ブロック構文の使い方

～プログラムでプロック構文を使ってみる～

Cocoa勉強会関西

２０１２年5月19日

ブロック構文とは？

■C言語レベルの構文によるランタイム機能

■Mac OS X 10.6／iOS 4.0 以降で利用可能

■ブロック定義スコープ内の変数を参照／変更

■マルチスレッドで利用可

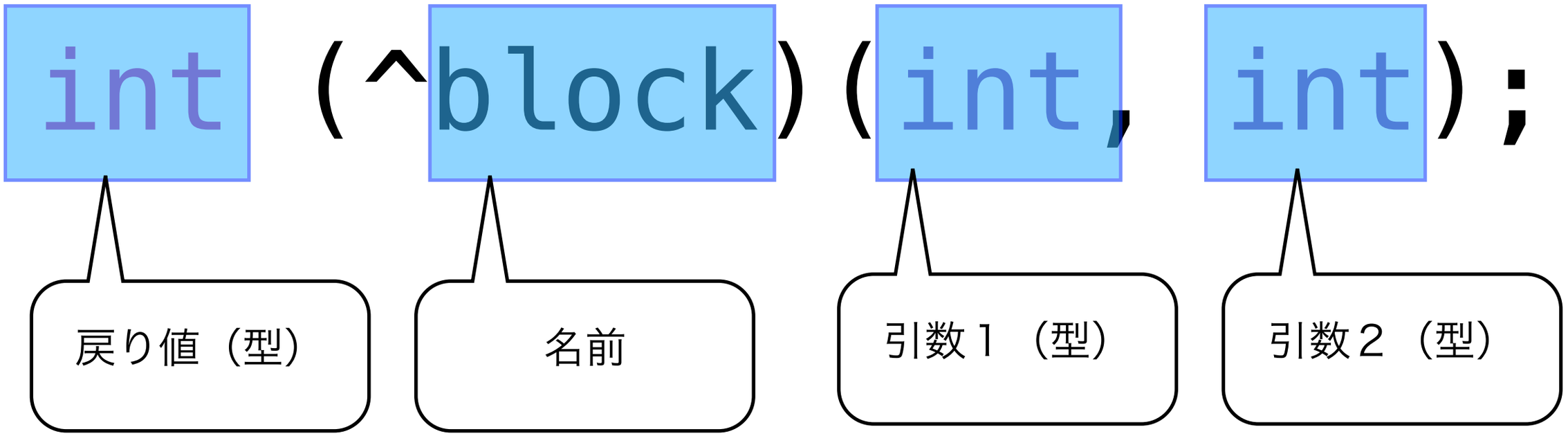
■コールバック関数として利用できる

ブロック構文の基本的なところ

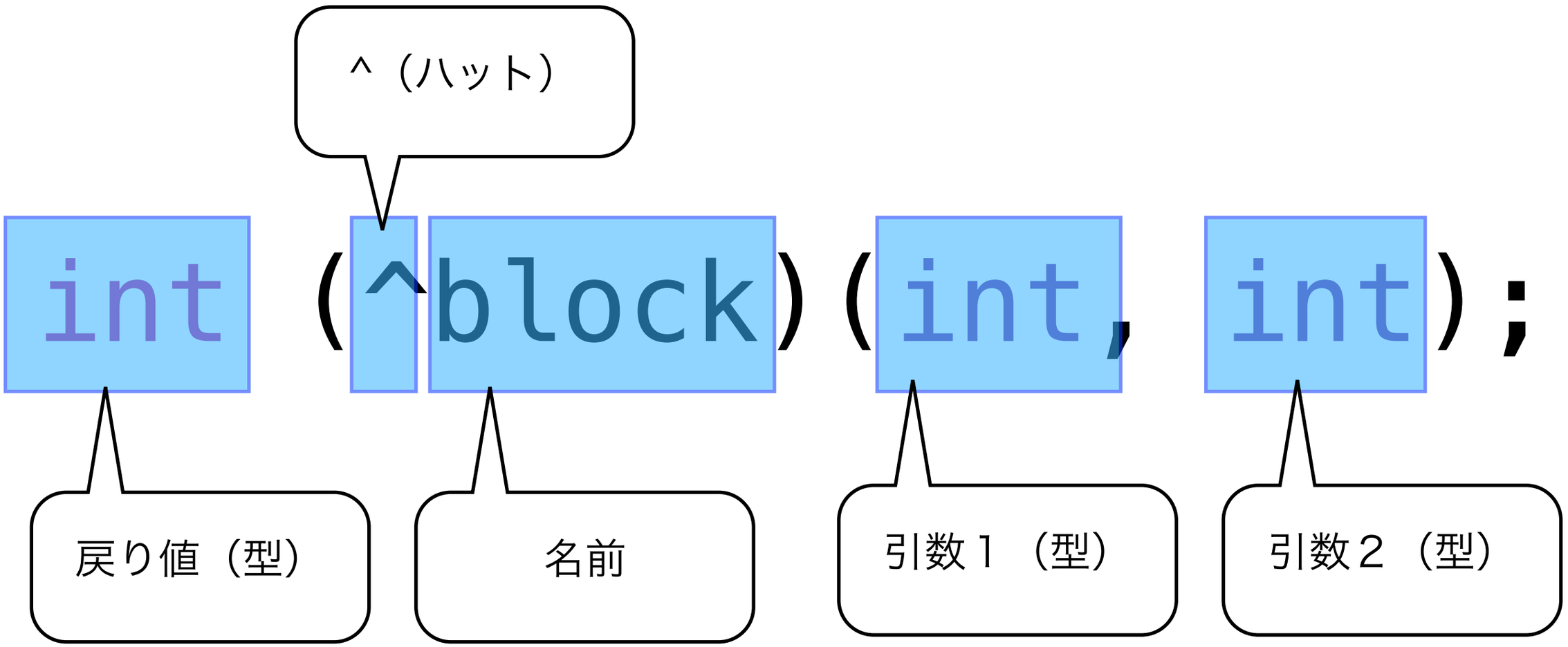
ブロックスの宣言

```
int (^block)(int, int);
```

