ANNEXE 3 : GERER LES CONDITIONS DANS UN FORMULAIRE

Lorsque le formulaire / la démarche comprend des informations à n'afficher que sous certaines conditions, deux méthodes complémentaires sont disponibles dans le générateur :

Conditions entre étapes : Afficher / masquer une étape en fonction de la valeur des champs d'une autre étape Cas d'utilisation : toutes les démarches qui doivent adapter le nombre de questions à afficher en fonction des informations données par l'usager
Conditions entre champs au sein d'une étape : Afficher / masquer des champs en fonction de la valeur des autres champs au sein d'une même étape Cas d'utilisation : Au-delà d'un certain nombre de conditions, le nombre d'étapes peut considérablement augmenter et rendre la démarche complexe. Il est alors possible de gérer les conditions entre champs à l'intérieur d'une étape.

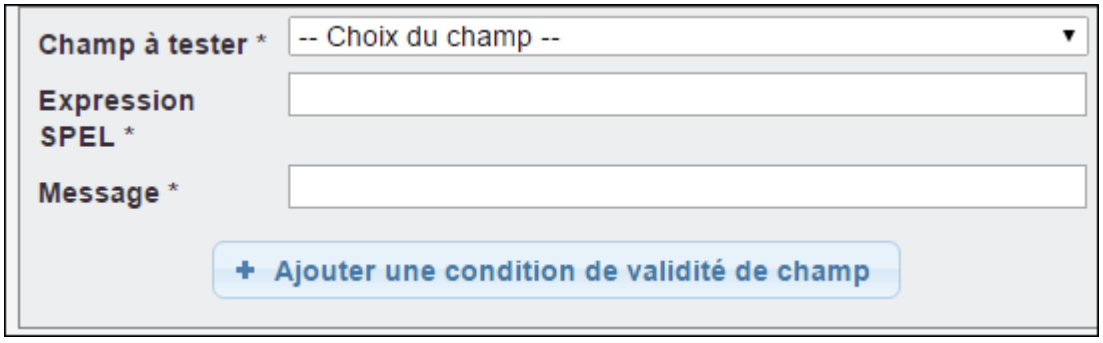
1ère méthode : Contrôles de champs avec des expressions SPEL

Principe des expressions SPEL

Les expressions SPEL servent à contrôler la valeur d'un champ ou plusieurs champs de l'étape en cours en fonction de valeurs d'autres champs.
Le contrôle est réalisé à la validation de l'étape et donne lieu à l'affichage d'un message d'erreur le cas échéant. Le message d'erreur est positionné sur un champ qui doit être obligatoire et valorisé pour que le contrôle s'applique. Il n'est pas possible de positionner le contrôle sur un champ tableau, ressource, séparateur, fichier.

Intégration des expressions SPEL

Ces expressions doivent être intégrées dans les champs avancés des étapes.



Champ à tester : champ de l'étape sur lequel apparaîtra le message en cas de non-respect de l'expression SPEL à tester. Le test ne sera effectué que si le champ en question est obligatoire et valorisé dans le formulaire.
Message : message d'erreur à afficher au niveau du champ précité en cas de

non-respect de l'expression SPEL à tester.
Expression SPEL : expression exploitant le langage SPEL et les valeurs traduisibles (voir §6) devant renvoyer la valeur true ou false, selon que l'expression est vérifiée ou pas (voir éléments de syntaxe ci-dessous).

Syntaxe utilisable dans les expressions SPEL

Opérateurs arithmétiques :

Signe	Signification
+	Addition
–	Soustraction
*	Multiplication
/	Division
%	Modulo

Dans le cas de champs de type Date, les opérations d'addition et de soustraction renvoient des nombres de jours.
Opérateurs logiques : and (et), or (ou)
Il est possible d'utiliser OR ou ||, qui sont remplacés automatiquement par « or », de même que AND ou &&, qui sont remplacés automatiquement par « and ».
L'utilisation de parenthèses () permet de définir le regroupement en cas de tests multiples.

