\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Список литературы:**<https://github.com/RAptorZed/Presentation-Twig.git> **-** GitHub(репозиторий с исходниками).  
<http://x-twig.ru/> - русифецированая документация  
<http://twig.sensiolabs.org/pdf/Twig.pdf> - книга.  
<http://ruseller.com/lessons.php?rub=37&id=1652> - стаття (основной материал).  
<https://github.com/fabpot/Twig/archive/master.zip> - GitHub(ссылка на репозиторий twig).

<http://x-twig.ru/coding_standards/>- код стандарт twig.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Презентация**  
1) Вступление.

Обычное PHP приложение состоит из целого набора страниц, которые включают в себя как статический HTML код (меню, списки, изображения и т.д.), так и динамический контент (вывод данных из БД, xml файла, сервисы, …). С помощью Twig мы можем разделить данные процессы, создавая шаблоны со специальными маркерами, вместо которых в последствии будет вставляться динамический контент.

Значения для данных маркеров формируются в основном PHP скрипте; там же происходит общение с базой данных, xml парсинг и другие всевозможные операции. Таким образом, ваша страница будет строиться на основе 2х источников: шаблона с специальными вставками и PHP скриптов, где мы храним основной функционал. Это даёт возможность PHP разработчикам и дизайнерам одновременно работать над одними и теми же страницами.   
 Иными словами twig помогает разделить бизнес логику приложения и вывод данных на страницу.

2) Установка.

Существует множество способов установки Twig-a. Самый простой и быстрый - это скачивание компонента с [GitHub](https://github.com/fabpot/Twig/archive/master.zip), после чего архив необходимо распаковать, и перекинуть каталог lib в папку с нашим проектом. (ссылка в списке литературы)  
 Через терменал - можна использовать такие команды:





3)Примеры  
 Для использования Twig-а, нам нужно пройти следующие шаги:

1. Инициализировать авто-загрузчик Twig-а, для того чтобы классы шаблонизатора подгружались автоматически.   
     
   **<?php**// подгружаем и активируем авто-загрузчик Twig-а  
   **require\_once** 'Twig/Autoloader.php';  
   Twig\_Autoloader::*register*();
2. Инициализировать загрузчик шаблонов. В нашем случае этo Twig\_Loader\_FileSystem. В качестве аргумента передаём путь к каталогу с шаблонами.   
   // указывае где хранятся шаблоны  
   $loader = **new** Twig\_Loader\_Filesystem('templates');
3. Создать объект самого Twig и передать ему уже сконфигурированные настройки.  
   // инициализируем Twig

$twig = **new** Twig\_Environment($loader);

1. Подгрузить нужный нам шаблон с помощью метода loadTemplate, передав в него название используемого шаблона. В качестве результата метод вернёт экземпляр шаблона.

// подгружаем шаблон

$template = $twig->loadTemplate('thanks.tmpl');

1. Сформировать массив вида "ключ-значение", где ключи - это названия переменных, а значения - данные, выводимые в шаблоне. Затем этот массив нужно передать в метод render(), который совместит шаблон с переданными данными и вернёт сгенерированный результат.

// передаём в шаблон переменные и значения

// выводим сформированное содержание

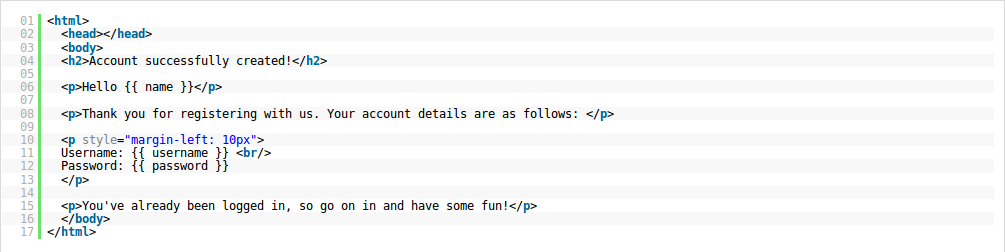
**echo** $template->render(**array**(

'name' => 'Clark Kent',

'username' => 'usernametwr13123123',

'password' => 'userpassword3123123213123',

));

(3-1)Для того чтобы посмотреть, как работает Twig, давайте рассмотрем простой пример:  
 Сохраните данный файл templates/thanks.tmpl. Обратите внимание на то, что все переменные, помещены в двойные фигурные скобки. Подобная запись подскажет Twig-у, где и как осуществлять вставку данных.

