FLAnimatedImageView+WebCache

```
self.sd_setImageWithURL_placeholderImage_options_progress_completed(url,placeholder,optionsSDWebImageDownloaderProgressBlock{<block>},SDExternalCompletionBlock{<block>});
```

UIImageView+WebCache

```
self.sd_setImageWithURL_placeholderImage_options_progress_completed(url,placeholder,optionsSDWebImageDownloaderProgressBlock{<block>},SDExternalCompletionBlock{<block>});
```

UIImageView+WebCache
UIImageView+HighlightedWebCache
UIButton+WebCache
MKAnnotationView+WebCache
NSImage+WebCache

UIView+WebCache

```
self.sd_internalSetImageWithURL_placeholderImage_options_operationKey_setImageBlock_progress_completed(url, placeholder, options,
operationKey,SDSetImageBlock,SDWebImageDownloaderProgressBlock,SDExternalCompletionBlock) {
    self.sd_cancelImageLoadOperationWithKey:
    self_sd_setImage: imageData: basedOnClassOrViaCustomSetImageBlock:
    SDWebImageCombinedOperation *operation = SDWebImageManager.sharedManager.loadImageWithURL_options_progress_completed(url, options,
    SDWebImageDownloaderProgressBlock, SDInternalCompletionBlock<completedBlock> {
        SDExternalCompletionBlock completedBlock(image, error, cacheType, url);
    });
    self.sd_setImageLoadOperation_forKey(operation, validOperationKey);
}
```

SDWebImageManager

```
SDWebImageCombinedOperation *operation = self.loadImageWithURL_options_progress_completed(url, options, SDWebImageDownloaderProgressBlock, SDInternalCompletionBlock) {
    SDWebImageCombinedOperation *operation = [SDWebImageCombinedOperation new];
    NSString *key = [self cacheKeyForURL:url];
    operation.cacheOperation = self.imageCache.queryCacheOperationForKey_done(key, SDCacheQueryCompletedBlock<doneBlock> {
        Case1: SDWebImageDownloadToken *subOperationToken = self.imageDownloader.downloadImageWithURL_options_progress_completed(url, options,
        SDWebImageDownloaderProgressBlock, SDWebImageDownloaderCompletedBlock<completedBlock> {
            //允许外部在图片下载下来之后缓存之前对图片进行一些操作(如圆角操作)。
            self.delegate.imageManager_transformDownloaderImage_withURL(self, downloadedImage, url);
            self.imageCache.storeImage_imageData_forKey_toDisk_completion(transformedImage, downloadedData, key, cacheOnDisk, nil);
            SDInternalCompletionBlock completionBlock(image, data, error, cacheType, finished, url);
        });
        self.imageDownloader_cancel(subOperationToken);
        Case2: SDInternalCompletionBlock completionBlock(image, data, error, cacheType, finished, url);
    });
}
```

SDWebImageDownloader

```
SDWebImageDownloadToken *subOperationToken = self.downloadImageWithURL_options_progress_completed(…);
= self.addProgressCallBack_completedBlock_forURL_createCallBack(SDWebImageDownloaderProgressBlock, SDWebImageDownloaderCompletedBlock, url,
SDWebImageDownloaderOperation*<createCallBack>{
    SDWebImageDownloaderOperation *operation = …;
    return operation;
}) {
    SDWebImageDownloaderOperation *operation = self.URLOperations[url] ? : createCallback();
    1、以url作为key在self.URLOperations中存取operation对象。
    2、id downloadOperationCancelToken = operation.addHandlersForProgress_completed(SDWebImageDownloaderProgressBlock, SDWebImageDownloaderCompletedBlock);
        token.downloadOperationCancelToken = downloadOperationCancelToken;
        return token;
}
```