ブロックスの宣言



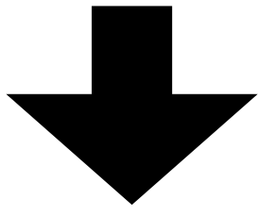
ブロックスの宣言



ブロックスの宣言

```
int (^block)(int, int);
```

C言語の関数だと…



```
int    block(int, int);
```

ブロックスを使う

```
int (^block)(int,int) = ^(int a,int b){
    return a + b;
};
```

```
intc = block(1,2);

NSLog(@"%d", c);
```

ブロックスを使う

```
int (^block)(int,int) = ^(int a,int b){
    return a + b;
};
```

```
intc = block(1,2);
```

```
NSLog(@"%d", c);
```

ブロックスを使う

宣言

初期化

```
int (^block)(int,int) = ^(int a,int b){
    return a + b;
};
```

```
intc = block(1,2);
```

呼び出し

```
NSLog(@"%d", c);
```

ブロックスを使う

宣言

初期化

```
int (^block)(int,int) = ^(int a,int b){
    return a + b;
};
```

```
intc = block(1,2);
```

呼び出し

```
NSLog(@"%d", c);
```

ブロックスを使う

宣言

初期化

```
int (^block)(int,int) = ^(int a,int b){
    return a + b;
};
```

```
intc = block(1,2);
```

呼び出し

```
NSLog(@"%d", c);
```

ブロックスを使う

宣言

初期化

```
int (^block)(int,int) = ^(int a,int b){
    return a + b;
};
```

```
intc = block(1,2);
```

呼び出し

```
NSLog(@"%d", c);
```

3

メソッドの引数としてブロックスを使う

メソッドの宣言

```
-(void)blockFunction:(int(^)(int a,int b)) block;
```

メソッドの引数としてブロックスを使う

メソッドの宣言

```
-(void)blockFunction:(int(^)(int a,int b)) block;
```

メソッド名

型宣言

名前

メソッドの引数としてブロックスを使う

メソッドの実装

```
-(void)blockFunction:(int(^)(int a,int b))block;
{
    int c = block(1,2);
    NSLog(@"c=%d", c);
}
```

メソッドの引数としてブロックスを使う

メソッドの実装

```
-(void)blockFunction:(int(^)(int a,int b))block;
{
    int c = block(1,2);
    NSLog(@"c=%d", c);
}
```

メソッドの呼び出し

```
[self blockFunction:^int(int a, int b) {
    return a+b;
}];
```

メソッドの引数としてブロックスを使う

メソッドの実装

```
-(void)blockFunction:(int(^)(int a,int b))block;
{
    int c = block(1,2);
    NSLog(@"c=%d", c);
}
```

