بسم الله الرحمن الرحيم

underQL

دليل الإستخدام

الإصدار ١.٠.٠ بيتا

عبدالله عيد المحمادي

cs.abdullah@hotmail.com

إن هذا المشروع مجاني وبإمكانك المشاركة في تطويره بشتى الطرق فإن كنت مبرمجاً فمرحباً بك معنا في عملية التطوير وإن كنت مصمماً فنحن نقدر تصاميمك بأن تعمل معنا في كل مايخص أمور التصميم وإن كنت محرراً فنحن نحتاجك في كتابة المستندات الخاصة بأدلة الإستخدام وإن كنت لا تجيد كل تلك الأشياء فمرحباً بك ناقداً ومعلقاً ويسرنا أن ترسل لنا ملاحظاتك وستكون على الرحب والسعى فيد الله مع الجماعة فقط كل ماعليك فعله هو زيارة الموقع الرسمي لـ underQL على الإنترنت على www.underql.com فأهلاً ومرحباً بك.

تم تحريره ٥ – ٩ – ١٤٣٢ هـ الساعة ١٧- ٨ صباحاً

ماهو underQL ؟

ببساطة فإن underQL هو عبارة عن كلاس PHP يسهل لك التعامل مع قواعد بيانات MySQL بحيث يسهل عليك التعامل مع الأربع العمليات الأساسية التي يستخدمها المبرمجون في برمجياتهم بكثرة ألا وهي SELECT, INSERT, UPDATE, DELETE بحيث تتمكن من كتابتها بسرعة عالية وبطريقة كتابتك لشفرات بي اتش بي العادية ويمكنك إجراء الإستعلامات التي كانت تأخذ منك عشرات الأسطر في سطر واحد أو سطرين.

إن مايميز underQL هو قدرتها على تقليل الوقت المستهلك في كتابة الشفرة البرمجية التي تتعامل مع قواعد بيانات MySQL بحيث تصبح النسبة التي تساعد فيها underQL على تقليل كتابة الشفرة تتراوح مابين ٤٠ إلى ٦٠ بالمئة أي بكلام أخر فإن البرنامج الذي يأخذ بالطريقة العادية ١٠ أيام فإنه سيأخذ من ٤ – ٦ أيام بإستخدام underQL .

توجد نسختين من underQL وهما single و multi فالأولى تتكون من ملف واحد فقط هو underQL.php وهي تفضل لمن يريد أن يستخدمها مباشرة في الكود أما نسخة multi فهي نفس الشفرة الموجودة في single ولكنها مقسمة إلى مجموعة من الملفات وهي تفيد من يريد التطوير على underQL ، لذلك يفضل أن تقوم بإنزال النسخة single في حالة أردت إستخدام المكتبة كما هي ولاتريد التعديل والتطوير عليها .

بالنسبة لتسمية المكتبة فبما أنها تتعامل مع SQL في الأساس فإن الحرفين QL أتت من SQL وأما بالنسبة لكلمة under فكما هو معروف فإن PHP تسمح بتسمية المتغيرات بإستخدام underscore ولذلك تم عمل كائن بإسم $_ وهو يمثل كائن underQL الأساسي وهو معرف تلقائياً مع underQL ويمكنك إستخدامه مباشرة ومن هنا تم أخذ كلمة under وأما سبب حذف حرف الـ S وعدم جعله underSQL هو وجود علامة $ في اسم الكائن $_ وبهذا الشكل تكون اسم الكلاس underQL.

رخصة الإستخدام

إن underQL تحت ترخيص MPL (Mozilla Public License) وهو مفتوح المصدر ويمكنك إستخدامه والتعديل عليه وتطويره ونشره بإسم أخر كيفما تقتضيه حاجتك.

كائن $_

هناك كائن اسمه $_ ومنه جاءت تسمية المكتبة وهو جاهز للإستخدام بمجرد أن تقوم بتضيمن المكتبة بإستخدام include وبإمكانك التخلي عنه وإستخدام كائن جديد بإسم خاص فيك ولقد تم وضع هذا الكائن للإختصار وأيضًا يمكنك استخدام اسم اخر وهو $underQL كمتغير أيضاً وهو يؤدي نفس الغرض لأنه يشير لنفس الكائن $_ . بكلام أخر فإن هذا الكائن موجود لتسريع عملية الكتابة لأن اسمه مختصر جداً.