Comparateurs :

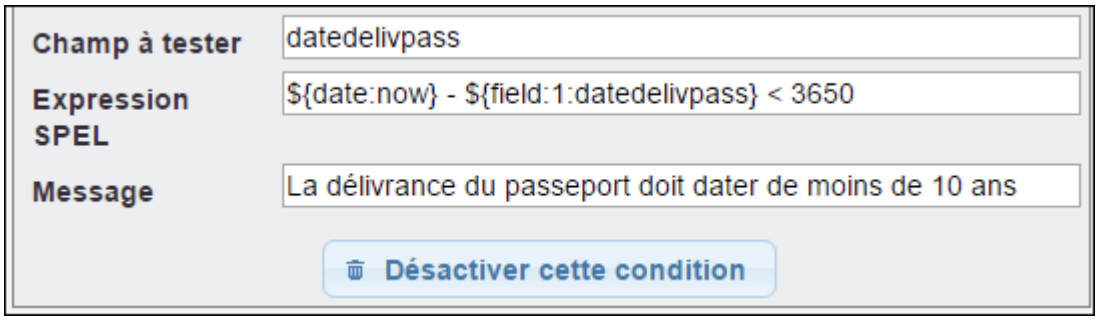
Signe	Signification	Remarque
<	Strictement inférieur	
>	Strictement supérieur	
<=	Inférieur ou égal	
>=	Supérieur ou égal	
==	Egal	Ex : ${field:S1:F2}=='1′ pour vérifier que la clé 1 est choisie.
!	Différent de	
contains.	Contient, pour le cas de champs contenant plusieurs valeurs.	Ex : ${field:S1:F2}.contains('1') pour vérifier que la clé 1 fait partie des choix.

Règles de gestion à connaître :

- Au-delà des valeurs de champs du formulaire, il est possible de faire référence à la date du jour avec ${date:now}, ou à une date spécifique avec ${date:DD/MM/YYYY}.
- Il est possible de définir des listes de valeurs en les encadrant par des accolades { }
- Les chaînes de caractères doivent être encadrés par des apostrophes ' ', l'apostrophe devant être doublée dans la chaîne de caractères pour être prise en compte.
- Pour écrire des chiffres décimaux, le séparateur de décimales doit être le point « . » et non la virgule « , ».

Exemple 1 :
Dans une étape S1, il faut faire en sorte que si l'usager choisit « Autres » à la question F1 « de quel mode de garde bénéficie votre enfant ? » « 1. Crèche ; 2. Nourrice ; 3. Autre », le champ F2 suivant « Précisions » soit obligatoire pour passer à l'étape suivante. Mettre la condition de validité sur le champ F1 de l'étape S1 : ${field:S1:F1} != '3' or ${field:S1:F2} !=

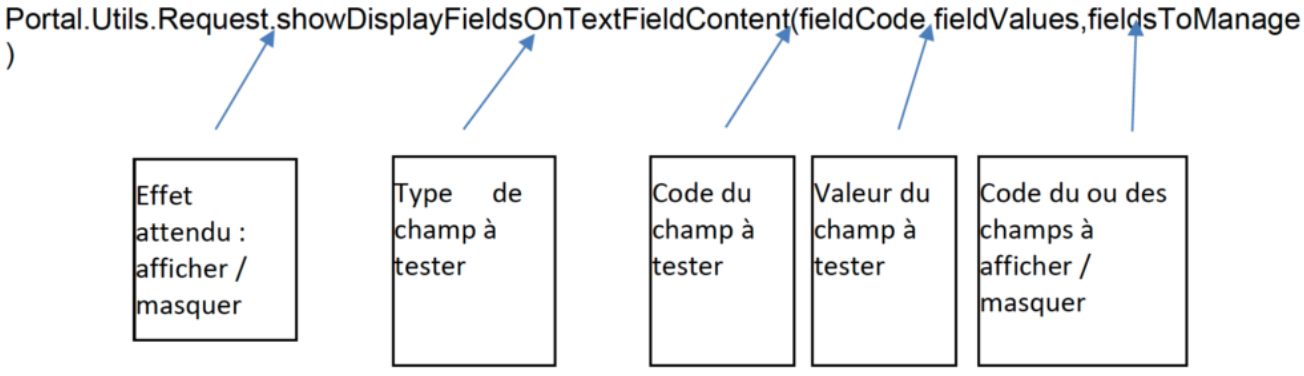
Exemple 2 :
Dans une étape S2, il faut faire en sorte que l'usager choisisse dans le champ F3 « Date de délivrance du passeport » une date de moins de 10 ans. Mettre la condition de validité sur le champ F3 de l'étape S2 : ${date:now} – ${field:S2:F3} < 3650



2ème méthode : Conditions entre champs au sein d'une même étape avec un code JavaScript

Principes du code JavaScript

L'intégration de code JavaScript permet de contrôler l'affichage de champs de la démarche (masquer/afficher des champs ou rendre des valeurs de champs inaccessibles).
Le champ à contrôler doit obligatoirement être facultatif.
La fonction JavaScript qui se compose ainsi :



Intégration du code JavaScript

Le code doit être intégré dans les champs avancés de l'étape.

