# Mise en place de la sérialisation

## Présentation

### Document concerné

Ce document traite de la sérialisation/dé-sérialisation d’un objet persistant.

Je rédige ce court document pour préciser le travail de programmation nécessaire.

### Versions

06/12/2016 : version initiale

### Auteur

Invisible Media

## Projet traité

### Cadre du projet

Le présent projet est un logiciel de calcul numérique et algébrique.  
Ce projet est un moyen de former des équations très longues et fastidieuses et, de les factoriser selon une approche commune.

But : résoudre des équations, trouver l’ensemble des solutions algébriquement et obtenir une équation en fonction de paramètres et de données numériques fixées.

### Ensemble des fonctionnalités

L’ensemble des fonctionnalités est détaillé dans ce document :

[Toutes les fonctionnalités (dégroupé).xlsx](Toutes%20les%20fonctionnalités%20(dégroupé).xlsx)

### Nom de la fonctionnalité traitée ici

Lecture/sauvegarde par sérialisation

## Besoin relatif

La sérialisation binaire est la meilleure des solutions. Contrairement à la sérialisation par XML, le seul attribut nécessaire à une sérialisation binaire est d’ajouter l’attribut **[Serializable]** devant la déclaration de la classe.

Les listes à plusieurs éléments sont entièrement gérées sans aucune adaptation supplémentaire ; ce qui n’est pas le cas dans les listes avec une sérialisation XML.

Les données peuvent être dynamiques ; aucun typage fort n’est nécessaire ; la sauvegarde est réalisée par rapport au contenu interne de l’objet et la sérialisation s’opère récursivement sur chaque champ.

## Relation avec les autres fonctionnalités

### Utilisation

La donnée est consignée, lue et modifiée au cours du temps via le modèle de données

### Explications

Il s’agit de savoir traiter toutes les versions du modèle de données.

Or, une sérialisation binaire implique que toutes les données sont inscrites dans le fichier persistant.

Une fois ce fichier sauvegardé sur disque, pour le reprendre, il suffit de dé-sérialiser.

Excepté que si le logiciel est à une version ultérieure, toutes les données inexistantes dans le fichier déclenche une erreur de donnée **null** qu’il est difficile de tenir compte. Dans ce cas, le logiciel devrait revenir à une version précédente.

Lorsque le système utilise la version antérieure, il encadre également l’ensemble des versions ultérieures.

## Programmation

La fonction de sérialisation est réalisée avec la classe .NET *BinaryFormatter* ; elle instancie les fonctions de sérialisation et de dé-sérialisation.

La sérialisation est formée par la lecture de l’ensemble des objets contenus dans l’objet à persister. Cet objet persistant est simplement une liste de couples nom/valeur dont la valeur doit être du type associé à la définition du nom indiqué dans la classe statique de définition des couples nom/type. Cette vérification est effectuée à chaque changement de la valeur de l’élément associé au nom.

La dé-sérialisation nécessite le type de donnée à priori contenu dans le fichier. La classe statique des couples nom/type n’est pas présente dans le fichier ; elle est juste construite dès le lancement de l’application.

Une fois la dé-sérialisation effectuée, les données ont été chargées. Si une erreur survient (ce qui ne doit jamais arriver) alors le chargement aura échoué.

A noter que chaque fichier comporte un numéro de version du logiciel ; si un fichier n’est pas de la bonne version, ce problème est corrigé une fois le fichier écrasé et sauvegardé avec la nouvelle version.