# セキュアなWORDPRESSプラグインの作り方

2011年8月27日 (8月31日修正)
WordBeach Nagoya 中級セッション
WordBench 川崎
池田 百合子

# 自己紹介

- WordPress プラグイン作者
  - Ktai Style
  - Ktai Entry
- 18年の Mac ユーザー
- 旅行が好き
  - 今回も青春18きっぷで往復
- http://www.yuriko.net/
- @lilyfanjp

# ターゲット

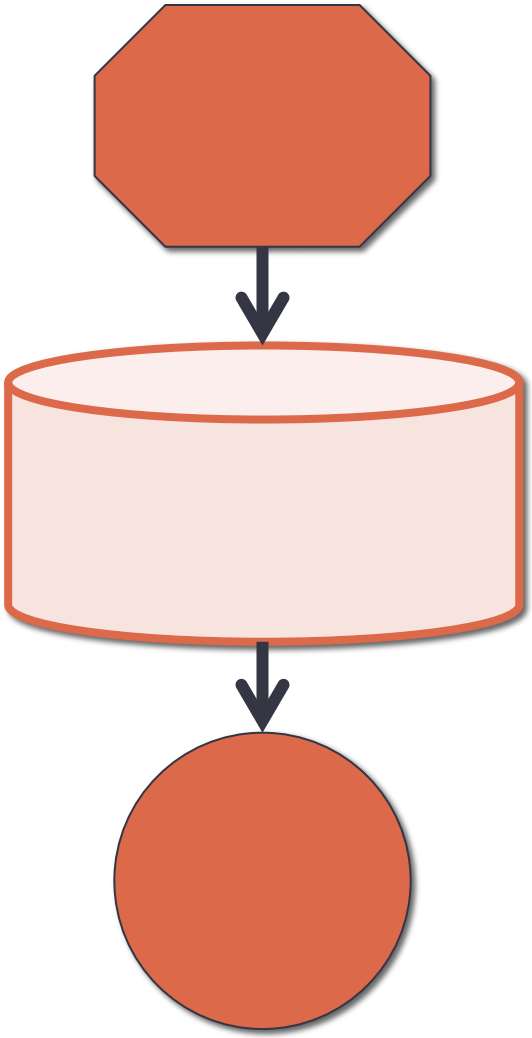
- テキストエディターをいじれる
- PHP は多少使える
- WordPress の使用経験あり
- プラグインの作成経験は不問
- 多少の英語力 (できれば)

# プラグインの仕組み

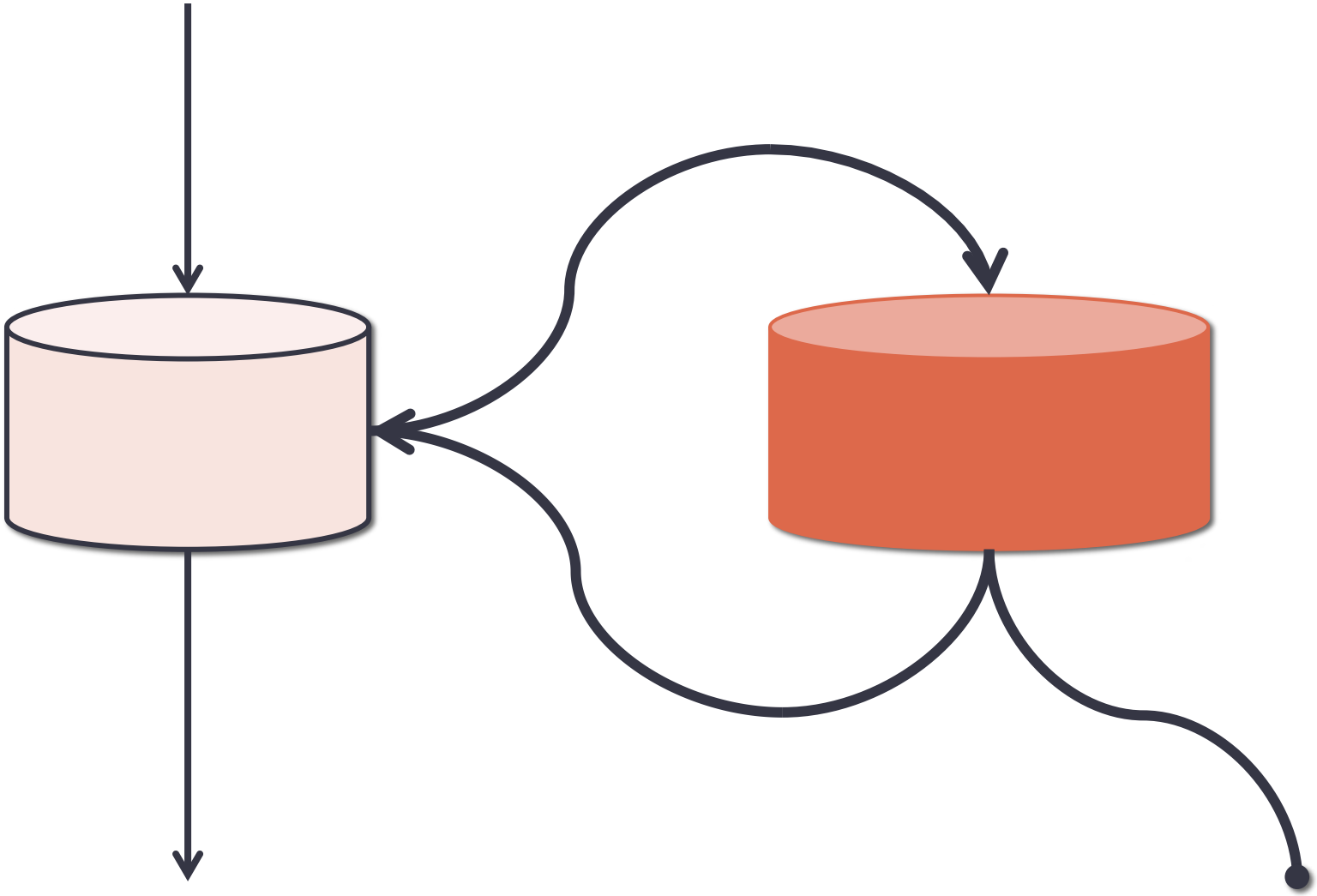
# フックという仕掛け

- WordPress 自体に仕掛けがある
  - データをいじるもの→フィルターフック
  - 動作をいじるもの→アクションフック

# フィルター



# アクション



# フィルターの例

```
function the_content($more_link_text=null,$stripteaser=0, $more_file='') {
  $content = get_the_content($more_link_text, $stripteaser, $more_file);
  $content = apply_filters('the_content', $content);
  $content = str_replace(']]>', ']]&gt;', $content);
  echo $content;
}

function strip_del($content) {
  $content = preg_replace('#<del[^>]*>.*?</del>\\s*#s',
 '', $content);
  return $content;
}
add_filter('the_content', 'strip_del');
```

# Delete Del 実施例

使用前

イベント参加のため、~~新名古屋駅~~名鉄名古屋駅から名鉄に乗った。

使用後

イベント参加のため、名鉄名古屋駅から名鉄に乗った。

# フックの探し方

- Codex で使いたいフックを探す

  http://wpdocs.sourceforge.jp/プラグイン_API/アクションフック一覧

  http://wpdocs.sourceforge.jp/プラグイン_API/フィルターフック一覧

- ソースコードから探す
  - 投稿関連なら
    - wp-includes/post.php
    - wp-includes/post-template.php
- wp-includes/pluggable.php に注意
  - ログイン処理などがここに多い

# プラグインの作り方

# プラグインはPHPコード

- プラグインは単なる PHP ソースコード
- 文字コードは UTF-8。BOMなし
- 改行コードは LF (CRLF は避ける)
- これらを扱えるテキストエディタが必要
  - EmEditor, PSPad, 秀丸, ...
  - テキストエディットapp, Jedit, CotEditor, SubEthaEdit, ...
  - もちろん vi, Emacs も OK

# 標準プラグイン情報

- プラグインとして認識されるには
  先頭に標準プラグイン情報を入れる。

```
/*
Plugin Name: プラグインの名前 (必須)
Plugin URI: プラグインの紹介ページ
Description: 説明
Version: バージョン番号 (x.y.z) (必須)
Author: 作者名
Author URI: 作者のウェブサイト
*/
```

- 順序は不問

# プラグインの例

```
/*
Plugin Name: Delete Del
Version: 0.7.0
Author: IKEDA Yuriko
*/

function strip_del($content) {
   $content = preg_replace('#<del[^>]*>.*?</del>\\s*#s',
  '', $content);
   return $content;
}
add_filter('the_content', 'strip_del');
```

# プラグインの名前

- 単純で分かりやすく唯一であること
- 英数字、記号(ハイフン、アンダースコア)と空白のみ
- 機能を端的に示す英単語を複数使う
  - 繋げたり、造語にしてもよい
  - 普通名詞で1単語は避ける
- 自分の名前・ハンドル・屋号は含まない
- 「WordPress」は含まない
- 「wp-なんちゃら」は避ける