メソッドの呼び出し

```
[self blockFunction:^int(int a, int b) {
    return a+b;
}];
```

メソッドの引数としてブロックスを使う

メソッドの実装

```
-(void)blockFunction:(int(^)(int a,int b))block;
{
    int c = block(1,2);
    NSLog(@"c=%d", c);
}
```

メソッドの呼び出し

```
[self blockFunction:^int(int a, int b) {
    return a+b;
}];
```

（1） メソッド呼び出し

メソッドの引数としてブロックスを使う

メソッドの実装

```
-(void)blockFunction:(int(^)(int a,int b))block;
{
    int c = block(1,2);
    NSLog(@"c=%d", c);
}
```

（2） ブロックス呼び出し

メソッドの呼び出し

```
[self blockFunction:^int(int a, int b) {
    return a+b;
}];
```

（1） メソッド呼び出し

メソッドの引数としてブロックスを使う

メソッドの実装

```
-(void)blockFunction:(int(^)(int a,int b))block;
{
    int c = block(1,2);
    NSLog(@"c=%d", c);
}
```

（２） ブロックス呼び出し

メソッドの呼び出し

```
[self blockFunction:^int(int a, int b) {
    return a+b;
}];
```

（１） メソッド呼び出し

（３） ロジック処理

メソッドの引数としてブロックスを使う

メソッドの実装

```
-(void)blockFunction:(int(^)(int a,int b))block;
{
    int c = block(1,2);
    NSLog(@"c=%d", c);
}
```

（2） ブロックス呼び出し

3

（1） メソッド呼び出し

メソッドの呼び出し

```
[self blockFunction:^int(int a, int b) {
    return a+b;
}];
```

（3） ロジック処理

ブロック構文の代表的な使用方法

コールバックとして

ブロックスを利用

NSURLConnectionを
使ったダウンロード

NSURLConnectionを使った画像のダウンロード

ヘッダ

```
@interface ViewController : UIViewController

@property (strong, nonatomic) NSMutableData *dlData;
@property (weak, nonatomic) IBOutlet UIImageView *imageView;

-(IBAction)download:(id)sender;

@end
```

メソッド

```
-(IBAction)download:(id)sender
{
    NSString*     urlPath = @"http://xxxxx/image.png";
    NSURL*        url = [NSURL URLWithString:urlPath];
    NSURLRequest* request = [NSURLRequest requestWithURL:url];

    NSURLConnection*  connection;
    connection = [NSURLConnection connectionWithRequest:request
                                                delegate:self];
}
```

NSURLConnectionを使った画像のダウンロード

ヘッダ

```
@interface ViewController : UIViewController

@property (strong, nonatomic) NSMutableData *dlData;
@property (weak, nonatomic) IBOutlet UIImageView *imageView;

-(IBAction)download:(id)sender;

@end
```

ダウンロードされるデータ

メソッド

```
-(IBAction)download:(id)sender
{
    NSString*     urlPath = @"http://xxxxx/image.png";
    NSURL*        url = [NSURL URLWithString:urlPath];
    NSURLRequest* request = [NSURLRequest requestWithURL:url];

    NSURLConnection*  connection;
    connection = [NSURLConnection connectionWithRequest:request
                                                delegate:self];
}
```

NSURLConnectionを使った画像のダウンロード

ヘッダ

```
@interface ViewController : UIViewController

@property (strong, nonatomic) NSMutableData *dlData;
@property (weak, nonatomic) IBOutlet UIImageView *imageView;

-(IBAction)download:(id)sender;

@end
```

ダウンロードされるデータ

ダウンロードした画像を
表示するImage View

メソッド

```
-(IBAction)download:(id)sender
{
    NSString*     urlPath = @"http://xxxxx/image.png";
    NSURL*        url = [NSURL URLWithString:urlPath];
    NSURLRequest* request = [NSURLRequest requestWithURL:url];

    NSURLConnection*  connection;
    connection = [NSURLConnection connectionWithRequest:request
                                                delegate:self];
}
```

NSURLConnectionを使った画像のダウンロード

ヘッダ

```
@interface ViewController : UIViewController

@property (strong, nonatomic) NSMutableData *dlData;
@property (weak, nonatomic) IBOutlet UIImageView *imageView;

-(IBAction)download:(id)sender;

@end
```