ملاحظة

يمكنك استخدام كائن underQL واحد لجميع الجداول بحيث تقوم بتغيير الجدول عن طريق الدالة table في كل مرة تريد التعامل فيها مع جدول معين .

لمزيد من المعلومات نرجوا مراجعة الموقع الرسمي على www.underql.com أو يمكنك الحصول على أخر النسخ المحدثة عن طريق http://code.google.com/p/underql .

استعلام SELECT والحصول على البيانات

سنتفرض أن لدينا جدول للمهام بإسم `tasks` ويحتوي الجدول على ثلاثة حقول وهي : `id`, `title`, `description` وهي لرقم المهمة وعنوان المهمة ووصف المهمة على التوالي ولو افترضنا أن الجدول يحتوي على عدد من البيانات فإننا في حال أردنا إستخراج البيانات من الجدول أو بكلام أخر إستخراج كل البيانات فإننا سنكتب مايلي :

```php
<?php

 include('underQL.php');
 $_('tasks');
 $_->fetch();
for($i = 0; $i < $_->count(); $i++)
 {
   echo $_->title;
   echo '<br />';
   echo $_->description;
   echo '<br />';
   $_->fetch();
 }

?>
```

وظيفة الدالة `fetch` هو جلب الصف التالي كما هو واضح وفي حال انتهت الصفوف فإنها سترجع القيمة `null` ولكن بما أننا استخدمنا `count` فهذا سيقف بعد أخر صف.

تحديد الحقول

ربما يتبادر إلى ذهنك السؤال التالي : كيف أحدد الحقول التي أريدها في الإستعلام وببساطة أقول لك أنك لن تحتاج إلى تعديل الكثير فالمكتبة وجدت بإذن الله لكي تسهل عليك الأمور البرمجية ولذلك دعنا نفترض أنك تريد تحديد عمودين وهما رقم المهمة وعنوانها `id`, `title` فعندها يصبح المثال الموجود في الأعلى كالتالي :

```php
<?php
 include('underQL.php');
 $_('tasks','id,title');
 $_->fetch();
for($i = 0; $i < $_->count(); $i++)
 {
   echo $_->id;
   echo '<br />';
   echo $_->title;
   echo '<br />';
   $_->fetch();
 }
?>
```

وكما ترى فإن دالة count تحضر لنا عدد السجلات من أخر استعلام قمنا بتنفيذه.

الشروط والمعلومات الإضافية الأخرى

الآن ربما تفكر في إضافة شرط أو ربما تريد ترتيب الصفوف أو اضافة SQL على تعليمة جلب البيانات وعندها نقول لك لاتقلق فإنه بإمكانك وضع شرط أو وضع ماتريد وذلك بعد البارامتر الخاص بتحديد اسماء الحقول وكملاحظة فإنك في حالة أردت وضع جميع الحقول فكل ماعليك وضعه هو علامة النجمة * مكان أسماء الحقول وسيتم وضع جميع الحقول بشكل تلقائي وللتوضيح أكثر دعنا نفترض أننا نحاول جلب المهمة رقم ١٥ عندها يكون المثال كالتالي :

```
<?php

 include('underQL.php');
 $_('tasks','*','WHERE id = 15');
 $_->fetch();
 echo $_->title;
 echo '<br />';
 echo $_->description;

?>
```

دالة select

يحتوي underQL على دالة بإسم select وهي تعمل نفس عمل المثال السابق غير انها لاتحتوي على اسم الجدول وهي تفترض أن أسم الجدول تم تحديده بإستخدام الدالة table ولكتابة نفس المثال الموجود بالأعلى بواسطة دالة select فإن المثال سيكون كالتالي :

```
<?php

 include('underQL.php');
 $_->table('tasks');
 $_->select('*','WHERE id = 15');
 $_->fetch();
 echo $_->title;
 echo '<br />';
 echo $_->description;

?>
```

