

Урок №7

Работа с пользователями. Авторизация и разделение прав доступа.

В этом уроке

- Вы узнаете, как грамотно организовать авторизацию пользователей на сайте;
- какие для этого нужно таблицы в БД;
- что такое шифрование паролей, и зачем оно нужно;
- поймете, как организовать систему со смежными правами доступа у пользователей.

1. Авторизация

Раз Вы добрались до данного урока, то, скорее всего, представляете общие идеи создания авторизации на сайте. Как обычно, у нас есть два хранилища, с помощью которых мы можем ставить метки пользователям: куки и сессии. Вокруг них и будет крутиться вся реализация системы авторизации на сайте.

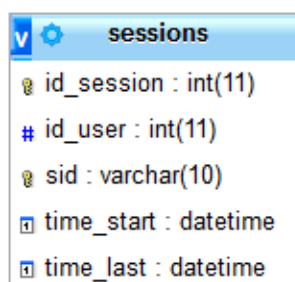
Для начала нам необходимо где-то хранить соотнесение пар логин-пароль. В базе создадим для этого таблицу **users**:



	users
🔑	id_user : int(5)
🔑	login : varchar(256)
📄	password : varchar(32)
#	id_role : int(5)

Данные четыре поля являются базовыми. В реальности в таблице users могут быть и другие, например, имя, фамилия, возраст и т.п. Первые три поля являются интуитивно понятными, а четвёртое нас пока не интересует, так как отвечает не за авторизацию, а за разделение прав доступа.

Дополнительно к данной, создадим таблицу **sessions**, в которой будем хранить информацию о заходах пользователей на сайт. Без создания этой таблицы будет невозможно определить, онлайн ли сейчас человек на сайте.



	sessions
🔑	id_session : int(11)
#	id_user : int(11)
🔑	sid : varchar(10)
🕒	time_start : datetime
🕒	time_last : datetime

Подробнее разберём поля данной таблицы.

- **id_session** – первичный ключ;
- **id_user** – номер авторизованного пользователя;
- **sid** – случайная последовательность букв и цифр для идентификации пользователя;
- **time_start** – время открытия сессии, т.е., время авторизации на сайте;
- **time_last** – время последнего посещения любой страницы сайта, т.е., время последней активности пользователя.

В момент, когда человек успешно авторизуется на сайте, мы кидаем ему метки в куки и сессию PHP, а в таблицу **sessions** вносится новая запись. Данная запись будет существовать около 20 минут. По истечении этого времени мы считаем, что пользователь не онлайн на сайте и удаляем строку из таблицы.

Ниже приводится таблица совершаемых пользователем действий, и их влияние на метки об авторизации:

Действие\Сущность	Кука, если ставили галочку запомнить	Кука, если НЕ ставили галочку запомнить	Сессия PHP	Запись в таблице sessions
Авторизация	создаётся	отсутствует	создаётся	создаётся
Выход из системы	удаляется	её и не было	удаляется	удаляется
Отошли от компьютера	остаётся	отсутствует	остаётся	существует примерно 20 мин
Закрыли браузер	остаётся	отсутствует	удаляется	существует примерно 20 мин

Соответственно, для того, чтобы при заходе пользователя на страничку определить, авторизован он или нет, нам нужно посмотреть на все поставленные метки.

В первую очередь мы будем ориентироваться на сессию PHP, так как она является краткосрочным хранилищем. В случае её отсутствия единственным возможным вариантом того, что пользователь авторизован, является наличие куки. Отсутствие куки и сессии однозначно говорит о том, что человек не авторизован.

Наличие только куки означает, что человек долго отсутствовал, и его нужно автоматически авторизовать заново. А вот наличие только сессии PHP заставляет нас посмотреть, есть ли такая же запись в базе в таблице **sessions**. Если нет, человек не авторизован.

Ниже приводится таблица состояния меток и итогового результата:

Кука	Сессия PHP	Запись в таблице sessions	Вывод
0	0	0	не авторизован
0	0	1	не ставил галочку, закрыл браузер - не авторизован
0	1	0	не ставил галочку, отсутствовал более 20 минут - не авторизован
0	1	1	не ставил галочку, авторизован
1	0	0	поставил галочку, долго отсутствовал - авторизован
1	0	1	поставил галочку, закрыл браузер - авторизован
1	1	0	поставил галочку, отсутствовал более 20 минут - авторизован
1	1	1	поставил галочку, авторизован

Если мы считаем, что человек авторизован, то в любом случае возобновляем сессию PHP и запись в базе, даже если они отсутствовали. Реализацию данного механизма в PHP смотрите в исходниках в модели M_Users.

