Tracker Field Validation

jQuery Ajax validation

In **Tiki6**, jQuery Ajax validation was introduced. It works perfectly with fields that take text input but is not really suited or relevant to fields that have pre-configured selections, etc. (drop downs).

The Validation feature, under jQuery plugins and add-ons settings in the Features → Interface panel has to be enabled for this to work.

When editing or adding a tracker field while configuring a tracker, you can choose a validation option. As of **Tiki12**, the following validation procedures have been added:

- **captcha**: Checks if the input text matches an antibot-captcha.
- **distinct**: Checks if the input text has been used before in the tracker for this field - thus ensuring all tracker items can be unique with respect to the validated field
- **pagename**: Checks if the input text matches an existing page name
- **username**: Checks if username has been used already and is valid according to the rules configured in Tiki
- **password**: Checks if the password is valid according to the rules configured in Tiki
- **regex**: checks the entered text against any perl regular expression. Enter the regular expression as the "Validation parameter", (Do not include the starting and ending "/" enclosing the regular expression. These will be added automatically.)

Not all validation methods require a "Validation parameter" to be set.

Regardless of which validation method is selected, you can specify a default custom error message if desired, in "Validation error message:"

Adding new custom validation procedures

Developers can add new custom validation procedures in lib/validators. Each validator must be stored in a php file named "validator_nameofvalidator.php" and there must be a main function in the file with the general structure as follows:
function validator_nameofvalidator($input, $parameter = '', $message = '') {
    ...
    ...
    if ($error) {
        return tra($message); // error message to be shown
    }
    return true; // successful validation
}

It is also necessary to add a line in file `lib/core/Services/Tracker/Controller.php` in the list of all validators

```
lib/core/Services/Tracker/Controller.php
'regex' => tr('Regular Expression (Pattern)'),
    'username' => tr('User Name'),
    'nameofvalidator' => tr('My Validator I Wrote by Myself'),
```

$parameter will be what is configured as the "Validation parameter:" in the tracker field settings, and $message is what is configured as the "Validation error message:" in the tracker field settings. You don't have to return $message as the error message if you don't want to (you can return any error message you want depending on the type of the error it is).

### Adding custom per-page validation

Since Tiki 9.1, it is possible to add custom validation in a wiki page which contains a Tracker plugin through jQuery. If the jQuery validation fails, it should set:

```
nosubmitItemForm1 = true;
```

for the first tracker plugin, `nosubmitItemForm2` for the second, etc.

It is necessary that this jQuery plugin is above the Tracker plugin.

Example validation for a field of type "ins_fill" (Multiple Fill Fields introduced in Tiki 9.0) which should start with a number:

```{JQ()}
nosubmitItemForm1 = false;
$('#editItemForm1').submit(function() {
    //alert('départ');
    var valeurs = $('#ins_fill').val();
    //alert('valeurs ' + valeurs);
    var listeValeurs = valeurs.split("\n");
    var $errorCount = 0;
    var $errorDots = "<span id='errorfill' class='errorfill' style='white-space: nowrap; float: left '>"; // build a vertical string for pointing lines in error (optional)
    var $validPattern = /(^\$*\d+\$*),|(^\$*\$)/; // this pattern ensures the first item is a number (comma separated fields)
    for (var i in listeValeurs) {
        //alert(listeValeurs[i]);
        if ( listeValeurs[i].search($validPattern) != -1 ) {
            //alert("'"+listeValeurs[i]+"' est valide");
```
$errorDots += "<br />");
} else {
  $errorCount++;  
  $errorDots += "× <br />");
}

$('#errorfill').remove();
$('label.errorspiral').remove(); // Let's not interfere with the error class which
is controlled by jQuery's validation feature
$('label.errorspiral').css({'width':'99%'});
if( $errorCount == 0 ) { // C'est bon
  //alert('good');
  nosubmitItemForm1 = false;
} else {
  nosubmitItemForm1 = true;
$('#ins_fill').before($errorDots+"</span>").css({'width':'90%'}).addClass('errorspiral');
$('label.errorspiral').css({'float':'left'});
alert("ERREUR" + "\n" + "La bonne syntaxe est : nombre,critère"+"\n"+"Nombre
de lignes fautives : " + $errorCount);
}
if( valeurs == '' ) { // Do our own detection of required field so we can have
translated messages
  nosubmitItemForm1 = true;
  $('#ins_fill').addClass('errorspiral').after('<label class="errorspiral" for="ins_fill" generated="true">' + tr('This field is required') + '</label>');
  nosubmitItemForm1 = true;
}
return false;
});