Wikiprint Book

Title: Trac のチケットワークフローシステム

Subject: SilverFrost - TracWorkflow

Version: 3

Date: 06/04/26 06:15:31

SilverFrost 目次

Trac のチケットワークフローシステム

Trac のチケットデータベースはコンフィグ可能なワークフローを提供します。

デフォルトのワークフロー

0.10 からアップグレードした Environment

`trac-admin <env> upgrade` を実行したとき、`trac.ini` に `[ticket-workflow]` セクションが追加され、 0.10 でのワークフロー (original ワークフロー) と同様のアクションをするようにデフォルトの設定値が設定されます。

original ワークフローは下図を参照してください:

ワークフローグラフを表示するには JavaScript を有効にしてください。

original ワークフローにはいくつかの重要な "欠点" があります。 新しいチケットを承認 (accept) したときにステータスは 'assigned' に設定されますが、 'assigned' のチケットを再割り当て (reassign) するとステータスは 'new' に設定され、直観的ではありません。 これは original ワークフローから "basic" ワークフローに移行することで解決します。 original ワークフローから basic ワークフローへの移行には contrib/workflow/migrate_original_to_basic.py が役に立つかもしれません。

0.11 で新規作成した Environment

0.11 の環境が新規に作成されるとき、デフォルトのワークフローが trac.ini に構成されます。このワークフローは basic ワークフローです (basic ワークフローは `basic-workflow.ini` 内に記述されています)。 basic ワークフローは 0.10 でのワークフローとは少し違います。

basic ワークフローは下図を参照してください:

ワークフローグラフを表示するには JavaScript を有効にしてください。

そのほかのワークフロー

Trac のソースツリーの中でいくつかのワークフローのサンプルを提供しています。 contrib/workflow の `.ini` コンフィグセクションを探してみてください。その中のひとつにあなたが探しているものがあるでしょう。それらをあなたの `trac.ini` ファイルの `[ticket-workflow]` セクションに貼り付けてください。しかし、もしあなたがすでに起票済みのチケットをもっていて、それらのチケットのステータスが新しいワークフローに含まれて

これらの例の ダイヤグラム を見ることができます。

基本的なワークフローのカスタマイズ

Note: チケットの "ステータス群 (Statuses or states)" は独立した状態で定義することはできません。チケットがとりうるステータスはワークフローで定義された状態遷移から自動生成されます。つまり、チケットを新規

`trac.ini` に `[ticket-workflow]` セクションを作成します。 `[ticket-workflow]` セクション内の各エントリはチケットが取り得るアクションです。 `simple-workflow.ini` の `accept` を例に説明します:

```
accept = new,accepted -> accepted
accept.permissions = TICKET_MODIFY
accept.operations = set_owner_to_self
```

1 行目は `accept` の動作についての定義です。 `accept` は `new` と `accepted` のステータスで有効であり、ステータスが `new` か `accepted` の場合に `accept` が実行されるとステータスが `accepted` になることを表しています。 2 行目は、ユーザが `accept` を行うために必要な権限についての定義です。 3 行目は `accept` を行ったときに、同時にチケットに対して行う操作についての定義です。 `set_owner_to_self` は、チケットの所有者をログイン中のユーザに更新することを表します。同一エントリーに対して複数の定義を行う場合は、カンマ区切りのリストとして設定するこ

actionname`.operations` で使用できる値は以下の通りです:

- del_owner -- チケットの所有者を削除します
- set_owner -- チケットの所有者を選択された所有者か入力された所有者に設定します
  - actionname`.set_owner` カンマ区切りのリストか1つの値を設定することができます

- set_owner_to_self -- チケットの所有者をログインユーザに設定します
- del_resolution -- チケットの解決方法を削除します
- set_resolution -- チケットの解決方法を選択された解決方法か入力された解決方法に設定します
  actionname.set_resolution カンマ区切りのリストか1つの値を設定することができます。 例:

```
resolve_new = new -> closed
resolve_new.name = resolve
resolve_new.operations = set_resolution
resolve_new.permissions = TICKET_MODIFY
resolve_new.set_resolution = invalid,wontfix
```

- leave_status -- "変更しない 現在のステータス: <現在のステータス>" (英語版では "leave as <current status>")
  を表示してチケットへの変更を行いません

Note: set_owner と del_owner などのように相反する操作を同時に指定した場合の動作は不定です。

```
resolve_accepted = accepted -> closed
resolve_accepted.name = resolve
resolve_accepted.permissions = TICKET_MODIFY
resolve_accepted.operations = set_resolution
```

.name 属性を使用した場合の例です。この例のアクションは resolve_accepted ですが、 .name で別名を付けることによって、ユーザからは resolve として見えます。

すべてのステータスで利用可能なアクションであることを表す値として、 * を使用することができます。分かりやすい例は leave です:

```
leave = * -> *
leave.operations = leave_status
leave.default = 1
```

これは '.default' 属性の使用例でもあります。 .default 属性の値は整数であることを期待します。そして、アクションが表示される順番は .default 属性の値で決まります。 .default の値が最も大きいアクションが最初に表示され、デフォルトで選択されます。残りのアクションは .default の値に従い、降順で表示されます。 .default の値を指定しない場合のデフォルト値は0になります。 .default の値には負の値を指定することもできます。

ワークフローにはハードコードされた 2, 3 の制限があります。新しく作成されたチケットのステータスは new になり、チケットには closed のステータスが存在する必要があります。さらにデフォルトのレポート/カスタムクエリでは closed 以外のすべてのステータスをアクティブなチケットとして扱います。

ワークフローを作成・編集するのに contrib/workflow/workflow_parser.py が役に立つかもしれません。
contrib/workflow/workflow_parser.py は GraphViz が理解でき、ワークフローを視覚化するための .dot ファイルを作ることができます。

以下に例を示します (インストールパスは環境により異なる場合があります) 。

```
cd /var/local/trac_devel/contrib/workflow/
sudo ./showworkflow /srv/trac/PlannerSuite/conf/trac.ini
```

実行結果は trac.pdf として出力されます。 (trac.ini 同じディレクトリに出力されます。)

http://foss.wush.net/cgi-bin/visual-workflow.pl でワークフローのパーサのオンラインでのコピーができます。

ワークフローを変更したあと、変更を適用するために Apache
を再起動する必要があります。これは大切なことです。なぜならあなたがスクリプトを起動したとき、それでも変更箇所は現れますが、すべての古いワークフローが

例: ワークフローにテストを追加する

trac.ini の [ticket-workflow] セクションに以下の記述を追加することで optional testing を実現できます。チケットのステータス (status) が new, accepted, needs_work の場合にチケットを testing 状態に遷移させることができます。 testing ステータスでは reject して needs_work 状態に戻すか、 pass して closed 状態に進めることができます。 pass させた場合、 closed での解決方法 (resolution) は自動的に fixed に設定されます。以前のワークフローはそのまま残っているので、このセクションで設定した内容をスキップすることもできます。 (訳注:
通常、チケットのクローズを行うためには TICKET_MODIFY 権限が必要です。このワークフローでは testing

状態からのクローズには権限が不要なので、報告者（reporter）に修正結果をテストしてもらう場合などに有効です）

```
testing = new,accepted,needs_work,assigned,reopened -> testing
testing.name = Submit to reporter for testing
testing.permissions = TICKET_MODIFY

reject = testing -> needs_work
reject.name = Failed testing, return to developer

pass = testing -> closed
pass.name = Passes Testing
pass.operations = set_resolution
pass.set_resolution = fixed
```

tracopt.ticket.commit_updater と testing ワークフローの組み合わせる方法

tracopt.ticket.commit_updater は Trac 0.12 で 古い trac-post-commit-hook を置き換える オプションのコンポーネントです。

デフォルトで、このコンポーネントはチェンジセットのログメッセージの中の close や fix
などのキーワードに反応し、対応するワークフローのアクションを実行します。

もし、上記で述べたような testing ステージがあるような複雑なワークフローを使用していて、キーワードに closes があった場合にステータスを closed にする代わりに、 testing に移したいならば、かなりコードを改変させる必要があるでしょう。

trac-post-commit-hook については、 Trac 0.11 レシピ を参照して下さい。このコンポーネントの修正方法がいくらかわかるでしょう。

例: レビュー状態を追加する

"testing" ステータスが利用者によっては、異なる状況を指すような Trac の使い方をしている場合、実装固有の詳細な箇所は "testing"
に分類せず、デフォルトのワークフローの assigned と closed
ステータスの間に、必要に応じて分岐できるステータスを追加したいと考えるはずです。新しいステータスは reviewing とすべきでしょう。
"submitted for review" されたチケットは、どのようなステータスからでも reassigned になります。レビューが通過した場合、 resolve
アクションを再利用して、チケットを close します。通過しない場合は reassign アクションを再利用して通常のワークフローに戻します。

新しい reviewing ステータスは review アクションに関連付けます。以下のように記述してください:

```
review = new,assigned,reopened -> reviewing
review.operations = set_owner
review.permissions = TICKET_MODIFY
```

デフォルトの Trac 0.11 ワークフローに統合するために、 reviewing ステータスを accept と resolve
アクションに追加します。以下のようになります:

```
accept = new,reviewing -> assigned
[…]
resolve = new,assigned,reopened,reviewing -> closed
```

必要に応じて reviewing からステータスを変更せずに、チケットの担当者（owner）
を変更するための新しいアクションを追加します。この設定を行うと、 new
ステータスに遷移させることなくレビューの担当者を変更することができるようになります。

```
reassign_reviewing = reviewing -> *
reassign_reviewing.name = reassign review
reassign_reviewing.operations = set_owner
reassign_reviewing.permissions = TICKET_MODIFY
```

