カスタムチケット属性

Trac
ではチケットにユーザ定義の属性を追加できます。カスタムチケット属性を使用すると、型付けされた、プロジェクト特有のプロパティをチケットに持たせることか

設定方法

カスタムチケット属性を設定するためには、 trac.ini ファイルを変更します。カスタムフィールドは、 trac.ini ファイルの [ticket-custom] セクションに書く必要があります。

各属性の定義は以下のように記述します:

```
 ■■■ = ■■■
(■■■.■■■■■ = ■)
...
```

構文の詳細は以下の例を見てください。

属性のタイプとオプション

text: シンプルな(1行の)テキスト
- label: 説明となるラベル
- value: デフォルト値
- order: ソート時の並び順 (全てのカスタムフィールドで共通するソートの並び順)
- format:
  - 通常のテキストを指す plain
  - WikiFormatting として整形を行う wiki (0.11.3 以降)
  - 参照可能な値として整形を行う reference (1.0 以降)
  - スペースで区切られた参照可能なリストとして整形を行う list のいずれかを指定する(1.0 以降)

checkbox: ブーリアン値をもつチェックボックス
- label: 説明となるラベル
- value: デフォルト値 (0 または 1)
- order: ソート時の並び順

select: ドロップダウンするリストボックス
- label: 説明となるラベル
- options: リストに表示する値を | (vertical pipe) 区切りで記述
- value: デフォルト値 (options の値から一つを指定)
- order: ソート時の並び順

radio: ラジオボタン。 HTML の select 要素と同じ
- label: 説明となるラベル
- options: リストに表示する値を | (vertical pipe) 区切りで記述
- value: デフォルト値 (options の値から一つを指定)
- order: ソート時の並び順

textarea: 複数行のテキストエリア
- label: 説明となるラベル
- value: デフォルトで設定されるテキスト
- cols: 入力領域のカラム幅
- rows: 入力領域の行数
- order: ソート時の並び順
- format: 通常のテキストを指す plain もしくは WikiFormatting として整形を行う wiki のいずれかを指定する (0.11.3 以降)

サンプル

```
[ticket-custom]
```

```
test_one = text
test_one.label = Just a text box

test_two = text
test_two.label = Another text-box
test_two.value = Default [mailto:joe@nospam.com owner]
test_two.format = wiki

test_three = checkbox
test_three.label = Some checkbox
test_three.value = 1

test_four = select
test_four.label = My selectbox
test_four.options = one|two|third option|four
test_four.value = two

test_five = radio
test_five.label = Radio buttons are fun
test_five.options = uno|dos|tres|cuatro|cinco
test_five.value = dos

test_six = textarea
test_six.label = This is a large textarea
test_six.value = Default text
test_six.cols = 60
test_six.rows = 30
```

Note: *select* タイプのフィールドを非必須 (optional) にしたい場合、 ■■■■■■.*options* オプションの先頭に ■■■■■■■■■■ *(|)* を設定してください。

カスタム属性を含むレポート

カスタムチケット属性は `ticket` テーブルに保存されるのではなく、 `ticket_custom` テーブルに保存されます。したがって、レポートのカスタム属性を表示するためには `ticket` と `ticket_custom` の 2 テーブルを join する必要があります。 `progress` と設定されたカスタムチケット属性の使用例を示します。

```
SELECT p.value AS __color__,
  id AS ticket, summary, owner, c.value AS progress
 FROM ticket t, enum p, ticket_custom c
 WHERE status IN ('assigned') AND t.id = c.ticket AND c.name = 'progress'
AND p.name = t.priority AND p.type = 'priority'
 ORDER BY p.value
```

Note この例は progress が設定されたチケットだけを表示します。すべてのチケットを表示するのではありません。既にいくつかのチケットを作成した後でカスタムチケット属性を定義した

しかし、すべてのチケットエントリを (progress が定義されていないエントリーも一緒に) 表示したいのであれば、クエリにおいてあらゆるカスタムフィールドに `JOIN` を使用する必要があります。

```
SELECT p.value AS __color__,
  id AS ticket, summary, component, version, milestone, severity,
  (CASE status WHEN 'assigned' THEN owner||' *' ELSE owner END) AS owner,
  time AS created,
  changetime AS _changetime, description AS _description,
  reporter AS _reporter,
 (CASE WHEN c.value = '0' THEN 'None' ELSE c.value END) AS progress
 FROM ticket t
    LEFT OUTER JOIN ticket_custom c ON (t.id = c.ticket AND c.name = 'progress')
    JOIN enum p ON p.name = t.priority AND p.type='priority'
 WHERE status IN ('new', 'assigned', 'reopened')
```

```
 ORDER BY p.value, milestone, severity, time
```

この LEFT OUTER JOIN ステートメントに特に注意してください。

データベースを更新する

上記に記述したとおり、カスタムフィールド設定以前に作成されたチケットには、該当するフィールドの値が定義されていない状態になります。以下のような SQL を Trac のデータベースで直接実行することで、カスタムフィールドの初期値を設定することができます (SQLite 向けの SQL ですので、 DBMS に応じて調整してください)。カスタムフィールド 'request_source' が存在しない、全てのチケットにデフォルト値 'None' が挿入されます:

```
INSERT INTO ticket_custom
  (ticket, name, value)
  SELECT
     id AS ticket,
     'request_source' AS name,
     'None' AS value
  FROM ticket
  WHERE id NOT IN (
     SELECT ticket FROM ticket_custom
  );
```

複数のカスタムフィールドを追加している場合、 ticket 表への副問合わせで対象となるカスタムフィールドの名前を指定しなければなりません (訳注: 通常は上記の例ではなく、こちらを使うといいでしょう):

```
INSERT INTO ticket_custom
  (ticket, name, value)
  SELECT
     id AS ticket,
     'request_source' AS name,
     'None' AS value
  FROM ticket
  WHERE id NOT IN (
     SELECT ticket FROM ticket_custom WHERE name = 'request_source'
  );
```

See also: TracTickets, TracIni