كائن underQL جديد

يمكنك استخدام كائن جديد وذلك لكي تخصصه للجدول دون أن تستخدم الكائن $_ بكلام أخر لو أردنا أن نقوم بإنشاء كائن جديد بنفس إسم الجدول فإننا نقوم بالتالي :

```php
<?php

include('underQL.php');
$tasks = new underQL('tasks');
$tasks->select('*','WHERE id = 15');
$tasks->fetch();
echo $tasks->title;
echo '<br />';
echo $tasks->description;

?>
```

لاحظ الآن وبما أنك أنشأت كائن جديد بإسم `$tasks` وقمت بتحديد إسم الجدول من البداية في الأعلى فإن الكائن أصبح يمثل الجدول `tasks` . يمكنك من خلال هذا الأسلوب أن تنشئ كائنات بعدد الجداول الموجودة عندك بحيث يمثل كل كائن جدول وهذا سيسهل عليك التعامل مع العلاقات بين الجداول .

إدخال البيانات

يوفر لك `underQL` طريقة سريعة لإدخال البيانات وكل ماعليك فعله فقط هو وضع البيانات لأنه سيتعرف على أسماء الحقول تلقائياً ولتوضيح عملية الإدخال إلى الجدول `tasks` لاحظ المثال التالي :

```php
<?php

include('underQL.php');
$tasks = new underQL('tasks');
$tasks->id = 433;
$tasks->title = 'My Task';
$tasks->description = 'This is a simple task';
$tasks->insert();

?>
```

الآن كما تلاحظ فكل ما فعلناه هو فقط التعامل مع أسماء حقول الجدول وكأنها متغيرات أي بكلام أخر فإن كل مافعلته هنا هو وضع القيم بشكل مباشر ومن ثم استدعاء دالة `insert` لتقوم بحفظ البيانات وهذا كل شئ مع ملاحظة أنه يمكنك حذف السطر الذي يحتوي على المتغير `id` وذلك لأنه `auto increment` أي يتم وضع قيمته بشكل تلقائي .

تعديل البيانات

يوفر لك underQL طريقة لتعديل البيانات ولاتختلف عن الإدخال إلا بشئ بسيط جداً فلو أخذنا المثال الموجود في الأعلى وعدلنا عليه لينفذ عملية التعديل ، فإننا لو فرضنا أننا نريد تعديل المهمة رقم ٧ فإننا سنقوم بالتالي .

```php
<?php

include('underQL.php');
$tasks = new underQL('tasks');
$tasks->title = 'My Task';
$tasks->description = 'This is a simple task';
$tasks->update('id = 7');

?>
```

هذا كل شئ ، نعم فالعملية بسيطة ضع القيم وحدد الصف الذي تريد تعديله كشرط بداخل دالة update وعندها سيتم التعديل ويمكنك ترك القيمة خالية في الدالة update وفي هذه الحالة سيتم تعديل كامل الجدول بالبيانات التي وضعتها إن لم تحدد شرط محدد لدالة update .

حذف البيانات

أعتقد أنك قمت بالتخمين فدالة الحذف هي أبسط دالة ويمكنك حذف السجلات جميعها أو تحديد شرط معين بإستخدام دالة delete فلو أردنا مثلاً حذف السجل رقم ١١ عندها نقوم بالتالي .

```php
<?php

include('underQL.php');
$tasks = new underQL('tasks');
$tasks->delete('id = 11');

?>
```

بهذا الأمر يتم الحذف مع التحذير أنه في حال استدعيت delete بدون أن تمرر لها أي قيمة فإنه سيتم حذف جميع البيانات الموجودة في الجدول.

العلاقات بين الجداول

قد تبادر إلى ذهنك كيفية التعامل مع العلاقات بين الجدوال و underQL يقدمها بطريقة سهلة جداً فلو فرضنا أن لديك جدولين احدهما للمستخدمين والآن جدول التدوينات مثلاً وكانت أسماء الجداول هي users, blogs وكنا نريد إحضار جميع التدوينات التي تخص مستخدم معين حيث سنفترض أن المستخدم الحالي للموقع رقمه هو ١٥ وأننا قد تعاملنا مع من قبل بواسطة كائن $user .

```
<?php

include('underQL.php');
$users = new underQL('users');
$blogs = new underQL('blogs');
$user->select('*','WHERE id = 15');
$user->fetch();
…
$blogs->select('*','WHERE uid = '.$user->id);
…

?>
```

في البرامج الواقعية ربما لن تكون العلاقات بهذا الشكل ولكن مهما كانت العلاقات فإنك تملك الجداول على شكل كائنات من الأساس ويمكنك بناء أي علاقات بينهما بكل يسر وسهولة ودون أي مشاكل بإذن الله تعالى .