完全な [ticket-workflow] への設定は以下のようになります:

```
[ticket-workflow]
accept = new,reviewing -> assigned
```

```
accept.operations = set_owner_to_self
accept.permissions = TICKET_MODIFY
leave = * -> *
leave.default = 1
leave.operations = leave_status
reassign = new,assigned,accepted,reopened -> assigned
reassign.operations = set_owner
reassign.permissions = TICKET_MODIFY
reopen = closed -> reopened
reopen.operations = del_resolution
reopen.permissions = TICKET_CREATE
resolve = new,assigned,reopened,reviewing -> closed
resolve.operations = set_resolution
resolve.permissions = TICKET_MODIFY
review = new,assigned,reopened -> reviewing
review.operations = set_owner
review.permissions = TICKET_MODIFY
reassign_reviewing = reviewing -> *
reassign_reviewing.operations = set_owner
reassign_reviewing.name = reassign review
reassign_reviewing.permissions = TICKET_MODIFY
```

例: new チケットでの解決方法 (resolution) を制限する

resolve_new という操作では、 new 状態のチケットで使用可能な、解決方法 (resolution) を設定しています。既に存在する resolve アクションを変更し、 -> の前から new のステータスを削除することで、2種類の resolve アクションが使用できるようになっています。 new のチケットでは制限された解決方法 (resolution) となり、それ以外の一旦 accept されたチケットでは通常通りとなります。

```
resolve_new = new -> closed
resolve_new.name = resolve
resolve_new.operations = set_resolution
resolve_new.permissions = TICKET_MODIFY
resolve_new.set_resolution = invalid,wontfix,duplicate

resolve = assigned,accepted,reopened -> closed
resolve.operations = set_resolution
resolve.permissions = TICKET_MODIFY
```

高度なワークフローのカスタマイズ

ここまでのカスタマイズで十分でないならば、プラグインを使用することでワークフローのさらなる拡張が可能です。プラグインを使用すると、ワークフローに (code_review のような) 操作を追加できます。また、単純なステータスの変更だけでない (トリガを構築するなどの) 2 次的な操作を実行することができます。いくつかの簡単な例は sample-plugins/workflow を参照してください。

プラグインを使用した拡張でさえも十分でないならば ConfigurableTicketWorkflow のコンポーネントを無効にし、ConfigurableTicketWorkflow を完全に置き換える十分な機能を持ったプラグインを作成することも可能です。

ワークフローのステータスをマイルストーンのプログレスバーに追加する

新しいステータスをワークフローに追加した場合、マイルストーンのプログレスバーへの表示もカスタマイズできます。 TracIni を参照してください。

次のステップに向けたアイデア集

(訳注:
この項はワークフローシステムの実装に関するアイデア集です。現在実装されているものではないので、プラグインを作成するときなどに参考にしてください)

New enhancement ideas for the workflow system should be filed as enhancement tickets against the `ticket system` component. If desired, add a single-line link to that ticket here. Also look at the AdvancedTicketWorkflowPlugin as it provides experimental operations.

If you have a response to the comments below, create an enhancement ticket, and replace the description below with a link to the ticket.

- the "operation" could be on the nodes, possible operations are:
  - preops: automatic, before entering the state/activity
  - postops: automatic, when leaving the state/activity
  - actions: can be chosen by the owner in the list at the bottom, and/or drop-down/pop-up together with the default actions of leaving the node on one of the arrows.

This appears to add complexity without adding functionality; please provide a detailed example where these additions allow something currently impossible to implement.

- operations could be anything: sum up the time used for the activity, or just write some statistical fields like

A workflow plugin can add an arbitrary workflow operation, so this is already possible.

- set_actor should be an operation allowing to set the owner, e.g. as a "preop":
  - either to a role, a person
  - entered fix at define time, or at run time, e.g. out of a field, or select.

This is either duplicating the existing *set_owner* operation, or needs to be clarified.

- Actions should be selectable based on the ticket type (different Workflows for different tickets)

Look into the AdvancedTicketWorkflowPlugin's *triage* operation.

- I'd wish to have an option to perform automatic status changes. In my case, I do not want to start with "new", but with "assigned". So tickets in state "new" should automatically go into state "assigned". Or is there already a way to do this and I just missed it?

Have a look at TicketCreationStatusPlugin and TicketConditionalCreationStatusPlugin

- I added a 'testing' state. A tester can close the ticket or reject it. I'd like the transition from testing to rejected to set the owner to the person that put the ticket in 'testing'. The AdvancedTicketWorkflowPlugin is close with set_owner_to_field, but we need something like set_field_to_owner.
- I'd like to track the time a ticket is in each state, adding up 'disjoints' intervals in the same state.