Содержание

1. Подсистема взаимодействия с БД WordPress

Текущая стабильная версия: 3.6.1 (15.10.2013).

1. Перечень поддерживаемых СУБД.

MySQL, частично портирована PostgreSQL.

2. Используется ли сторонняя библиотека или собственный код системы для взаимодействия с БД.

Используется самописная система взаимодействия с БД. Были предложения о взаимодействии с ADOdb.

3. Поддерживаются ли настраиваемые администратором префиксы к именам таблиц?

Да, префиксы таблиц настраиваются с помощью `wpdb::set_prefix( $prefix, $set_table_names = true )`.

4. Поддержка ООП (используется ли ORM – какая библиотека, какой подход). При поддержке ORM – как решены вопросы сложных запросов с JOIN по нескольким таблицам?

ORM не используется.

5. Делится ли система на компоненты и поддерживаются ли разные их версии?

Система делится на следующие компоненты: ядро, темы, плагины. Обновление может затронуть любой файл из начальной установки WordPress. Сторонние же плагины и темы при этом не обновляются.

6. Как поддерживаются сообщения об ошибках при работе с БД?

Для работы с ошибками существует класс `WP_Error`, исключения не используются.

7. Создание и обновление таблиц БД происходит в декларативном или в процедурном стиле?

Схема БД описывается процедурно, обновляется также процедурно, следует помещать SQL-запрос в функцию `dbDelta`, которая гарантирует минимальные и безопасные изменения в схеме БД. Создание дополнительных таблиц не приветствуется WordPress.

8. Кто осуществляет эскейпинг введенных пользователем данных — библиотека взаимодействия с БД или программист? Используются ли подготовленные операторы (prepared statements). Какие типы параметров поддерживаются?

В WP3.6.1 используются подготовленные операторы (функция `wpdb::prepare( $query, $args )`). Типы параметров — integer, float, string. Ранее с 0.71 использовалась функция `wpdb::escape( $data )`. Эскейпинг осуществляется программистом.

9. Влияние хранения данных на архитектуру

Данные хранятся в основном, в 3 таблицах:

- wp_posts;
- wp_comments;
- wp_users.

Таким образом, хранимый контент следует относить к 2 основным категориям: либо к постам, либо к комментариям.

10. Каким образом осуществляется защита от доступа к изменяемым данным: транзакции, блокировки (локи) на уровне таблиц и строк:

В самой CMS каких-либо механизмов блокировки не найдено. Так как основная БД Wordpress — MySQL, а MySQL поддерживает несколько механизмов хранения данных, среди которых — MyISAM и InnoDB, то возможны варианты:

1) При использовании MyISAM получаем:

- отсутствие транзакций;
- отсутствие построчной блокировки;

- наличие блокировки таблиц.
2) При использовании InnoDB:
- наличие транзакций;
- наличие построчной блокировки;
- наличие блокировки таблиц.

Поскольку, с MySQL 5.5 механизм хранения по умолчанию — InnoDB, можно сказать, что есть возможность проведения транзакций и блокировки на уровнях таблиц и строк.

11. Есть ли возможность создавать поля типа TimeStamp с автоматической подстановкой значений при обновлении строки?

Есть возможность установки TimeStamp в полях таблиц.

12. При наличии функций, генерирующих SQL - есть ли возможность самостоятельно написать сложный SQL-запрос (SELECT) самостоятельно и выполнить его?

Генерации SQL как таковой не найдено. Т.е. программист должен писать на SQL. Для функций общения с БД необходимо передавать SQL-запрос. Есть возможность создать сложный SELECT-запрос, к примеру, выборка постов по параметрам - для этого используется функция `wp_query`.

13. Как выглядит возвращаемый функциями DML объект, представляющий строку таблицы?

– Строку таблицы можно получить через `wpdb::get_row`, возвращается ассоциативный массив, в котором ключи представляют собой названия полей, а значения - значения полей.

Пример получения строки:

```
$link = $wpdb->get_row("SELECT * FROM $wpdb->links WHERE link_id = 42");
echo $link->link_id;
```

2. Подсистема взаимодействия с WEB WordPress

1. Каким образом представлены индивидуальные страницы, URL для них? Организация страниц и получение параметров.

Поскольку основным назначением WordPress остаётся создание блогов, существует 2 типа страниц:

- страницы постов;
- обычные страницы.

Что такое обычные страницы?

Это места для хранения информации, не привязанной ко времени, тегам и другим вещам, которые характерны для постов.

В общем, если не использовать страницы типа “post” можно построить обычный сайт-не блог.

Каждая страница создаётся отдельным файлом php.

Пути по умолчанию имеют следующие виды так называемых пермалинков (permalinks):

1. Post Permalink: example.com/?p=123
2. Page Permalink: example.com/?page_id=123
3. Category Permalink: example.com/?cat=123
4. Tag Permalink: example.com/?tag=tag-name
5. Archive Permalink: example.com/?m=201212 or example.com/?m=2012

Можно настроить свой вид ссылки, например для лучшей SEO, используя слова-шаблоны в URL.

2. Как организовано получение параметров (GET, POST), проверка их типов и фильтрация значений?

Для получения входных параметров используются стандартные механизмы PHP, такие как суперглобальные массивы $_GET и $_POST, механизмов WordPress для проверки типов не было найдено.

3. Как происходит вывод HTML в программе?

Страницы могут использовать HTML-файлы с PHP-вставками.

Вывод HTML производится модулями самостоятельно.

4. Используются ли шаблоны страниц, какая система шаблонизации?

Да, используются шаблоны страниц. Система шаблонизации по умолчанию - это вставки PHP-кода в HTML. Есть возможность подключить сторонние шаблонизаторы, например, Smarty.

5. Имеется ли специальная библиотека для организации веб-форм?

Специальной библиотеки не используется. Однако, существуют плагины для создания веб-форм, например FormMaker, Gravity и др.

6. Система тем. Могут ли темы влиять на код вывода элементов или они содержат лишь описания стилей?

Тема WordPress состоит из трёх основных типов файлов. Первый - это таблица стилей под названием style.css, которая контролирует внешний вид страниц сайта. Второй представляет собой файл дополнительной функциональности (functions.php). Остальные файлы - это файлы шаблонов, которые определяют каким образом выводится информация из базы данных на ту или иную веб-страницу.

Минимальная тема может состоять из 2 файлов: style.css и index.php.

7. Имеется ли возможность определять блоки, дополнительные к основному содержанию страницы, как плагины?

Есть такая возможность. Можно создать скрипт, который будет выполнять некоторые действия, а затем встроить его в шаблон страницы используя механизм включения PHP.

8. Использование javascript. Имеется ли система вывода javascript или каждая страница решает этот вопрос индивидуально?

Определённых инструментов для этого нет, каждая страница подключает файл как обычно в HTML/PHP.

9. Использование AJAX. Есть ли AJAX-библиотека, собственной разработки

или сторонняя, кратко о библиотеке?

AJAX-библиотека самописная - встроенная в ядро, активно используется как WP, так и сторонними модулями.

3. Управление доступом WordPress

1. Какие виды аутентификации система поддерживает. Возможно ли добавление новых видов аутентификации как плагинов? Как устроен плагин аутентификации? Привязана ли аутентификация жестко к паре данных логин/пароль или плагин может определять, какая информация ему нужна для аутентификации?

Аутентификация в Wordpress построена на системе плагинов аутентификации. Сайт может использовать один или несколько плагинов аутентификации вместе. Типовой плагин аутентификации реализует функцию-крючок (hook) authenticate. Также есть возможность использовать функцию wp_authenticate_user, которая позволяет производить дополнительные проверки.

Кроме модуля аутентификации Wordpress Default (Логин/Пароль), присутствуют следующие:

- LDAP;
- Active Directory;
- OpenID;
- Smart cards;
- множество других плагинов, например:

1) Duo Two-Factor Authentication (двухфакторная аутентификация);
2) Loginza (аутентификация через социальные сети);
3) Swekey (аутентификация с помощью USB-токенов);

и т.д.