En fonction du type de champ à tester pour valider la condition, plusieurs cas sont disponibles. A l'intégration les expressions doivent être modifiées / adaptées à la démarche en cours.
Les éléments à modifier sont indiqués en gris dans chaque code. Eviter de « copier coller » cette partie de code, mieux vaut saisir directement ces éléments dans le générateur de démarche.
Si plusieurs valeurs doivent déclencher la condition dans le code à tester, elles doivent être indiquées les unes à la suite des autres séparées par une virgule. Idem si plusieurs étapes doivent être affichées ou masquées.
Il est possible d'intégrer plusieurs codes dans la même étape : il suffit de les séparer par un point-virgule.
Exemple :
Portal.Utils.Request.hideDisplayFieldsOnCheckboxValue('F10',['V4','V6','V7','V8'],['F11','F12'])
Il est possible de rendre une cellule d'un tableau non modifiable (exemple : dans le cas d'un tableau où une cellule affiche la somme d'une colonne, celle-ci doit être de préférence non modifiable par l'utilisateur). L'ajout de l'élément suivant true dans le code Javascript indique que la cellule de destination sera non modifiable.
Exemple : Portal.Utils.Request.tableSum('F1','C1','F1_L26_C1',true).

Exemples

Action	Champ testé	Code
Montrer un champ	Champ radio	Portal.Utils.Request.showDisplayFieldsOnRadioChecked('fieldCode',['fieldValues'],['fieldsToManage'])
Cacher un champ (affiché	Champ radio	Portal.Utils.Request.showDisplayFieldsOnRadioChecked('fieldCode',['fieldValues'],['fieldsToManage'])
Montrer un champ	Menu	Portal.Utils.Request.showDisplayFieldsOnSelectValue('fieldCode',['fieldValues'],['fieldsToManage'])
Cacher un champ	Menu	Portal.Utils.Request.hideDisplayFieldsOnSelectValue('fieldCode',['fieldValues'],['fieldsToManage'])
Montrer un champ	Case à cocher	Portal.Utils.Request.showDisplayFieldsOnCheckboxValue('fieldCode',['fieldValues'],['fieldsToManage'])
Cacher un champ	Case à cocher	Portal.Utils.Request.hideDisplayFieldsOnCheckboxValue('fieldCode',['fieldValues'],['fieldsToManage ''])
Montrer un champ	Texte libre	Portal.Utils.Request.showDisplayFieldsOnEmptyTextField('fieldCode',['fieldsToManage']) Montre le champ si le champ testé est vide.
Cacher un champ	Texte libre	Portal.Utils.Request.hideDisplayFieldsOnEmptyTextField('fieldCode',['fieldsToManage'] Montre le champ si le champ testé est vide.
Masquer un choix	Texte libre	Portal.Utils.Request.hideDisplayFieldOptionsOnTextFieldContent('fieldCode',['fieldValue'], 'fieldToManage,['fieldValue'])
Afficher la plus petite valeur d'une colonne	Tableau	Portal.Utils.Request.tableMin('fieldCode', ['fieldColumn'],['outputCode']) La valeur sera affichée dans le champ outputCode.
Afficher la plus grande valeur d'une colonne	Tableau	Portal.Utils.Request.tableMax('fieldCode', ['fieldColumn'],['outputCode']) La valeur sera affichée dans le champ outputCode.

Afficher la somme d’une colonne	Tableau	Portal.Utils.Request.tableSum(‘fieldCode’, [‘fieldColumn’],[‘outputCode’]) La valeur sera affichée dans le champ outputCode.
Afficher le résultat d’un calcul		Portal.Utils.Request.calcul(”, ”, true); Dans l’élément , indiquer le code du champ dans lequel le résultat doit s’afficher et dans l’élément l’opération qui doit être réalisée. Dans la formule les champs à évaluer sont à citer entre accolades : ex Portal.Utils.Request.calcul(‘F4′,'({F1}+{F2}+{F3})*(19.6/100)’,true);