<TR><TH>　性別　</TH>
		<TD>
			<INPUT TYPE="radio" NAME="mf" value="M"> 男
<br>
			<INPUT TYPE="radio" NAME="mf" value="F"> 女
		</TD>
	</TR>
	<TR><TH>Password</TH><TD><INPUT TYPE="PASSWORD"
NAME="pw"></TD></TR>
	</TABLE>
	<INPUT TYPE="submit" VALUE="送信する">
	</FORM>
</BODY>
```

2. PHP での変数の受け取り
フォームデータを送信された PHP ファイルでは、メソッドに応じて $_GET, $_POST 配列にデータが格納されている。各配列のキーが name に、値が value に相当する。isset 関数に複数の引数を渡すことで「すべての引数が定義されているかどうか」をチェックすることができる。フォームデータから送信される GET/POST 配列には、「空文字列」が入っているので、「文字列長」でチェックする。
例： get.php

```
<html>
<body>
<?php
if( !isset($_GET['name'],$_GET['nr'],$_GET['mf'],$_GET['pw']) ||
        strlen($_GET['name']) == 0 ||
        strlen($_GET['nr']) == 0 ||
        strlen($_GET['mf']) == 0 ||
        strlen($_GET['pw']) == 0
)
{
        echo 'Missing Data';
}
else
{
        foreach($_GET as $k => $v)
        {
                echo "$k is $v.";
                echo "<br>";
        }
        }
?>
</body>
</html>
```

3. 開発システム要件

最終課題として、以下のような高機能掲示板を作成する。

- ユーザ登録ができ、登録ユーザだけがパスワード認証を経て書き込みできる。（オプショナル：ただし、ユーザ登録は有効な tama.ac.jp ドメインのメールアドレスを持つ人だけに限定する。）
- パスワード認証後は、明示的にログアウトするかブラウザを終了するまでログイン状態を保つ。
- 書き込みは、ツリー構造をとる。根は複数あってよいが、簡単にするためには表示されない「０番目の書き込み」を仮定してこれを唯一の根とするとよい。（オプショナル：書き込みは、ネットワーク構造をとる。多対多の対応になることに注意）
- ユーザの書き込みに対して、他のユーザが「評価」を与えられる。あるユーザは、ある書き込みに対して一度しか評価できず、評価を変更することはできない。
- ユーザは自分に関する情報の変更ができる。
- 平均評価の高いユーザ順のリストを表示できる。
- （オプショナル：新しい書き込みがあるとメールで通知する）
- （オプショナル：データベース上に画像を登録できる）
- （オプショナル：重要そうな書き込みを自動的に検出できる）

演習：

高機能掲示板に必要なデータベース構造を検討し、レポートしなさい。

Web プログラミング（４）

1. MySQL サーバとの接続

詳細は、
http://jp.php.net/mysql/ PHP:MySQL 関数
http://dev.mysql.com/usingmysql/php/ Using MySQL with PHP
を参照のこと。

基本構造：
```
$con = mysql_connect(サーバ名, ユーザ名, パスワード);
mysql_select_db(データベース名);
データベース処理
mysql_close($con);
```

エラー処理：
mysql_connect は、接続に失敗すると false を返す。

```
$con = mysql_connect(サーバ名, ユーザ名, パスワード);
if( $con == false )
    die('Cannot connect.');
```

または

```
$con = mysql_connect(サーバ名, ユーザ名, パスワード)
    or die('Cannot connect');
```

・or は代入演算子より優先順位が低い。
・smart evaluation によって、or の左が真なら右は評価されない。

mysql_select_db も同様に、選択に失敗すると false を返す。
```
mysql_select_db(データベース名) or die('Could not select');
```

例： ctest.php （サンプルとして添付されている cdcol データベースに接続）
```
<html>
<body>
<?php
$con = mysql_connect('localhost', 'root', 'password')
    or die('Cannot connect');
mysql_select_db('cdcol') or die('Cannot select');
echo 'Connect and Select success!';
mysql_close($con);
?>
</body>
</html>
```

2. MySQL サーバでの SQL 文の実行

基本構造：
```
mysql_query(SQL 文);          //  結果集合が返らない SQL
$rs = mysql_query(SQL 文);    //  結果集合が返る SQL
```

・SQL 文には最後の ";" を含めない。
・いずれもクエリに失敗すると false を返す。
・結果集合が返らない SQL に成功すると true が返る。

インジェクション対策：
SQL 文の文字列を生成する際に、文字列結合で生成せず、sprintf - mysql_real_escape_string の組み合わせを使用する。

例： itest.php （サンプルとして添付されている cdcol データベースにデータを挿入）

```
<html>
<body>
<?php
	$con = mysql_connect('localhost', 'root', 'password')
		or die('Cannot connect');
	mysql_select_db('cdcol') or die('Cannot select');

	$query = "insert into cds (titel, interpret, jahr)";
	$query .= sprintf("values ('%s','%s','%s')",
		mysql_real_escape_string("New Book"),
		mysql_real_escape_string("author"),
		mysql_real_escape_string("1998")
	);

	mysql_query($query);

	if ( !$result ) {
		$mes  = 'Error : ' . mysql_error() . "<br>\n";
		$mes .= 'Whole query: ' . $query;
		die($mes);
	}

	mysql_close($con);
?>
</body>
</html>
```

3. 結果集合の処理

基本構造：

```
$rs = mysql_query(SQL 文);
while( $data = mysql_fetch_assoc($rs) )
{
        個別のレコードに対する処理
}
```

・mysql_get_assoc は、結果集合の注目行を連想配列に返し、次の行に移動する
・最後まで読み出すと、false を返す

または（１レコードだけの答えが返ってくるとわかっていれば）