القواعد Rules

يوفر underQL نظام خاص بوضع القواعد بحيث يمكنك وضع القواعد والقيود على البيانات التي يتم إدخالها بحيث تضع قواعدك الخاصة مرة واحدة فقط وسيقوم underQL بتطبيق تلك القواعد كلما تم ادخال أو تعديل البيانات. لكي تكون البرمجة منظمة فإن هناك كائن خاص بالقواعد اسمه UQLRule كما يلي شرحه.

كائن UQLRule

وظيفة هذا الكائن هي تخزين القواعد الخاصة بك لجدول معين أي بكلام أخر فإن لكل جدول سيكون كائن UQLRule يحدد قواعد ذلك الجدول ولست ملزماً بإنشاء القواعد لأنك تستطيع تعديل وإدخال البيانات بدون نظام القواعد ولكن ستجد نفسك مجبراً عليها في البرمجيات الحقيقية لأنك في الغالب تقوم بوضع الكثير من الشروط أثناء كتابة برامجك وعلى أية حال فإنه تجدر الإشارة هنا إلى أنه يجب عليك وضع القواعد قبل التعامل مع كائن underQL والسبب هو أنه يجب عليك أن تربط كائن UQLRule بعد الإنتهاء منه بكائن underQL الخاص بنفس الجدول وللتوضيع أكثر فإننا سنقوم بعمل كائن جديد يمثل القواعد الخاصة بجدول tasks كالتالي :

```php
<?php

 include('underQL.php');

 $tasks_rules = new UQLRule('tasks');
 $tasks_rules->title('length',15);
 $tasks_rules->title('required');
 $tasks_rules->description('required');

?>
```

كما ترى فإن طريقة وضع القواعد بسيطة جداً وهي كالتالي ، تقوم أنت بتعريف كائن جديد للقواعد UQLRule وتعطيه اسم الجدول الذي تريد وضع القواعد له وفي هذه الحالة وضعنا الجدول tasks بعدها سيتعرف الكائن تلقائياً على أسماء الحقول لديك وببساطة تقوم أنت بإستخدام أسماء الحقول كأنها دوال، فتصبح القواعد على شكل دوال أي إذا أردت أن تضع قاعدة لحقل معين فقط استدعيه كأنه دالة وقم بتمرير اسم القاعدة كأول باراميتر ويمكنك تكرار الإستدعاء إذا أردت وضع أكثر من قاعدة على نفس الحقل كما فعلنا مع title هنا وسنأتي على أسماء القواعد لكن كتوضيح فإننا وضعنا قاعدة length وهي تختص بالطول وعندها لن يقبل underQL وضع قيمة أكبر من ١٥ في حقل العنوان وقاعدة required تعني أنه مطلوب ولايمكن أن تترك قيمته خالية وهكذا لـ description .

يوجد الكثير من القواعد كما ذكرنا وسنسردها بإذن الله لنبين معنى كل قاعدة وطريقة استخدامها ولكن الآن مايهمنا هو أن تلاحظ أن كائن القواعد منفصل كلياً عن كائن underQL لذلك يجب هنا بعد تعريف القواعد أن نقوم بإنشاء كائن underQL ومن ثم نقوم بالربط بينهما بإستخدام الدالة attachRule كالتالي :

```php
<?php

 include('underQL.php');

 $tasks_rules = new UQLRule('tasks');
 $tasks_rules->title('length',15);
 $tasks_rules->title('required');
 $tasks_rules->description('required');

 $tasks = new underQL('tasks');
 $tasks->attachRule($tasks_rules);

?>
```

ببساطة قمنا بتمرير إسم كائن القواعد `$tasks_rules` إلى الدالة `attachRule` وبهذه الطريقة تم ربط القواعد بكائن `underQL` وسيتم تطبيقها بمجرد محاولتك لإدخال أو تعديل أي بيانات على جدول `tasks` . يجب التنبيه هنا إلى أن القواعد يتم كتابتها مرة واحدة فقط هي وكائن `underQL` الخاص بجدول معين ومن ثم يتم استخدامهما بشكل مباشر.