ダウンロードされるデータ

ダウンロードした画像を
表示するImage View

ダウンロードする画像の
URL

メソッド

```
-(IBAction)download:(id)sender
{
    NSString*     urlPath = @"http://xxxxx/image.png";
    NSURL*        url = [NSURL URLWithString:urlPath];
    NSURLRequest* request = [NSURLRequest requestWithURL:url];

    NSURLConnection*  connection;
    connection = [NSURLConnection connectionWithRequest:request
                                                delegate:self];
}
```

NSURLConnectionを使った画像のダウンロード

ヘッダ

```
@interface ViewController : UIViewController

@property (strong, nonatomic) NSMutableData *dlData;
@property (weak, nonatomic) IBOutlet UIImageView *imageView;

-(IBAction)download:(id)sender;

@end
```

ダウンロードされるデータ

ダウンロードした画像を
表示するImage View

ダウンロードする画像の
URL

メソッド

```
-(IBAction)download:(id)sender
{
    NSString*     urlPath = @"http://xxxxx/image.png";
    NSURL*        url = [NSURL URLWithString:urlPath];
    NSURLRequest* request = [NSURLRequest requestWithURL:url];

    NSURLConnection*  connection;
    connection = [NSURLConnection connectionWithRequest:request
                                                delegate:self];
}
```

NSURLRequest作成

NSURLConnectionを使った画像のダウンロード

ヘッダ

```
@interface ViewController : UIViewController

@property (strong, nonatomic) NSMutableData *dlData;
@property (weak, nonatomic) IBOutlet UIImageView *imageView;

-(IBAction)download:(id)sender;

@end
```

ダウンロードされるデータ

ダウンロードした画像を
表示するImage View

ダウンロードする画像の
URL

メソッド

```
-(IBAction)download:(id)sender
{
    NSString*     urlPath = @"http://xxxxx/image.png";
    NSURL*        url = [NSURL URLWithString:urlPath];
    NSURLRequest* request = [NSURLRequest requestWithURL:url];

    NSURLConnection*  connection;
    connection = [NSURLConnection connectionWithRequest:request
                                               delegate:self];
}
```

NSURLRequest作成

NSURLConnection作成　非同期でダウンロード開始

NSURLConnectionを使った画像のダウンロード

NSURLConnectionのデリゲートメソッド

```
//ダウンロード開始。データを受けるNSDataオブジェクトを作成する
- (void)connection:(NSURLConnection *)connection
didReceiveResponse:(NSURLResponse *)response
{
  self.dlData = [[NSMutableData alloc] initWithLength:0];
}
```

```
//ダウンロード中。データを受ける
- (void)connection:(NSURLConnection *)connection
  didReceiveData:(NSData *)data
{
  [self.dlData appendData:data];
}
```

NSURLConnectionを使った画像のダウンロード

NSURLConnectionのデリゲートメソッド

```
//ダウンロード完了
- (void)connectionDidFinishLoading:(NSURLConnection *)connection
{
    UIImage*  image = [UIImage imageWithData:self.dlData];
    imageView.image = image;
    self.dlData = nil;
}
```

NSURLConnectionを使った画像のダウンロード

iOS 5からこういうメソッドが追加されました。

```
+(void)sendAsynchronousRequest:(NSURLRequest*)request
                         queue:(NSOperationQueue*)queue
completionHandler:(void^(NSURLResponse* responce, NSData* data, NSError* error))handler;
```

実装例

```
NSString*      urlPath = @"http://xxxxx/image.png";
NSURL*         url = [NSURL URLWithString:urlPath];
NSURLRequest*  request = [NSURLRequest requestWithURL:url];

[NSURLConnection sendAsynchronousRequest:request
                                   queue:[[NSOperationQueue alloc] init]
completionHandler:^(NSURLResponse* responce, NSData* data, NSError* error){
             if(data){


             }
}];
```

NSURLConnectionを使った画像のダウンロード

iOS 5からこういうメソッドが追加されました。

```
+(void)sendAsynchronousRequest:(NSURLRequest*)request
                         queue:(NSOperationQueue)queue
completionHandler:(void^(NSURLResponse* responce, NSData* data, NSError* error))handler;
```

実装例

```
NSString*     urlPath = @"http://xxxxx/image.png";
NSURL*        url = [NSURL URLWithString:urlPath];
NSURLRequest* request = [NSURLRequest requestWithURL:url];

[NSURLConnection sendAsynchronousRequest:request
                                   queue:[[NSOperationQueue alloc] init]
completionHandler:^(NSURLResponse* responce, NSData* data, NSError* error){
            if(data){


            }
}];
```