При анализе кода также обратите внимание на использование функции md5 при проверке правильности ввода пароля. Дело в том, что хранить пароль в незащищённом виде в базе опасно, поэтому применяются алгоритмы необратимого шифрования. Если кто-то вдруг украдёт шифр пароль, то всё равно не сможет его использовать для авторизации на сайте.

2. Простая система прав

Простая система разделения прав доступа заключается в создании всего лишь одного дополнительного поля в таблице **users**, например, **access_level**. У обычного пользователя значение данного поля будет равно 1, у модератора 2, у админа 3 и т.п.

В PHP для проверки прав доступа достаточно будет на определённых страницах проверять, является ли значение поля **access_level** большим или равным заранее установленному.

Такая система подойдёт для большинства сайтов, потому что чаще всего права доступа имеют иерархический вид, т.е., модератор может то же самое, что и обычный

пользователь плюс что-то дополнительно, администратор – то же самое, что и модератор плюс ещё что-то дополнительное.

3. Смежные права

Смежные права – такой тип разделения доступа к страницам сайта, при котором отсутствует иерархичность, описанная двумя строками выше.

Рассмотрим возможные смежные права на примере блога.

Действие \ Тип пользователя	Создание статьи	Редактирование статьи	Просмотр всех статей	Просмотр Конкретной статьи	Редактирование списка пользователей сайта
Обычный	1	1, но только свои	1	1	0
Модератор	0	1	1	1	0
Админ	1	1	1	1	1

Как мы видим, модератор выбивается из привычной иерархической системы, так как не может создавать записи. В сложных проектах смежные права встречаются очень часто.

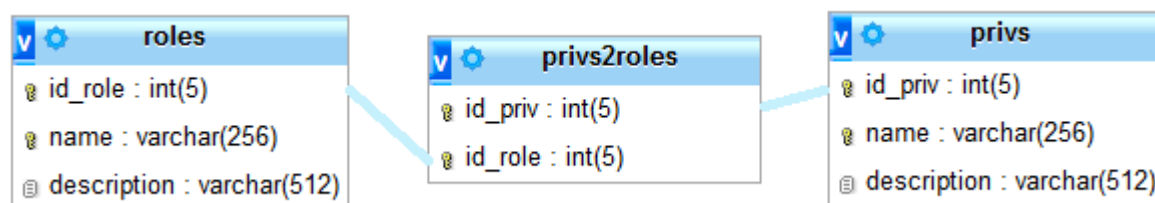
Рассмотрим, каким же образом мы можем их организовать. Сначала разберёмся с основными понятиями.

Пользователь – сущность, ассоциируемая с конкретным человеком, имеющим доступ к системе.

Привилегия – возможность, которой может обладать пользователь (например, «просмотр конкретной страницы»).

Роль – группа привилегий.

В базе нам потребуется создать ещё три дополнительные таблицы:



Сейчас вспомните четвёртый столбец в таблице **users** – **id_role**. Он показывает, кем пользователь является на сайте. Сами же сущности ролей и привилегий связаны как «многие ко многим». Для проверки прав доступа в РНР нужно будет по номеру пользователя и имени запрошенного действия сделать такой запрос к базе, который

определит, существует ли соответствие в записях, т.е., обладает ли данный пользователь этой привилегией. Реализация данного запроса будет являться одним из пунктов Вашего домашнего задания.

Рассмотренная система является универсальной и может использоваться Вами в совершенно любых проектах.

Самоконтроль

- ✓ Какие поля являются основными в таблице пользователей
- ✓ Зачем применяется шифрование паролей
- ✓ Зачем мы ввели таблицу сессий
- ✓ Как различные действия пользователя влияют на куки, сессии PHP и записи в базе в таблице sessions
- ✓ Как мы понимаем, авторизован человек на сайте или нет
- ✓ Чем простая система разделения прав доступа отличается от смежных прав
- ✓ Что такое привилегия
- ✓ Что такое роль
- ✓ Зачем сделана дополнительная таблица privs2roles

Домашнее задание

1. Реализовать функцию Can в модели M_Users.
2. Добавить возможность регистрации нового пользователя.