لو قمنا الآن بعمل إدخال للبيانات أو تعديل فإن القيم التي سيتم إدخالها أو تعديلها سيتم تطبيق القواعد عليها مباشرة وبقي عليك أن تعرف فيما إذا كانت القيم المدخلة قد طابقت القواعد أم لا ويمكنك ذلك بسهولة عن طريقة دالة `isRulesPassed` وهي دالة ترجع `true` في حال أن القيم المدخلة طابقت جميع القواعد وإذا لم تكن القيم مطابقة للقواعد فإن الدالة سترجع `false` . عندما يتم إرجاع `false` من دالة `isRulesPassed` فإن `underQL` يقوم بإرجاع رسالة الخطأ عن طريق الدالة `getRuleError` والتي ترجع لك رسالة نصية تحدد نوع الخطأ ويمكنك الإستفادة منها بعرضها على المستخدم ويجدر التنبيه هنا إلى أنه وعلى الرغم من عدم مطابقة القواعد فإن الإدخال أو التعديل سيستمر إذا لم تقم بإيقافه أنت أي بكلام أخر فإن `underQL` سيعطيك تنبيه على أن القواعد غير مطابقة وأنت من يتخذ إجراء الإيقاف وللتوضيح أكثر لاحظ المثال التالي والذي يشتمل على جميع ماذكر .

```php
<?php

 include('underQL.php');
 $tasks_rules = new UQLRule('tasks');
 $tasks_rules->title('length',15);
 $tasks_rules->title('required');
 $tasks_rules->description('required');
 $tasks = new underQL('tasks');
 $tasks->attachRule($tasks_rules);
 $tasks->id = 433;
 $tasks->title = 'My Task';
 $tasks->description = 'This is a simple task';

 if(!$tasks->isRulesPassed());
  die($tasks->getRuleError());
 $tasks->insert();

?>
```

كما تلاحظ فإنه بمجرد وضعك للقيم فإن underQL سيقوم بالتشييك على القيم وفقاً للقواعد التي وضعتها ولقد قمنا بوضع جملة if لكي نتأكد من وجود الخطأ وحددنا له أنه في حالة لم تتطابق القواعد فإننا سنوقف السكربت ونعرض رسالة الخطأ كما ترى بإستخدام die .

أود التنبيه هنا إلى أن السكربت في الأساس سيكون صغير وذلك لأننا ننصح دائماً بعمل ملف خارجي مثلاً بإسم myrules.php ووضع جميع قواعدك التي تخص جميع جداولك ومن ثم تقوم فقط بإستدعائه بإستخدام include وبهذا يصبح الشكل الواقعي للمثال الموجود بالأعلى على فرض أنك وضعت ملف underQL.php بداخل ملف myrules.php كالتالي :

```
<?php

 include('myrules.php');
 $tasks->id = 433;
 $tasks->title = 'My Task';
 $tasks->description = 'This is a simple task';
 if(!$tasks->isRulesPassed());
  die($tasks->getRuleError());
 $tasks->insert();

?>
```

في بعض الأحيان ربما تود إيقاف القواعد مؤقتاً ففي كثير من الأوقات ربما نحتاج إلى إدخال البيانات بدون وضع القيود وهذا قد يحدث لذلك فإن underQL توفر لك دالة اسمها detachRule وهي عكس الدالة attachRule حيث تقوم الدالة بفصل القواعد الخاصة بجدول معين عن كائن underQL وبذلك يمكنك إدخال البيانات بدون قيود القواعد وذلك لأنه تم إزالتها مؤقتاً ويمكنك بعد ذلك إرجاعاها بإستخدام الدالة attachRule بنفس الطريقة وللتوضيح لاحظ المثال التالي .

```
<?php

include('myrules.php');
$tasks->detachRule('tasks');
$tasks->id = 433;
$tasks->title = 'My Task';
$tasks->description = 'This is a simple task';
$tasks->insert();
$tasks->attachRule($tasks_rules);

?>
```

الآن فإن القيم المدخلة لن يتم التشييك عليها ولكن يجب الإنتبها إلى أنه أننا استخدمنا دالة attachRule بداخل ملف myrules.php وبهذا الأسلوب يمكنك إيقاف القواعد مؤقتاً ويجدر بنا هنا الإنتباه إلى أنه يجب عليك تمرير إسم الجدول كنص لدالة detachRule وذلك لكي يزيل القواعد الخاصة به وليس تمرير الكائن مثلما تفعل مع attachRule .

