## 生命周期

参考资料: https://blog.csdn.net/qw\_xingzhe/article/details/81051336

Laravel的生命周期从public/index开始，在public/index.php结束。.

1. 文件载入composer生成的自动加载设置，包括所有你 composer require的依赖。
2. 生成容器Container，Application实例，并向容器注册核心组件（HttpKernel，ConsoleKernel ，ExceptionHandler）（对应代码2，容器很重要，后面详细讲解）。
3. 处理请求，生成并发送响应（对应代码3，毫不夸张的说，你99%的代码都运行在这个小小的handle 方法里面）。
4. 请求结束，进行回调（对应代码4，还记得**[可终止中间件](https://laravel.com/docs/5.3/middleware" \l "terminable-middleware" \t "_blank)**吗？没错，就是在这里回调的）。

### 服务容器

Laravel 服务容器是用于管理类的依赖和执行依赖注入的工具。依赖注入这个花俏名词实质上是指：类的依赖项通过构造函数，或者某些情况下通过「setter」方法「注入」到类中。

1. laravel表单提交

自定义中间件 session保存登陆信息

1. 引入验证码 和验证

<https://learnku.com/articles/6736/laravel-code-captcha>

<https://blog.csdn.net/everything1209/article/details/53466725>

1. 正则表达式
2. 上传

1、获取上传的文件

$file=$request->file('file');

2、获取上传文件的文件名（带后缀，如abc.png）

$filename=$file->getClientOriginalName();

3、获取上传文件的后缀（如abc.png，获取到的为png）

$fileextension=$file->getClientOriginalExtension();

4、获取上传文件的大小

$filesize=$file->getClientSize();

5、获取缓存在tmp目录下的文件名（带后缀，如php8933.tmp）

$filaname=$file->getFilename();

6、获取上传的文件缓存在tmp文件夹下的绝对路径

$realpath=$file->getRealPath();

7、将缓存在tmp目录下的文件移到某个位置，返回的是这个文件移动过后的路径

$path=$file->move(path,newname);

move()方法有两个参数，第一个参数是文件移到哪个文件夹下的路径，第二个参数是将上传的文件重新命名的文件名

8、检测上传的文件是否合法，返回值为true或false

$file->isValid()

9．文件上传

Php的常用做法

<https://www.cnblogs.com/lihaiyan/p/4274442.html>

js的常用方法

<https://blog.csdn.net/qq_26702033/article/details/79637678>

#### 11.使用form表单提交

##### 2. 使用FormData对象

**FormData** 接口提供了一种表示表单数据的键值对的构造方式，经过它的数据可以使用 [XMLHttpRequest.send()](https://developer.mozilla.org/zh-CN/docs/Web/API/XMLHttpRequest/send) 方法送出

因为表单上传过程中，整个页面就刷新了！ajax异步上传就可以达到只刷新局部位置

、

Form表单提交时全局刷新，有的时候业务需求并不是这样，全局刷新会浪费带宽，因此局部刷新产生—即ajax

<https://cloud.tencent.com/info/52e270ea09b899437f73fd99de10293f.html>

XMLHttpRequest Level 2添加了一个新的接口FormData，利用FormData对象，我们可以使用XMLHttpRequest的send()方法异步的提交数据，比起普通的ajax，使用FormData的最大优点是可以异步上传一个二进制文件

对于前端人员来说，图片处理是一个很常见的需求，由于图片稍微特殊，现在多数做法都是使用调用ajax接口通过http方法来提交，例如post方法提交，后台处理后返回一个图片路径给前端，前端根据这个路径写入img标签，但是基于当前的前后端分离的开发模式下，前后端代码往往不在同一个系统目录下，而且部署时可能liunx路径与windows路径不一样，这样后期路径更改可能会导致维护困难问题出现。

针对这种问题，这里我推荐使用图片转base64格式，再发给后端，后端只需将转码结果存入数据库即可，前端调用接口直接获取到base64数据直接写入img src 标签即可