# 公式リポジトリ登録を推奨

http://wordpress.org/extend/plugins/

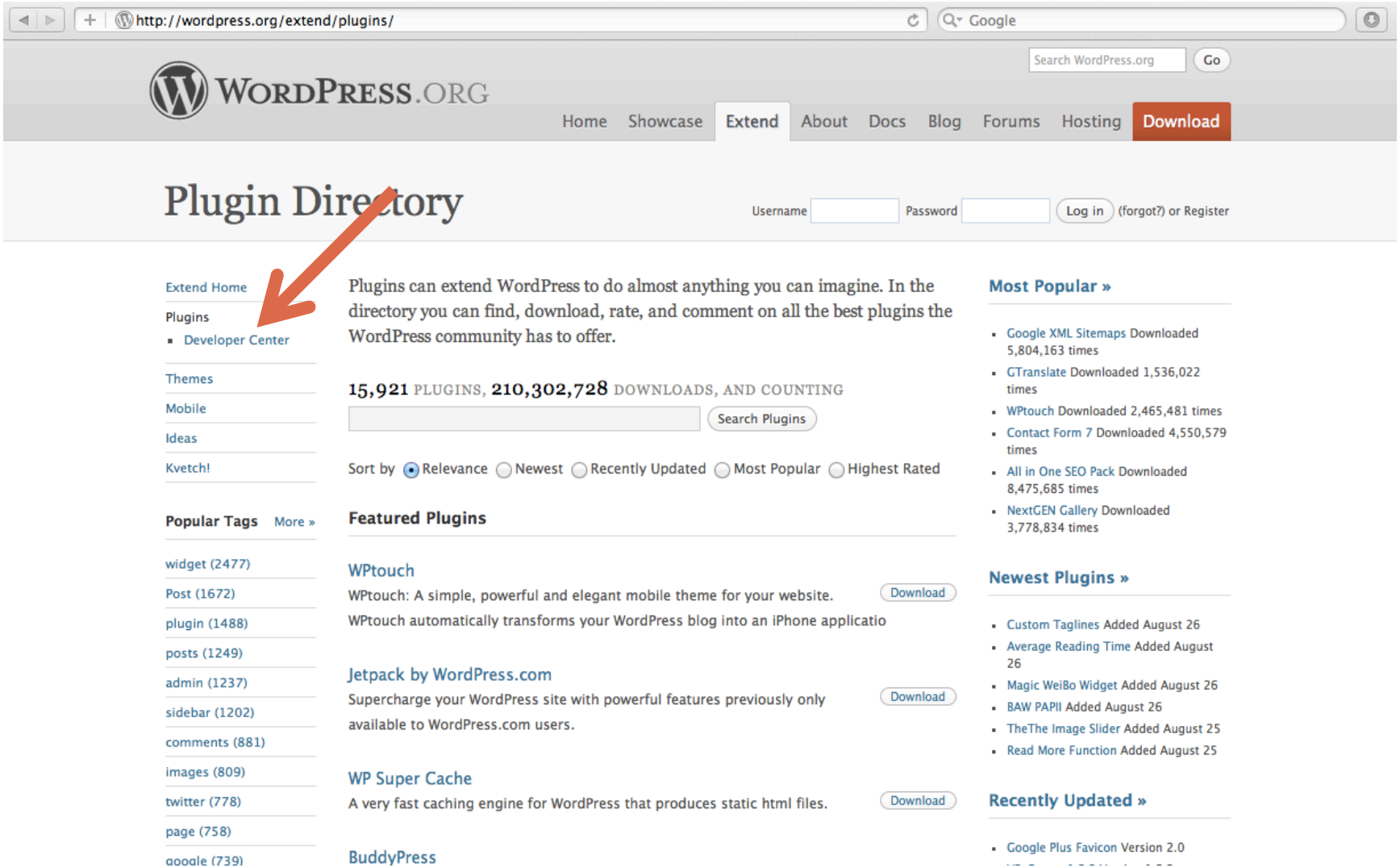
利点

- リポジトリ内で名前のユニーク性が保証
- 自動アップデートに対応

難点

- 英語の README 作成
- Subversion でのコード管理

# Developer Center を使おう



# プラグインのセキュリティー

# 考えなしに作ると脆弱性が

- 入力値は信用しない。必ず検証
  - 整数値／HTML／メールアドレス等
  - Codex 日本語版「データの検証」
    http://wpdocs.sourceforge.jp/Data_Validation
- 値をそのまま出すと、たいていXSS 発生
  - 値や文脈に応じて適切なエスケープ

# 主な脆弱性

# XSS 脆弱性 Cross Site Scripting

- 内容
  - スクリプトを意図せずサイトに埋め込める
- 実害
  - 管理者クッキーを取得して権限奪取
  - フィッシングサイトへの転送
- 原因
  - エスケープ忘れ
- 対策
  - 出力値を正しくエスケープ
- 一番多く発生するが、攻略は難しい