أنواع القواعد

فيما يلي قائمة بأسماء القواعد التي يمكنك إستخدامها وسنضع أسماء حقول إفتراضية للتوضيح مع التنبيه أن هذه القائمة تتغير من وقت لأخر وذلك لأن القواعد يتم إضافتها بشكل مستمر حسبما تقتضيه الحاجة لذلك نرجوا الإنتباه ومتابعة دليل الإستخدام بشكل مستمر وتحديثه أولاً بأول.

required

وهي تعني أن القيمة مطلوبة ولايمكن تركها خالية ويجب عليك وضع قيمة مناسبة.

```
<?php
 $tasks->title('required');
?>
```

length

وهي القاعدة الخاصة بتحديد طول القيمة التي يجب إدخالها ويجب أن لاتتجاوز القيمة التي وضعتها .

```
<?php
$tasks->title('length',25);
?>
```

هنا قمنا بتحديد أقصى طول على أنه ٢٥ وفي حالة تم تجاوز هذا الحد للعنوان الذي تم إدخاله فإن underQL ستعيد لك رسالة الخطأ بسبب عدم مطابقة القواعد .

number

وهي تعني أن القيمة المدخلة يجب أن تكون أرقاماً فقط.

```
<?php
 $tasks->id('number');
?>
```

symbol

وهي تعني أن القيمة المدخلة يجب أن تكون رموزاً فقط مثل ^%$#@! ولاتحتوي على أرقام وحروف.

```
<?php
 $user->password('symbol');
?>
```

وهنا قمنا بإجبار المستخدم على أن يضع رموز في مكان كلمة المرور للجدول `user` وربما يمكنك الإستفادة منها في أمور أخرى غير كلمة المرور حسبما تقتضيه الحاجة.

hex

وهي تعني أن القيمة المدخلة يجب أن تكون بنظام الـ `Hexadecimal` وهو النظام الست عشري والذي تكون قيمته من صفر إلى ٩ ومن A إلى F صغيرة أو كبيرة (a إلى f).

```
<?php
 $theme->color('hex');
?>
```

كما تلاحظ في المثال فإن أفضل شئ لإستخدام النظام الست عشري هو الألوان فالألوان في `HTML`, `CSS` يتم وضعها بالأسماء وبالنظام الست عشري لذلك في المثال قمنا بتحديد قيمة الحقل `color` على أن لا تقبل إلا قيمة `Hexadecimal` .

alpha

وهي تعني أن القيمة يجب أن تكون حروف فقط (إنجليزية) فهناك مشكلة أحياناً في التعامل مع الحروف العربية ولذلك هذه القاعدة تعني أنه يجب أن تكون حروفاً هجائية إنجليزية فقط أي لا يوجد أرقام ولا رموز.

```
<?php
 $tasks->title('alpha');
?>
```

وهذا يعني أن أي شئ غير حروف الهجاء الإنجليزية يعتبر خطأ وغير مطابق للقواعد وهذا يفيدنا ربما في حالة أردنا كتابة حقل معين باللغة الإنجليزية مثل الإسم بالإنجليزي.

alphanum

وهي تعني أن القيمة يجب أن تكون حروف (إنجليزية) وأرقام فهناك مشكلة أحياناً في التعامل مع الحروف العربية ولذلك هذه القاعدة تعني أنه يجب أن تكون حروفاً هجائية إنجليزية وأرقام أي خليط منهما أي لا يوجد رموز .

```
<?php
$user->nickname('alphanum');
?>
```

وهذا يعني أن أي شئ غير حروف الهجاء الإنجليزية أو الأرقام يعتبر خطأ وغير مطابق للقواعد وهذا يفيدنا ربما في حالة أردنا كتابة حقل معين باللغة الإنجليزية مثل اسم المستخدم أو كلمة المرور.