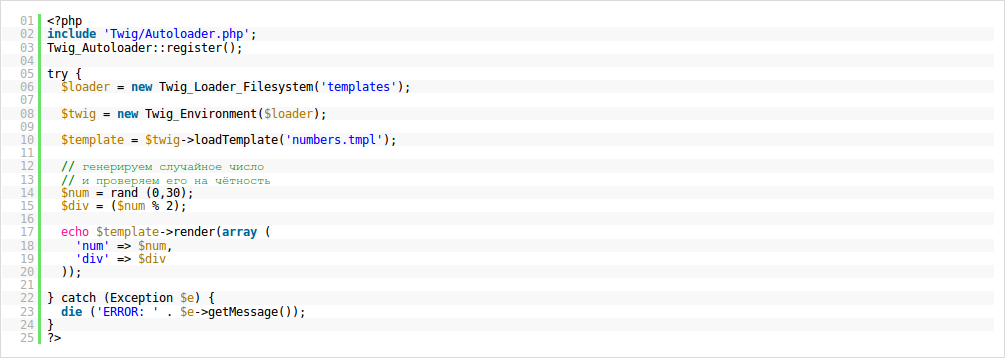
Затем, нам необходимо создать основной скрипт, где будет происходить формирование переменных и данных:



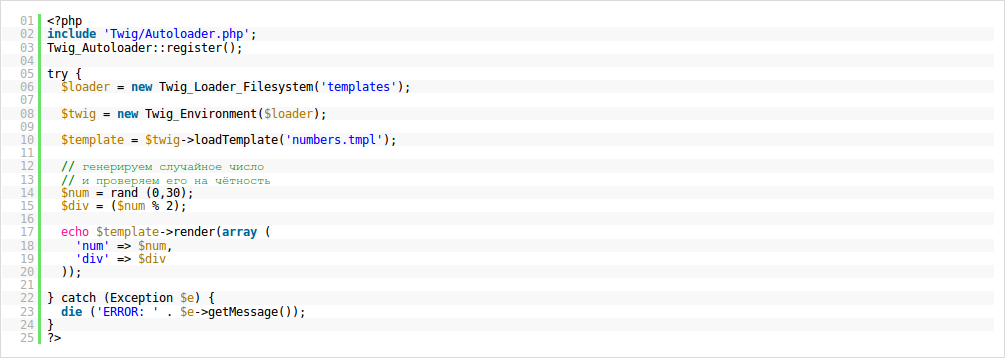
(3-2) Условия**.** Twig также предоставляет нам возможность создавать условные выражения ‘if-else-endif’. Пример:

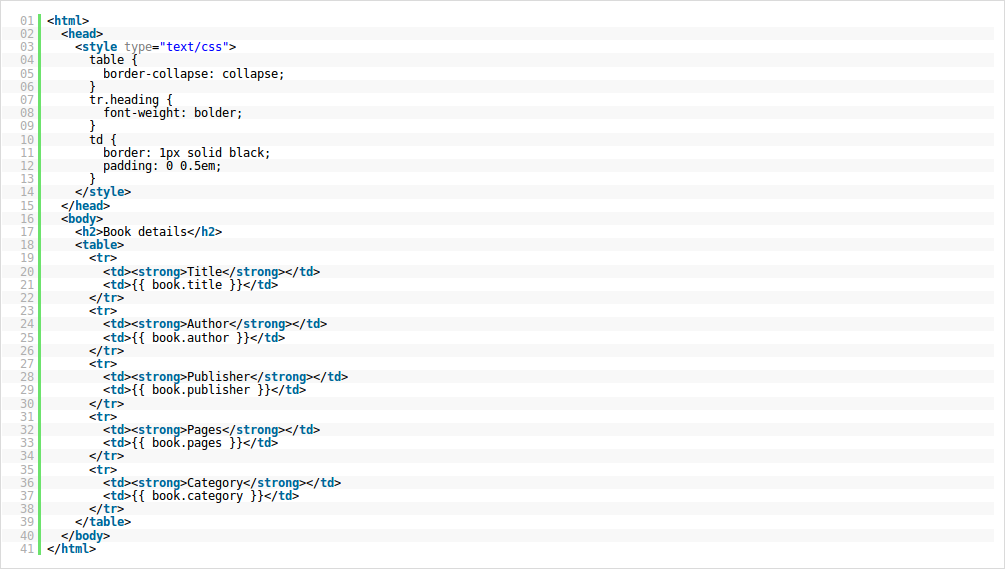


В зависимости от числа, которое генерируется в основном PHP скрипте, шаблон отобразит одно из двух сообщений. Вот и скрипт, где генерируется число от 0 до 30 и проверяется на чётность:



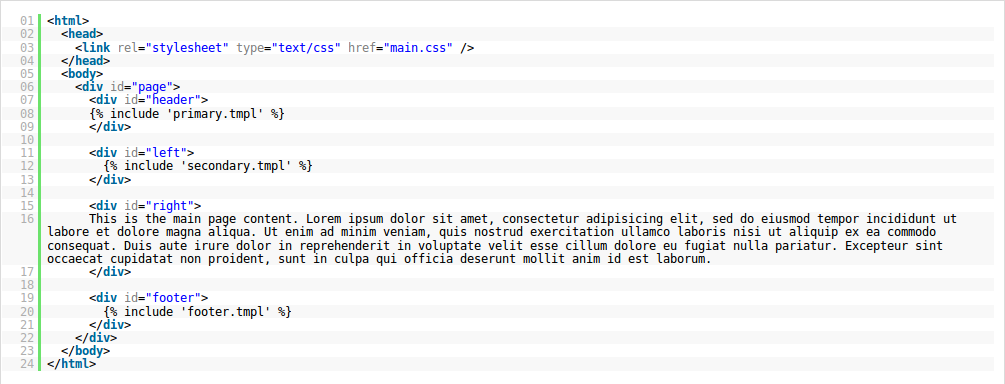
(3-4)Цыкл. Twig также поддерживает цикл ‘for’. Он очень удобен, если нам необходимо пройтись по массиву. Пример

В данном примере у нас простой не ассоциативный массив. На каждой итерации мы будем получать по одному элементу и выводить его в элементе списка. Вот и скрипт:  


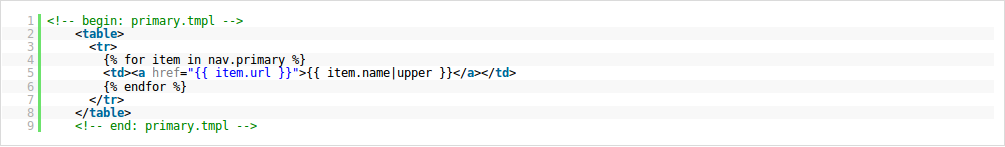
(3-5)Цыкл с ассоциативным массивом. Для того чтобы пройтись по ассоциативному массиву, мы можем обращаться к ключам через “точку”. Пример:  
  
 Для того чтобы достучаться до значений массива в шаблоне, сначала пишем имя переменной, в которой хранится сам массив. Затем ставим точку и пишем название ключа, по которому достаём данные:

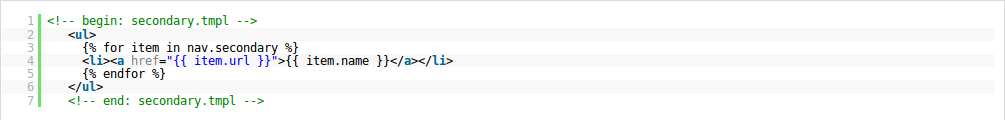
(3-6) Подгрузка шаблонов. Также в Twig-е есть ещё одна команда - `include`, которая позволяет подключать содержание других шаблонов. Это может пригодиться, когда вы захотите прикрепить к вашим файлам меню, заголовок или подвал.

Для демонстрации представьте, что данный код - это главный шаблон:



Все секции данной страницы находятся в отдельных файлах и подключаются сюда с помощью команды `include`. Давайте посмотрим, как выглядят подключаемые файлы:







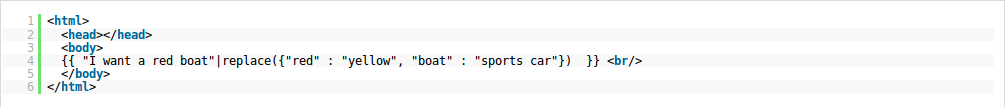
А вот и главный PHP скрипт:  
Обратите внимание, что нам не нужно загружать все шаблоны функцией loadTemplate. Главное подключить основной шаблон. Каждый мелкий под-шаблон загрузится автоматом. Переменные и значения, переданные в главный шаблон, будут доступны во всех подключаемых шаблонах.

(3-7)Фильтрация данных. Давайте рассмотрим, к примеру, фильтр ‘date’. Данный фильтр даёт нам возможность формировать дату и время, используя нативные для PHP маркеры. Пример:



Также вы можете воспользоваться фильтрами `upper`, `lower`, `capitalize`, `title` для контроля заглавных и прописных букв:  


Фильтр `striptags` уберёт из текста все HTML и XML элементы:  


Фильтр `replace` позволяет быстро и просто заменять какие-то значения в строке на нужные нам. Пример:

Вы уже видели фильтр `escape` в действии. В Twig также есть фильтр, который делает абсолютно противоположное действие - `raw`. Его следует использовать только для html кода, который вы считаете 100% безопасным.  


Если же вам нужно применить `escape` к большому блоку кода, то вы можете воспользоваться синтаксисом `autoescape`, передав булево значение true/false для активации и дезактивации фильтрации `escape`. Пример: