Validation Usage

This validation framework is not specific to QuickDialog, but it works in well with it.

The validation framework allows you to consistently validate data according to the type of data it is.

There are a number of standard validations:

- Email
- Alphabetic
- Alphanumeric
- Length of input field (Characters)
- Multi string (make sure that if one or more strings are present, others are as well).
- Null
- Number range (between 2 integer values)
- url

These validations are all based on the IMSValidatorProtocol.

It is therefore possible for you to create your own validators that perform tasks which are not checked by the above checks, including checking database for existing data etc.

If you write your own validator, they should be plugged into IMSValidationPlugin. There is sample code in that module to show you how to do that.

Each validator takes either an IMSValidationCheck object or an array of IMSValidationCheck objects.

The validation check consists of the following:

- input - the input data
- fieldName - the description of the field which you are checking
- article - the article (a, the, your etc) which allows you to customise the error message returned.

And optionally

- username

- password.

The standard validations have an associated bundle which has localizable.strings for each error message. I have not localised for any language other than English.

If you wish to have validations which are executed only if a previous validation didn't fail (excuse the negatives), then you need to set the groupName. Typically, you would do this if you want to check for a null value, and if it isn't null, then do another check. Doubling up on Multi field checks is also possible. The validations are performed in sequential order, and if there is a failure and groupName exists, then all the following validations are nto performed if the groupName is the same. Note that the groupName dependency is only relevant while the groupName remains the same, and when it is set to another value including @"". It is possible to use the same groupName in more than one place within the validation list; however you should be circumspect and use unique identifiers where you can.

I typically set up the validation in a manner similar to this:

```
- (NSMutableArray*)setupValidators:(SettingsInfo*)info
{
        NSMutableArray *validators = [[NSMutableArray alloc] init];
        // set up title validator
        IMSNullStringValidator *t = [[IMSNullStringValidator alloc] init];
        IMSEntryValidator *tv = [[IMSEntryValidator alloc] initWithValidation:t
                andCheck:[[IMSValidationCheck alloc] initWithInput:info.title andField:@"title" andArticle:@"your"]];;
        tv.groupName = @"t";
        [validators addObject:tv];
        IMSFieldSizeValidator *t1 = [[IMSFieldSizeValidator alloc] initWithMin:2 andMax:100];
        IMSEntryValidator *tv1 = [[IMSEntryValidator alloc] initWithValidation:t1
                andCheck:[[IMSValidationCheck alloc] initWithInput:info.title andField:@"title" andArticle:@"your"]];
        tv1.groupName = @"t";
        [validators addObject:tv1];

        // set up first name validators
        IMSNullStringValidator *f = [[IMSNullStringValidator alloc] init];
        IMSEntryValidator *fv = [[IMSEntryValidator alloc] initWithValidation:f
                andCheck:[[IMSValidationCheck alloc] initWithInput:info.first andField:@"first name" andArticle:@"your"]];
        fv.groupName = @"fn";
        [validators addObject:fv];
        IMSFieldSizeValidator *f1 = [[IMSFieldSizeValidator alloc] initWithMin:2 andMax:100];
        IMSEntryValidator *fv1 = [[IMSEntryValidator alloc] initWithValidation:f1
                andCheck:[[IMSValidationCheck alloc] initWithInput:info.first andField:@"first name" andArticle:@"your"]];
        fv1.groupName = @"fn";
```

```
[validators addObject:fv1];

// set up last name validators
IMSNullStringValidator *l = [[IMSNullStringValidator alloc] init];
IMSEntryValidator *lv = [[IMSEntryValidator alloc] initWithValidation:l
       andCheck:[[IMSValidationCheck alloc] initWithInput:info.last andField:@"last name" andArticle:@"your"]];
lv.groupName = @"ln";
[validators addObject:lv];
IMSFieldSizeValidator *l1 = [[IMSFieldSizeValidator alloc] initWithMin:2 andMax:100];
IMSEntryValidator *lv1 = [[IMSEntryValidator alloc] initWithValidation:l1
       andCheck:[[IMSValidationCheck alloc] initWithInput:info.last andField:@"last name" andArticle:@"your"]];
lv1.groupName = @"ln";
[validators addObject:lv1];

// set up job title validator
IMSNullStringValidator *j = [[IMSNullStringValidator alloc] init];
IMSEntryValidator *jv = [[IMSEntryValidator alloc] initWithValidation:j
       andCheck:[[IMSValidationCheck alloc] initWithInput:info.job andField:@"job title" andArticle:@"your"]];
jv.groupName = @"jt";
[validators addObject:jv];
IMSFieldSizeValidator *j1 = [[IMSFieldSizeValidator alloc] initWithMin:2 andMax:100];
IMSEntryValidator *jv1 = [[IMSEntryValidator alloc] initWithValidation:j1
       andCheck:[[IMSValidationCheck alloc] initWithInput:info.job andField:@"job title" andArticle:@"your"]];
jv1.groupName = @"jt";
[validators addObject:jv1];

// set up email validator
IMSEmailValidator *ee = [[IMSEmailValidator alloc] init];
IMSEntryValidator *ev = [[IMSEntryValidator alloc] initWithValidation:ee
       andCheck:[[IMSValidationCheck alloc] initWithInput:info.email andField:@"email address" andArticle:@"a"]];
ev.groupName = @"e";
[validators addObject:ev];
IMSEmailValidator *ee1 = [[IMSEmailValidator alloc] init];
IMSEntryValidator *ev1 = [[IMSEntryValidator alloc] initWithValidation:ee1
       andCheck:[[IMSValidationCheck alloc] initWithInput:info.email andField:@"email address" andArticle:@"a valid"]];
ev1.groupName = @"e";
[validators addObject:ev1];

// set up Url validator
IMSUrlValidator *ue = [[IMSUrlValidator alloc] init];
IMSEntryValidator *uv = [[IMSEntryValidator alloc] initWithValidation:ue
       andCheck:[[IMSValidationCheck alloc] initWithInput:info.url andField:@"synchronisation URL" andArticle:@"a valid"]];
[validators addObject:uv];
// set up two validator for password and userid
//QMultiStringValidator
IMSValidationCheck *uvc = [[IMSValidationCheck alloc] initWithInput:info.user andField:@"username" andArticle:@""];
```

```
        IMSValidationCheck *pvc = [[IMSValidationCheck alloc] initWithInput:info.password andField:@"password" andArticle:@""];
        IMSMultiStringValidator *m = [[IMSMultiStringValidator alloc] init];
        NSMutableArray *ma = [[NSMutableArray alloc] initWithObjects:uvc,pvc,nil];
        IMSEntryValidator *mv = [[IMSEntryValidator alloc] initWithValidation:m andChecks:ma];
        mv.groupName = @"uup";
        [validators addObject:mv];
        // set up two validator for url and userid
        //QMultiStringValidator
        IMSValidationCheck *uvc1 = [[IMSValidationCheck alloc] initWithInput:info.url andField:@"url" andArticle:@""];
        IMSValidationCheck *uvc2= [[IMSValidationCheck alloc] initWithInput:info.user andField:@"username" andArticle:@""];
        IMSMultiStringValidator *m1 = [[IMSMultiStringValidator alloc] init];
        NSMutableArray *ma1 = [[NSMutableArray alloc] initWithObjects:uvc1,uvc2,nil];
        IMSEntryValidator *mv1 = [[IMSEntryValidator alloc] initWithValidation:m1 andChecks:ma1];
        mv1.groupName = @"uup";
        [validators addObject:mv1];

        return validators;
    }
```

I would then run the validators as such:

```
    - (void)onDone:(QButtonElement*)buttonElement
    {
    [self loading:NO];
    SettingsInfo *info = [[SettingsInfo alloc] init];
    [self.root fetchValueIntoObject:info];

    //verify page
    NSString* theErrors = [IMSValidation validateEntries:[self setupValidators:info]];

    if (theErrors != nil)
    {
            [YRDropdownViewIMS showDropdownInView:self.quickDialogTableView
                                             type:YRError
                                            title:[mSds.validationBundle localizedStringForKey:@"ErrorValidation" value:@"ErrorValidation" table:nil]
                                           detail:theErrors
                                            image:[UIImage imageNamed:@"Alert.png"]
                                         animated:YES
                                        hideAfter:2.0f];


    }
    else
    {
```

I have customised the YRDropdownView (YRDropdownViewIMS) which hooks very nicely into any view so that messages can be displayed. It is not required by the validation framework.