NSURLConnectionを使った画像のダウンロード

iOS 5からこういうメソッドが追加されました。

```
+(void)sendAsynchronousRequest:(NSURLRequest*)request
                         queue:(NSOperationQueue)queue
completionHandler:(void^(NSURLResponse* responce, NSData* data, NSError* error))handler;
```

実装例

```
NSString*     urlPath = @"http://xxxxx/image.png";
NSURL*        url = [NSURL URLWithString:urlPath];
NSURLRequest* request = [NSURLRequest requestWithURL:url];

[NSURLConnection sendAsynchronousRequest:request
                                   queue:[[NSOperationQueue alloc] init]
completionHandler:^(NSURLResponse* responce, NSData* data, NSError* error){
            if(data){


            }
}];
```

ダウンロードに成功したら
dataが返される。

ブロックスでiOS 4でも使える
ダウンロードメソッドを作ってみる

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■インターフェース

```
typedef void(^DownloadBlocks)(NSData* data);

@interface ViewController : UIViewController{
   DownloadBlocks  dl_blocks;
}
@property   (nonatomic, retain) NSMutableData *dlData;
@property   (nonatomic, retain) IBOutlet UIImageView *imageView;

-(void)dlBlocks:(NSString*)urlPath Handler:(DownloadBlocks)handler;

-(IBAction)download:(id)sender;

@end
```

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■インターフェース

ブロックスを型定義する

```
typedef void(^DownloadBlocks)(NSData* data);

@interface ViewController : UIViewController{
   DownloadBlocks  dl_blocks;
}
@property   (nonatomic, retain) NSMutableData *dlData;
@property   (nonatomic, retain) IBOutlet UIImageView *imageView;

-(void)dlBlocks:(NSString*)urlPath Handler:(DownloadBlocks)handler;

-(IBAction)download:(id)sender;

@end
```

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■インターフェース

ブロックスを型定義する

```
typedef void(^DownloadBlocks)(NSData* data);

@interface ViewController : UIViewController{
   DownloadBlocks  dl_blocks;
}
@property   (nonatomic, retain) NSMutableData *dlData;
@property   (nonatomic, retain) IBOutlet UIImageView *imageView;

-(void)dlBlocks:(NSString*)urlPath Handler:(DownloadBlocks)handler;

-(IBAction)download:(id)sender;

@end
```

定義した型のインスタンス変数宣言

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■インターフェース

ブロックスを型定義する

```
typedef void(^DownloadBlocks)(NSData* data);

@interface ViewController : UIViewController{
   DownloadBlocks  dl_blocks;
}
@property   (nonatomic, retain) NSMutableData *dlData;
@property   (nonatomic, retain) IBOutlet UIImageView *imageView;

-(void)dlBlocks:(NSString*)urlPath Handler:(DownloadBlocks)handler;

-(IBAction)download:(id)sender;

@end
```

定義した型のインスタンス変数宣言

ダウンロードメソッド

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■メソッドの実装

```
-(void)dlBlocks:(NSString*)urlPath Handler:(DownloadBlocks)handler
{
   NSURL*           url = [NSURL URLWithString:urlPath];
   NSURLRequest*   request = [NSURLRequest requestWithURL:url];

   NSURLConnection*   connection;
   connection = [NSURLConnection connectionWithRequest:request delegate:self];
   if(connection){
       dl_blocks = Block_copy(handler);
   }
}
```

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■メソッドの実装

```
-(void)dlBlocks:(NSString*)urlPath Handler:(DownloadBlocks)handler
{
   NSURL*           url = [NSURL URLWithString:urlPath];
   NSURLRequest*   request = [NSURLRequest requestWithURL:url];

   NSURLConnection*   connection;
   connection = [NSURLConnection connectionWithRequest:request delegate:self];
   if(connection){
      dl_blocks = Block_copy(handler);
   }
}
```

ブロックスをコピーする

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■NSURLConnectionのデリゲートメソッド

```
//ダウンロード開始。データを受けるNSDataオブジェクトを作成する
- (void)connection:(NSURLConnection *)connection
didReceiveResponse:(NSURLResponse *)response
{
   self.dlData = [[NSMutableData alloc] initWithLength:0];
}
//ダウンロード中。データを受ける
- (void)connection:(NSURLConnection *)connection
{
   didReceiveData:(NSData *)data{
   [self.dlData appendData:data];
}
//ダウンロード完了
- (void)connectionDidFinishLoading:(NSURLConnection *)connection
{
   dl_blocks(self.dlData);
   Block_release(dl_blocks);
   dl_blocks = nil;
   self.dlData = nil;
}
```