# CSRF 脆弱性 Cross Site Request Forgeries

- 内容
  - スクリプトを使い、攻略サイトで重要な操作をさせる
- 実害
  - 新規投稿作成／投稿削除／パスワード変更など
- 原因
  - セッションチェックの不備
- 対策
  - wp_nonce を適切に使う。
    - `nonce_field(), check_admin_referer()`

# SQL インジェクション脆弱性

- 内容
  - 意図しないSQLクエリを実行
- 実害
  - 情報漏洩／サイト書き替え
- 原因
  - エスケープ忘れ／エスケープ削除し過ぎ
- 対策
  - プレースホルダーを適切に使う `$wpdb->prepare()`
- `$_GET, $_POST, $_SESSION` が `addslashes()` 済なので発生しにくい……

# ディレクトリートラバーサル脆弱性

- 内容
  - ディレクトリーを遡って重要ファイルを入手・改変
- 実害
  - wp-config.php を盗まれる
  - OS の /etc/hosts をいじられる
- 原因
  - パスに `../` を許している
- 対策
  - 特定のディレクトリより上はアクセスさせない。
  - `validate_file()` などを使う。

# HTTP ヘッダーインジェクション

- 内容
  - HTTP ヘッダに任意の内容を埋め込まれる
- 実害
  - 意図しないクッキー受け入れ
  - 意図しないリダイレクト→偽ページの表示
- 原因
  - 表示する内容に CRLF が入っている
- 対策
  - `header()` を使う (PHP 4.4.2+, 5.1.2+)
  - リダイレクトは `wp_redirect()`, `wp_safe_redirect()`

# セキュアなコーディング

# 入力値の検証

- セキュリティーよりはバグ潰しメイン
- プラグインの仕様をきっちり決める
  - 数値? 文字列? URL? メールアドレス? ファイルパス?

```
• $count = intval($count);
• if ( !is_string($_POST['fullname']) )
    { wp_die('名前を入力してください'); }
• $link= esc_url_raw($link);
• if ( !is_email($addr) )
    { wp_die('正しいアドレスを入れてください'); }
• $path = validate_file($_POST['path']);
```

# 出力値のエスケープ

- 入力値をそのまま出すとたいていXSS
  - `echo $_GET['fullname']` とかダメ
- 適切なエスケープが必要
  - `intval()`: 数値
  - `esc_html()`: HTML 平文
  - `esc_attr()`: 属性値
  - `esc_url()`: URL
  - `esc_js()`: スクリプト

# HTML のエスケープ

- ```
  printf('<a href="%s" title="%s">%s</a>',
  esc_url($url),
  esc_attr($title),
  esc_html($link));
  ```
  ```

# SQL クエリはプレースホルダで

- `$sql = $wpdb->prepare("SELECT * FROM `{$wpdb->prefix}ktaisession` WHERE sid = %s", $sid);`
- `$result = $wpdb->get_row($sql);`

# フィルターフックの場合

- 入力・出力ともに WordPress コード
  - 入力値の検証・出力時のエスケープは WordPress の仕事
- 入力値の「状態」および返り値の流れを把握すべし
- 例: the_content
  - 入力値は HTML コード
  - 返り値がそのまま出力
    - ただしdefault-filters.php で the_content 用フィルター設定あり
  - 返り値に `$_GET`,`$_SERVER` 等を生で含めてはいけない

# レビューとテスト

# コードレビューしましょう

- セキュリティーの観点でコードを見る
  - 入力値は検証されているか?
  - 出力値はエスケープされているか?
  - エスケープできない箇所は問題ない値のみが来るか?
- 値の「状態」に着目する
  - 入力値の生の状態
  - 検証された状態
  - HTML 用にエスケープされた状態
  - SQL 用のエスケープされた状態

# テストしましょう

- 入力値は嫌らしい値で
  - 数字しかない場所で文字列 `0x17` 歳
  - HTML コードや半角カナ `<s>ﾏｸﾗｹﾝ直子</s>`
  - スクリプト `<script>alert();</script>`
  - ウェブURLでスクリプト `javascript:alert();`
  - SQL 断片 `' OR 1=1;`
  - 不正なメールアドレス `-_-.@example.com`
  - 正当なメールアドレス `"i,yuriko"@example.jp`

# Windows 環境でのテスト

- 本番環境は UNIX が多い
  - Windows 環境での動作が考慮されてないことがある
- バグを見つけやすい
- ワークショップで WebMatrix の使い方を学ぼう
- Mac の人は Boot Camp や Virtualbox で

# 以上

# スライド配布

- http://www.yuriko.net/tag/slides/
- http://wordbeach.org/