```
$rs = mysql_query(SQL 文);
$data = mysql_fetch_assoc($rs);
該当レコードに対する処理
```

例： stest.php （サンプルとして添付されている cdcol データベースのデータを表示）

```
<html>
<body>
<?php
        $con = mysql_connect('localhost', 'root', 'password')
                or die('Cannot connect');
        mysql_select_db('cdcol') or die('Cannot select');

        $query = "select * from cds";
        $rs = mysql_query($query);
        if ( !$rs  ) {
                $mes  = 'Error : ' . mysql_error();
                die($mes);
        }
        if( mysql_num_rows($rs) == 0)
                die("No data");

        while( $data = mysql_fetch_assoc($rs) )
        {
                echo "{$data['titel']}, {$data['interpret']}, {$data['jahr']}";
                echo "<br>\n";
        }

        mysql_close($con);
?>
</body>
</html>
```

演習：

フォームを使用して、CD データベース cdcol へのデータの追加と一覧を行えるページを作成しなさい。

演習：

最終課題の高機能掲示板に使用するデータベースの構造を決定しなさい。また、MySQL 上で新たなデータベースを作成し、ユーザリストのテーブルを作成しなさい。

4　セッション管理

HTTP は、セッションレスな接続である。このため、連続したアクセスであっても、それぞれ独立な接続として扱われ、情報の受け渡しができない。ページ間で情報を受け渡したいとき、次のような方法がある。

(1)form 内に hidden 属性の input フィールドを作成し、値を代入しておく

```
<form action='test.php' method='get'>
    <input type='hidden' name='id' value='20411999'>
    <input type='text' ....
```

(2)クライアント側に cookie を送り、その中に値を代入しておく

(3)クライアント側に cookie を送り、固有のセッション番号を記録しておく（セッションクッキー）。セッション番号に対応する値を、サーバ側に記録する。

フォーム内に hidden/input を作成する手法や、cookie に値を代入する手法では、ユーザがその値を直接参照したり、変更して返送したりできるため、セキュリティが低い。

PHP では、セッションクッキーの管理が容易になるような仕組みが用意されている。

セッション管理の方法

1. session_start() でセッションを開始する。
1. セッションクッキーが存在しなければ、新しくセッションクッキーが作成され、$_SESSION 配列が初期化される。
2. セッションクッキーがすでに存在すれば、それまでに記録された変数が $_SESSION[(インデックス名)] で参照できるようになる。
2. $_SESSION 配列を書き換えた場合は、session_commit() でセッションを閉じる。このタイミングで、サーバ側にデータが記録される。
3. セッションを終了する場合（ログアウトなど）、session_destroy() でセッションクッキーを削除する。セッションクッキーは、ブラウザを終了させても削除される。念のため、セッションクッキーを削除する前に、$_SESSION 配列を空にしておくとよい。

注意点

1. session_start() は、cookie のやりとりを行うので、HTTP ヘッダを送る前に呼ばなければならない。UTF-8 エンコーディングのファイルには、BOM と呼ばれるデータが付加されることがあり、HTTP ヘッダが送出される場合があるので注意。（EmEditor では「名前を付けて保存」時に BOM の有無を選択できる）
2. session_start() すると、そのあと session_commit() を呼んでも、$_SESSION 変数を読み出すことはできる。session_commit() を呼んだ後は、$_SESSION 変数を変更しても記録されない。
3. session_start() から session_commit() までの時間が長いと、アクセスが集中したときにサーバが落ちる。$_SESSION 変数を読み出すだけでよい場合は、session_start() 後、直ちに session_commit() するほうがよい。

演習：

1. セッション管理を次のサンプルで実験せよ。session.html からユーザ ID を入力すると、それ以降セッションを終了するまで session1.php ではユーザのアクセス回数を記録できる。これらの情報は、サーバ側に保存され、ユーザ側ではこれを直接操作できない。

session.html

```
<HTML>
  <BODY>
    <FORM ACTION="session1.php" METHOD="GET">
      ユーザ ID<INPUT TYPE="text" NAME="uid"><INPUT TYPE="submit">
    </FORM>
  </BODY>
</HTML>
```

session1.php

```
<?php
        session_start();
        if( !isset( $_SESSION['count'] ))
                $_SESSION['count'] = 0;
        $_SESSION['count']++;
        if( isset($_GET['uid']) )
                $_SESSION['id'] = $_GET['uid'];
        session_commit();

        print "<html> <body>";
        print $_SESSION['id'].'<br>';
        print $_SESSION['count'].'<br>';
        print
        print "<a href='session2.php'>session exit page</a>
                </body></html>";
?>
```

session2.php

```
<?php
        session_start();
        $_SESSION = array();

        if (isset($_COOKIE[session_name()]))
           setcookie(session_name(), '', time()-3600, '/');

        session_destroy();

        print "<html> <body>セッション削除";
        print "<a href='session.html'>session start page</a>
                </body></html>";
?>
```