Пример работы с хуками аутентификации:

```
add_filter('authenticate',  'authenticate_username_password', 20, 3);

function authenticate_username_password( $user, $username, $password ){
  if ( is_a( $user, 'WP_User' ) )return $user;
  if ( empty( $username ) || empty( $password ) ){
  $error = new WP_Error();
  if( empty( $username ) )
    $error->add( 'empty_username', __( '<strong>ERROR</strong>: The username
```

```
field is empty.' ) );
  if( empty( $password ) )
    $error->add( 'empty_password', __( '<strong>ERROR</strong>: The password
field is empty.' ) );
  return $error;
 }
 $success = check_username_password( $username, $password );

 if( !$success )
  return new WP_Error(
   'invalid_username',
     sprintf( __( '<strong>ERROR</strong>: Invalid username. <a href="%s"
title="Password Lost and Found">Lost your password</a>?' ),
    site_url( 'wp-login.php?action=lostpassword', 'login' )
   )
  );
 $user_id = username_exists( $username );
 if( !$user_id ) {
  $random_password = wp_generate_password( 18, false );
  $user_id = wp_create_user( $username, $random_password );
 }
  $user = get_user_by( 'id', $user_id );
 return $user;
}
```

2. Какие виды контекстов (областей информации, на которые может быть назначена роль) система поддерживает? Существует ли иерархия контекстов?

Глобальных контекстов 3:

- контекст сайта из сети сайтов;
- контекст отдельного сайта;
- контекст панели сайта.

Некоторые Capabilities (Возможности) ролей зависят от контекста - например, роль Administrator в контексте отдельного сайта, а не сайта сети, даёт ему дополнительные Capabilities (об основных - следующий пункт):

- update_core (обновлять ядро)
- update_plugins (обновлять плагины)
- update_themes (обновлять темы)

- install_plugins (устанавливать плагины)
- install_themes (устанавливать темы)
- delete_themes (удалять темы)
- edit_plugins (изменять плагины)
- edit_themes (изменять темы)
- edit_users (изменять пользователей)
- create_users (создавать пользователей)
- delete_users (удалять пользователей)

То есть даёт возможность распоряжаться всем, вплоть до обновления ядра WordPress. Это делает её эквивалентной роли Super Administrator в контексте одного сайта.
Однако, если роль Administrator назначается в контексте одного из сайтов сети, то право на такие серьёзные действия, как обновление WordPress и установка плагинов имеет только Super Administrator, управляющий всеми сайтами сети.

3. Являются ли роли в системе фиксированными, или задаваемыми пользователем? Каким образом пользователь задает роль?
Wordpress имеет 6 предопределённых ролей: Super Administrator, Administrator, Editor, Author, Contributor и Subscriber.

Super Administrator - главная роль, позволяет выполнять любые действия над отдельными сайтами и сайтами сети. В Wordpress сетью сайтов (network) называется группа пользовательских сайтов, которые могут быть созданы на основе главного сайта. Т.е. для пользователей некоторого сайта появляется возможность создать свой блог, на основе этого главного сайта, который будет пользоваться теми же плагинами, темами и настройками.
Имеет Capabilities:
- manage_network (управлять сетью сайтов)
- manage_sites (управлять сайтами)
- manage_network_users (управлять пользователями сети сайтов)
- manage_network_plugins (управлять плагинами сети сайтов)

- manage_network_themes (управлять темами сети сайтов)
- manage_network_options (управлять настройками сети сайтов)

Administrator – роль, которая позволяет выполнять любые действия над отдельным сайтом и, практически любые (см. выше) над сайтом сети.
Имеет Capabilities:
- activate_plugins (активировать плагины)
- delete_pages (удалять страницы)
- delete_plugins (удалять плагины)
- delete_posts (удалять посты)
- delete_private_pages (удалять личные страницы)
- delete_private_posts (удалять личные посты)
- delete_published_pages (удалять опубликованные страницы)
- delete_published_posts (удалять опубликованные посты)
- edit_dashboard (редактировать рабочий стол панели управления)
- edit_files (редактировать файлы)
- edit_pages (редактировать страницы)
- edit_posts (редактировать посты)
- edit_private_pages (редактировать личные страницы)
- edit_private_posts (редактировать личные посты)
- edit_published_pages (редактировать опубликованные страницы)
- edit_published_posts (редактировать опубликованные посты)
- edit_theme_options (изменять настройки темы)
- export (экспорт записей, комментариев)
- import (импорт записей, комментариев)
- list_users (просматривать список пользователей)
- manage_categories (управлять категориями постов, ссылок)
- manage_links (управлять ссылками)
- manage_options (управлять настройками)
- moderate_comments (модерировать комментарии)
- publish_pages (публиковать страницы)
- publish_posts (публиковать посты)
- read_private_pages (читать личные страницы)

- read_private_posts (читать личные посты)
- read
- remove_users (удалять пользователей)
- switch_themes (менять тему для сайта)
- upload_files (загружать файлы)

Editor – роль, которая позволяет публиковать и управлять постами, включая посты других пользователей.
Имеет Capabilities:
- delete_pages (удалять страницы)
- delete_posts (удалять посты)
- delete_private_pages (удалять личные страницы)
- delete_private_posts (удалять личные посты)
- delete_published_pages (удалять опубликованные страницы)
- delete_published_posts (удалять опубликованные посты)
- edit_pages (редактировать страницы)
- edit_posts (редактировать посты)
- edit_private_pages (редактировать личные страницы)
- edit_private_posts (редактировать личные посты)
- edit_published_pages (редактировать опубликованные страницы)
- edit_published_posts (редактировать опубликованные посты)
- manage_categories (управлять категориями)
- manage_links (управлять ссылками)
- moderate_comments (модерировать комментарии)
- publish_pages (публиковать страницы)
- publish_posts (публиковать посты)
- read
- read_private_pages (читать личные страницы)
- read_private_posts (читать личные посты)
- upload_files (загружать файлы)

Author – роль, которая позволяет публиковать свои посты и управлять ими.
Имеет Capabilities:

- delete_posts (удалять посты)
- delete_published_posts (удалять опубликованные посты)
- edit_posts (редактировать посты)
- edit_published_posts (редактировать опубликованные посты)
- publish_posts (публиковать посты)
- read
- upload_files (загружать файлы)

Contributor – роль, которая позволяет писать посты и управлять ими, однако без возможности их публикации.
Имеет Capabilities:
- delete_posts (удалять посты)
- edit_posts (редактировать посты)
- read (читать)

Subscriber – роль, которая позволяет читать содержимое сайта и управлять своей учётной записью.
Имеет Capabilities:
- read

Плагин User Role Editor позволяет менять состав Capabilities для ролей (кроме Administrator).
Есть возможность программно создавать, изменять, удалять роли.

```
add_role('member', 'Member', array(
    'read' => true,
    'edit_posts' => true,
    'delete_posts' => true,
));

remove_role('subscriber');
```

4. Может ли пользователь переопределить возможности роли в конкретном контексте?

Да, используя плагин User Role Editor.

5. Каким образом программно происходит проверка возможности осуществления действий пользователя?

Проверка может осуществляться несколькими функциями

```
if ( user_can( $user_id, 'moderate_comments' ) ) {
    echo 'User '.$user_id.' can moderate comments';
}
```

```
if ( current_user_can_for_blog($blog_id, 'moderate_comments') ) {
    echo 'The current user can moderate comments for blog id='.$blog_id;
}
```

```
if ( current_user_can( 'moderate_comments' ) ) {
    echo 'The current user can moderate comments';
}
```

6. Каким образом система защищена от следующих типовых угроз:

6.1. Подделка запросов (cross-site request forgery - CSRF)

Защита от CSRF возложена на подключаемые модули.
Например, WP-Sentinel.

6.2. Вставка скриптов в код страницы (cross-site scripting - XSS)

Защита от XSS возложена на подключаемые модули.
Например, Anti-XSS attack.

6.3. Вставка кода - SQL injection

Защита от SQL Injection возложена на подключаемые модули.
Например, Injection Guard.

Кроме того, существуют решения, которые обеспечивают защиту от всех упомянутых типов уязвимостей.
Такими модулями являются:

- BulletProof Security - защищает от XSS, RFI, CRLF, CSRF, Base64, Code Injection и SQL Injection;
- All In One WP Security & Firewall;
- SmartFilter Security.