w.c.s. - Development #6844

Gestion des expressions

27 mars 2015 14:37 - Benjamin Dauvergne

Statut:	Fermé	**Début:**	27 mars 2015
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Planning:**	Non

Description

C'est une suite au ticket #1630 il me semble.

Je pense qu'il faudrait envisager un widget "Expression" qui centraliserait les aspects sécuritaires et ergonomique autour de la gestion des expressions Python dans nos interfaces.

Pour l'instant je crois qu'on a simplement un widget 'Préremplissage' mais c'est un cas particulier du cas général "expression".

Il y a deux aspects à spécifier:

- l'aide à la saisie (voir le #6843 pour une première idée par exemple), en proposant une liste de variables disponibles (on le fait déjà pour les templates il me semble), en signalant (par forcément via une erreur sachant qu'on ne sait pas toujours ce qui est disponible) les variables qui ne nous semblent pas disponibles, en repérant les trucs qui puent en faisant un peut de typage, ex.: 3 < form_xxx avec form_xxx qui est forcément une chaîne,
- une validation statique de la sécurité d'une expression: en limitant les expressions autorisées via une analyse de l'arbre syntaxique et une liste blanche, au préalable il faudrait accumuler toutes les expressions sur toutes nos plateformes pour voir ce qui est vraiment utilisé, la première version de cette whitelist devrait forcément valider ces expressions pour être acceptable,
- la gestion de l'exécution des expressions: intercepter les tracebacks et les logger par comme un traceback w.c.s. mais comme un traceback "userland" (il me semble que c'est déjà fait mais je le remet on sait jamais), limiter le temps d'exécution ? faire tourner l'expression dans une sandbox ? mettre en cache le fait qu'une expression est foireuse "consomme trop de ressources", "traceback à chaque fois" pour éviter les DOS.

L'autre intérêt de cette whitelist c'est que ça permettrait de donner une documentation exhaustive des expressions, plutôt que de dire aux gens d'apprendre Python.

C'est un ticket "longue réflexion", d'autres tickets spécifiques pourront s'y relier je pense.

Demandes liées:

Lié à w.c.s. - Development #1630: Vérifier la syntaxe des expressions de pré-...	**Fermé**	**07 septembre 2012**
Lié à w.c.s. - Development #6843: le signe == dans les expressions	**Fermé**	**27 mars 2015**
Lié à w.c.s. - Development #11042: Avoir un widget validation les expressions...	**Fermé**	**25 mai 2016**

Historique

#1 - 27 mars 2015 14:37 - Benjamin Dauvergne

- Lié à Development #1630: Vérifier la syntaxe des expressions de pré-remplissage ajouté

#2 - 27 mars 2015 14:38 - Benjamin Dauvergne

- Lié à Development #6843: le signe == dans les expressions ajouté

#3 - 27 mars 2015 15:45 - Benjamin Dauvergne

Références pour l'utilisation sûre de eval():

- http://tav.espians.com/a-challenge-to-break-python-security.html
- http://nedbatchelder.com/blog/201206/eval_really_is_dangerous.html
- un projet perso d'évaluation "safe" https://github.com/bdauvergne/python-safe-expression/blob/master/safe_expression/compiler.py
- un article LWN sur le sujet https://lwn.net/Articles/574215/
- un mail de l'auteur de pysandbox sur Python-Dev https://lwn.net/Articles/574323/
- http://stackoverflow.com/questions/3513292/python-make-eval-safe
- http://stackoverflow.com/questions/661084/security-of-pythons-eval-on-untrusted-strings

Pour pysandbox mon avis est que c'est une approche par blackliste qui essaie de laisser utilisable un maximum du langage (ça permet de faire un eval() dans le contexte d'un module pas un eval d'expression). Le fait de ne viser que les expressions me semble plus atteignable, déjà avec *builtin* vide et l'interdiction de toute variable commençant par '_' on devrait aller assez loin.

#4 - 27 mars 2015 15:53 - Benjamin Dauvergne

http://thepaulprog.blogspot.fr/2009/02/safelite-exploit.html

#5 - 27 mars 2015 17:04 - Benjamin Dauvergne

https://pypi.python.org/pypi/simpleeval

#6 - 27 mars 2015 17:09 - Benjamin Dauvergne

https://github.com/newville/asteval

#7 - 27 mars 2015 17:10 - Benjamin Dauvergne

https://pypi.python.org/pypi/evalidate/0.6

#8 - 29 mars 2015 10:57 - Frédéric Péters

- Lié à Development #6102: Notification des exceptions liées à un formulaire/workflow ajouté

#9 - 10 juin 2016 14:05 - Benjamin Dauvergne

- Lié à Development #11042: Avoir un widget validation les expressions calculées dans les workflows ajouté

#10 - 26 janvier 2017 15:19 - Thomas Noël

- Lié à Development #6102: Notification des exceptions liées à un formulaire/workflow supprimé

#11 - 02 mai 2017 10:25 - Benjamin Dauvergne

Idée évoquée durant l'eocamp bruxelles 2016:

- avoir une méthode sur la classe Workflow renvoyant toutes variables de substitutions visibles (ou des préfixes dans certains cas comme les appels de ws).
- même chose sur les formdef
- passer le formdef (ou un nouvel objet contexte qui prendrait lui le formdef en paramètre) aux champs d'expression et de template
- ne valider une expression si elle ne contient que des références à des variables issues du contexte (ou des références qui de par leur préfixe sont susceptibles d'exister), idem pour les templates

Contrainte nouvelles que ça impose:

- tant qu'on a pas branché un formulaire sur le bon workflow ni défini l'ensemble des options de workflows ou des appels de ws nécessaires, on ne peut pas utiliser les variables correspondantes dans des expressions, i.e. on ne peut plus faire les choses dans le désordre
- on ne peut pas écrire d'expression d'exemple qui ne marcherait pas en vue de finir plus tard

Bonus:

- faire cette analyse statiquement pour avoir un tableau de bord des erreurs actuellement présentes avec des liens vers les pages pour corriger, ça permettre de repartir de l'état actuel proprement ne corrigeant tout ce qui est bancale

#12 - 16 juillet 2023 09:25 - Frédéric Péters

- Statut changé de Information nécessaire à Fermé

- Planning mis à Non

> gestion des expressions Python

On a plutôt finalement décidé de quitter ça.