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■NSURLConnectionのデリゲートメソッド

```
//ダウンロード開始。データを受けるNSDataオブジェクトを作成する
- (void)connection:(NSURLConnection *)connection
didReceiveResponse:(NSURLResponse *)response
{
   self.dlData = [[NSMutableData alloc] initWithLength:0];
}
//ダウンロード中。データを受ける
- (void)connection:(NSURLConnection *)connection
{
   didReceiveData:(NSData *)data{
   [self.dlData appendData:data];
}
//ダウンロード完了
- (void)connectionDidFinishLoading:(NSURLConnection *)connection
{
   dl_blocks(self.dlData);
   Block_release(dl_blocks);
   dl_blocks = nil;
   self.dlData = nil;
}
```

ブロックスを呼ぶ

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■NSURLConnectionのデリゲートメソッド

```
//ダウンロード開始。データを受けるNSDataオブジェクトを作成する
- (void)connection:(NSURLConnection *)connection
didReceiveResponse:(NSURLResponse *)response
{
   self.dlData = [[NSMutableData alloc] initWithLength:0];
}
//ダウンロード中。データを受ける
- (void)connection:(NSURLConnection *)connection
{
   didReceiveData:(NSData *)data{
   [self.dlData appendData:data];
}
//ダウンロード完了
- (void)connectionDidFinishLoading:(NSURLConnection *)connection
{
   dl_blocks(self.dlData);
   Block_release(dl_blocks);
   dl_blocks = nil;
   self.dlData = nil;
}
```

ブロックスを呼ぶ

コピーしたブロックスをリリースする

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■メソッドの呼び出し

```
-(IBAction)download:(id)sender
{
   [self dlBlocks:@"http://xxxxx/image.png"
          Handler:^(NSData *data) {
             if(data){
                UIImage*  image = [UIImage imageWithData:data];
                imageView.image = image;
             }
          }];
}
```

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■ブロックスをプロパティとして保持する

インターフェース

```
typedef void(^DownloadBlocks)(NSData* data);

@interface ViewController : UIViewController


@property   (nonatomic, retain) NSMutableData *dlData;
@property   (nonatomic, retain) IBOutlet UIImageView *imageView;
@property   (nonatomic, copy) DownloadBlocks copy_blocks;

-(void)dlBlocks:(NSString*)urlPath Handler:(DownloadBlocks)handler;

-(IBAction)download:(id)sender;

@end
```

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■ブロックスをプロパティとして保持する

インターフェース

```
typedef void(^DownloadBlocks)(NSData* data);

@interface ViewController : UIViewController


@property   (nonatomic, retain) NSMutableData *dlData;
@property   (nonatomic, retain) IBOutlet UIImageView *imageView;
@property   (nonatomic, copy) DownloadBlocks copy_blocks;

-(void)dlBlocks:(NSString*)urlPath Handler:(DownloadBlocks)handler;

-(IBAction)download:(id)sender;

@end
```

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■ブロックスをプロパティとして保持する

実装

```
-(void)dlBlocks:(NSString*)urlPath Handler:(DownloadBlocks)handler
{
   NSURL*           url = [NSURL URLWithString:urlPath];
   NSURLRequest*   request = [NSURLRequest requestWithURL:url];

   NSURLConnection*   connection;
   connection = [NSURLConnection connectionWithRequest:request delegate:self];
   if(connection){
       self.copy_blocks = handler;
   }
}
```

ブロックスでiOS 4でも使えるダウンロードメソッドを作ってみる

■ブロックスをプロパティとして保持する

実装

```
-(void)dlBlocks:(NSString*)urlPath Handler:(DownloadBlocks)handler
{
   NSURL*           url = [NSURL URLWithString:urlPath];
   NSURLRequest*   request = [NSURLRequest requestWithURL:url];

   NSURLConnection*   connection;
   connection = [NSURLConnection connectionWithRequest:request delegate:self];
   if(connection){
       self.copy_blocks = handler;
   }
}
```

ブロックスをコピーする

資料など

『ブロックプログラミングトピック』

https://developer.apple.com/jp/devcenter/ios/library/documentation/Blocks.pdf

以上

ありがとうございました。