between

وهي تشبه `length` وتختص بطول القيمة ولكن الفرق بينهما هو أن `between` تحدد مدى معين.

```
<?php
$tasks->title('between',4,10);
?>
```

في المثال حددنا أصغر قيمة وأكبر قيمة وهذا يعني أن العنوان يجب أن لا يقل عن ٤ ولا يزيد عن ١٠ ولقاعدة `between` خيارات متعددة فلو قمنا بوضع القيمة صفر في مكان أصغر قيمة كالتالي

```
<?php
$tasks->title('between',0,10);
?>
```

فإن هذا يعني أن لاتزيد عن ١٠ وهنا في هذا الحالة تصبح مطابقة لقاعدة `length` ولو قمنا بعكس العملية بوضع القيمة صفر مكان أكبر قيمة كالتالي

```
<?php
$tasks->title('between',5,0);
?>
```

فهذا يعني أنه الطول مفتوح ولك لايقل عن ٥ . ويجب الملاحظة هنا إلى أنه في حالة مررت قيم أخرى سالبة أو صفر وصفر فإنه لن يتم تطبيق القاعد وسيتم تجاهلها .

الفلاتر filters

تقدم underQL مفهوم الفلاتر أثناء تعاملها مع البيانات ومن الإسم يتضح جلياً أن الفلتر هو عبارة عندالة تقوم بعمل شئ معين على البيانات قبل دخولها إلى الجدول أو خروجها منه ويمكنك عمل فلتر لحقل واحد أو مجموعة حقول دفعة واحدة ولتطبيق أي فلتر فإن underQL توفر كل الدالة filter وهي كالتالي

```
<?php

 $tasks->filter($filter_name,$filter_dir,$fields…);

?>
```

الباراميتر الأول $filter_name يمثل اسم الفلتر وسنأتي عليه بإذن الله وأما الباراميتر الثاني فيحدد إتجاه الفلتر والمقصود بذلك هو هل تريد تطبيق الفلتر أثناء إدخال البيانات وفي هذه الحالة سيتم تطبيقه قبل عمليات الإدخال والتعديل أم تريد تطبيق الفلتر أثناء خروج البيانات من الجدول وفي هذه الحالة سيتم تطبيق الفلتر أثناء الإستعلام وإظهار البيانات وهو يأخذ أحد قيمتين لتحديد الإتجاه وهما UQL_FILTER_IN وهي تعني في حالة الإدخال والتعديل و UQL_FILTER_OUT وهي تعني في حالة إخراج وإستعراض البيانات ، بكلام أخر في حالة عرض البيانات بإستخدام SELECT .

وقبل التوضيح فإن underQL تحتوي حتى الآن على فلترين فقط وجاري تطوير العديد من الفلاتر وفي حال كان لديك إقتراح لفلتر معين فنرجوا إرساله لنا والفلترين هما xss حيث يقوم هذا الفلتر بالتشييك على القيمة فيما إذا كانت تحتوي على شفرات مؤذية وتنظيفها والأخر هو html وهو يقوم بمسح جميع وسوم html من القيمة ويفضل إستخدامها في عمليات الإدخال وليس الإخراج .

تجدر الإشارة هنا إلى أنه يجب عليك عمل الفلتر قبل استخدام دالة insert أو update أو select بمختلف طرقها ويمكنك وضع الفلاتر مع ملف myrules.php وبهذا الأسلوب فإنك تقوم بوضع مختلف الفلاتر هناك ولن تضطر لكتابتها مرة أخرى ، وللتوضيح لاحظ المثال التالي .

```
<?php

 include('myrules.php');
 $tasks->filter('xss',UQL_FILTER_IN,'title','description');
 $tasks->id = 433;
 $tasks->title = 'My Task';
 $tasks->description = 'This is a simple task';
 $tasks->insert();

?>
```

