

Les contrats : la surface de communication

Référence d'article

Article complet

<https://b-ziq.com/fr/blog/filemaker-contrats-surface-communication/>

Doctrine fondamentale

- Une méthode liée par contrat exige une structure de communication explicite et indépendante du contexte dans les deux directions.
- L'enveloppe est l'unité de communication et la forme concrète du contrat.
- Toute méthode reçoit une enveloppe de requête et retourne une enveloppe de réponse.
- L'échec est une donnée contractuelle et doit être retourné à travers l'enveloppe de réponse.
- Toute invocation de méthode produit exactement l'un de trois résultats : succès, sortie anticipée ou échec.
- Un contrat valide déclare explicitement son entrée, sa sortie et ses modes d'échec.

Définitions

Enveloppe

Unité de communication. Forme concrète par laquelle un contrat de méthode est honoré à l'exécution.

Enveloppe de requête

Structure transportant l'identité de l'appel, l'entrée déclarée, les métadonnées d'appel, l'intention de rappel, le contexte de session, le contexte de fenêtre et l'historique d'exécution.

Enveloppe de réponse

Structure invariante retournée par toute méthode contenant l'état de succès, le résultat et les informations d'échec.

Sortie anticipée

Terminaison délibérée avant la complétion du chemin d'exécution complet qui n'est pas un échec. Représentée par `success: false` et `errorCode: 0`.

Erreur comme donnée

Principe selon lequel l'échec est retourné comme donnée structurée à travers l'enveloppe de réponse plutôt que produit sous forme d'effet de bord.

Propriété

- `app`, `module`, `method`, `obj`, `meta` et `callbackData` proviennent de l'appelant.
- `session` et `windowData` sont ajoutés par la frontière qui valide la requête.
- `callstack` est enrichi à chaque saut d'exécution par les appelants et les méthodes.
- L'appelant possède l'enveloppe de requête.
- La méthode possède l'enveloppe de réponse.

Responsabilités

Enveloppe de requête

- `app` porte l'identité applicative.
- `module` porte l'identité de module.
- `method` porte l'identité de méthode.
- `obj` porte l'entrée déclarée.
- `meta` porte les métadonnées d'appel.
- `callbackData` porte l'intention de rappel.
- `session` porte le contexte de session.
- `windowData` porte le contexte de fenêtre.
- `callstack` porte l'historique d'exécution.

Enveloppe de réponse

- `success` indique si la méthode s'est complétée comme prévu.
- `result` porte la sortie déclarée.
- `errorCode` identifie les conditions d'échec.
- `errorData` porte les informations d'échec structurées.

Contraintes

- Une méthode doit recevoir explicitement tout ce dont elle a besoin avant le début de l'exécution.
- Une méthode ne doit pas recevoir d'entrée à travers un état ambiant.
- Une méthode ne doit pas dépendre d'un contexte non déclaré.
- Une méthode doit retourner l'échec programmatiquement par le même canal que le succès.
- Rien ne peut être transmis en dehors de l'enveloppe.
- Rien ne peut être supposé arriver par un autre canal que l'enveloppe.
- Toute requête vers une méthode doit arriver sous forme d'enveloppe.
- Toute réponse d'une méthode doit repartir sous forme d'enveloppe.
- L'enveloppe de requête doit rendre explicites toutes les responsabilités de communication.

- Toute méthode doit retourner la même forme d'enveloppe de réponse.
- Les méthodes ne doivent pas afficher de boîtes de dialogue.
- Les méthodes ne doivent pas définir de variables globales contenant des chaînes d'erreur.
- Les méthodes ne doivent pas signaler un échec par un mécanisme extérieur à l'enveloppe de réponse.

Invariants

- L'enveloppe est l'unité de communication.
- L'enveloppe est la forme concrète du contrat.
- Un obj vide est une déclaration qu'aucune entrée n'est requise.
- La forme de l'enveloppe de réponse est invariante.
- Un succès et un échec retournent les mêmes clés.
- Toute invocation de méthode produit exactement l'un de trois résultats : succès, sortie anticipée ou échec.
- La distinction entre sortie anticipée et échec est lisible depuis l'enveloppe de réponse.
- L'échec retourné à travers l'enveloppe demeure une donnée.
- Rien ne franchit la frontière contractuelle sans être déclaré.

Validation

Un contrat est valide lorsque les trois conditions suivantes sont vraies :

- L'entrée est déclarée par la forme de obj dans l'enveloppe de requête.
- La sortie est déclarée par la forme de result dans l'enveloppe de réponse.
- L'échec est déclaré par les codes d'erreur que la méthode peut retourner.

La conformité contractuelle est évaluée en vérifiant :

- Que toute entrée requise existe dans la structure de requête déclarée.
- Que la sortie retournée correspond à la structure de réponse déclarée.
- Que les conditions d'échec retournées sont identifiées par des codes d'erreur déclarés.

Conséquences des violations

- Une méthode qui exige des données absentes de obj possède une dépendance non déclarée et n'est pas conforme au contrat.
- Une méthode qui dépend d'un état ambiant dépend du contexte.
- Une méthode qui modifie la structure de result modifie son contrat.
- Une méthode qui retourne une sortie non déclarée introduit un état ambiant à travers un canal structuré.

- Une méthode qui échoue avec un code d'erreur non déclaré produit une condition d'échec que le contrat ne décrit pas.
- Une méthode qui signale un échec à travers des boîtes de dialogue ou d'autres effets de bord viole la pureté de méthode.
- Une méthode qui produit des effets de bord ne peut plus être raisonnée comme une unité.
- Un changement de contrat non suivi est indiscernable d'une rupture silencieuse.
- Un contrat qui ne déclare pas l'entrée, la sortie et l'échec ne peut pas être audité complètement.