ببساطة مررنا اسم الفلتر وهو xss وقلنا طبق الفلتر في عملية الإدخال فقط ولو أردنا تطبيقه في عمليتي الإدخال والإخراج فسيكون الكود كالتالي

```php
<?php

 include('myrules.php');
 $tasks->filter('xss',UQL_FILTER_IN,'title','description');
 $tasks->filter('xss',UQL_FILTER_OUT,'title','description');
 $tasks->id = 433;
 $tasks->title = 'My Task';
 $tasks->description = 'This is a simple task';
 $tasks->insert();

?>
```

وبهذه الطريقة سيتم تطبيق الفلتر في حالتي الإدخال والإخراج ويجدر التنبيه هنا إلى أنه يمكنك وضع أي عدد تريده من الحقول ولست ملزماً بوضع جميع الحقول ولكن على الأقل يجب وضع حقل واحد مع الفلتر.

وبالنسبة لفلتر `html` فإنه بنفس المبدأ فجميع الفلاتر تطبق بنفس المبدأ فقط قم بتغيير كلمة `xss` إلى `html` وسيتم تطبيق فلتر `html` وإذا أردت الفلترين معاً فقم بعمل فلتر لكل منهما بشكل منفصل.

تنبيه

هناك فلتر خاص بفحص `SQL injection` وهو موجود بشكل افتراضي مع `underQL` ويقوم بالتشييك نيابة عنك في كل عملية إدخال أو تعديل ولاتحتاج لعمل فلتر خاص لهذا الموضوع .

فحص القيم checker

يقدم لك underQL مايسمى checker وهو بنفس فكرة الفلتر ولك يختلف عن في أنه يعيد إما true أو false وذلك للتأكد من شئ معين ولايقوم بإجراء أي عمل معين فقط يرجع للمبرمج القيمة لكي يتخذ المبرمج قراره بناء على القيمة التي أرجعها له الـ checker وتستخدم في حالة الإدخال والتعديل ويجب إستخدامها بعد إدخال القيم في المتغيرات .

يوجد حتى الآن checker واحد ضمن underQL وهو الخاص بالبريد الإلكتروني ووظيفته هو التأكد من أن القيمة المدخلة هي صيغة بريد إلكتروني أو لا ويجب التنبيه هنا إلى أن الـ checker يختلف عن الفتلر في قبوله لأسماء حقول متعددة فهو يقبل فقط اسم حقل واحد في كل مرة وصيغته كالتالي .

```
<?php

 $tasks->checker($checker_name,$field);

?>
```

حيث يمثل $checker_name اسم الـ checker ويمثل $field إسم الحقل الذي تريد فحصه والتأكد منه وللتوضيح لاحظ المثال التالي .

```
<?php

 include('myrules.php');
 $tasks->id = 433;
 $tasks->title = 'My Task';
 $tasks->description = 'This is a simple task';

 if(!$tasks->checker('email','title'))
  die('It is not email..!');

 $tasks->insert();

?>
```

ببساطة قمنا بوضع الـ checker ضمن if وذلك لأنها تعيد إما true أو false ووضعنا إسم الـ checker وهو email والحقل title.

تنبيه

نعيد التنبيه هنا إلى أنه يجب وضع الـ checker بعد وضع القيم كما فعلنا في المثال الموجود بالأعلى ولايهم إن كانت الحالة insert أو update .

في حالة كانت لديك أفكار لعمل `checker` معين فيسرنا أن ترسل لنا الفكرة لتطبيقها وتوزيعها مع الإصدارات القادمة.

الخاتمة

ستكون هناك إن شاء الله نسختين من دليل الإستخدام وهما هذه النسخة الخاصة بالمستخدم النهائي وجاري العمل على نسخة المطورين لتوضيح بعض الدوال والكلاسات لكي يستطيعوا تطوير الفلاتر والقواعد الخاصة بهم وعمل checkers خاص بهم بكل يسر وسهولة إن شاء الله تعالى وهذا مالدي الآن فإن أخطأت فمن نفسي والشيطان وإن أصبت فمن الله عز وجل ولاتنسى إرسال المشاكل التي واجهتك على موقع المشروع ونود التنبيه إلى أن دليل الإستخدام هذا يتم تحديثه بإستمرار فنرجوا أن تبقى متابع للتحديثات المستمرة وذلك بإنزال أخر الإصدارات من أدلة الإستخدام وصلى الله على سيدنا محمد على آله وصحبه أجميعن .