

PRID - Projet "Stuck Overflow"

Table des matières

PRID - Projet "Stuck Overflow"

- Table des matières

- Préalable

- Analyse

 - Acteurs

 - Diagramme de classes

 - Validations métier

 - Interface utilisateur

 - Diagramme de cas d'utilisation

 - Vue globale*

 - Vue détaillée*

 - Gestion de la réputation

 - Implémentation des règles métier

- Organisation du travail

- FAQ

- Modalités pratiques et consignes

- Planning

- Copies d'écrans de www.stackoverflow.com

 - Affichage d'une question

 - Création d'une question

 - Parcourir les questions

 - Liste des tags

 - Liste des utilisateurs

Préalable

[6/12] Le présent document présente le contexte fonctionnel du projet à réaliser pour le cours de PRID. Il sera complété ultérieurement par un autre document reprenant les consignes que vous devrez suivre pour réaliser le projet. Sur base du présent document, vous pouvez déjà commencer à développer votre application en commençant par la partie back-end.

[6/12] Le but de ce projet est de réaliser un **site web** de gestion de **questions et réponses** similaire à www.stackoverflow.com.

Avant de prendre connaissance de cet énoncé, nous vous recommandons de créer un compte sur www.stackoverflow.com et de vous familiariser avec le fonctionnement de ce

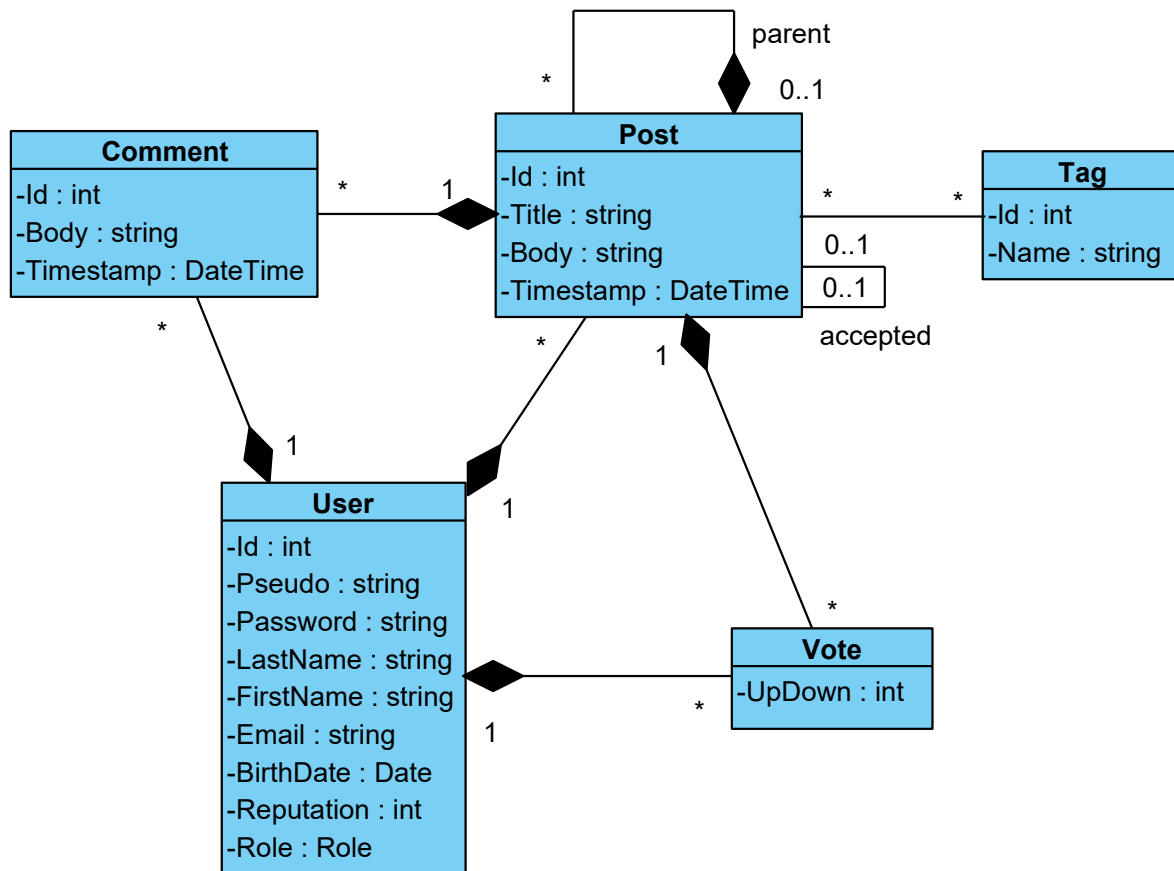
site. Les fonctionnalités du projet en sont très largement inspirées.

Analyse

Acteurs

- **Visitor** : représente un utilisateur anonyme qui ne s'est pas connecté.
- **Member** : représente un utilisateur connecté dont le rôle est `Member`.
- **Admin** : représente un utilisateur connecté dont le rôle est `Admin`.

Diagramme de classes



- Un `Post` peut représenter aussi bien une question qu'une réponse. S'il s'agit d'une question, le post pourra être le parent de zéro ou plusieurs réponses. Les réponses n'ont pas de `Title`. Le `Body` contient le texte de la question / réponse au format *markdown*. Parmi toutes les réponses à une question, une seule peut être acceptée par le membre qui a posé la question.
- Un `Tag` est un mot-clef qui peut être associé uniquement aux posts de type question (exemple : `JavaScript`, `Base de données`, `MySQL`, ...).
- Un `Comment` représente un commentaire fait par un utilisateur sur un `Post`. Le `Body`

contient le texte du commentaire au format *markdown*. N'importe quel membre peut faire un ou plusieurs commentaires sur un post.

- Un `vote` représente un vote d'un utilisateur vis-à-vis d'un post. Ce vote peut être positif (`up` ou `+1`) ou négatif (`down` ou `-1`). N'importe quel membre peut voter pour ou contre un post (sauf les siens et pour autant qu'il ait suffisamment de points de réputation - voir plus loin). Lorsqu'on affiche un post, on affiche son `score` qu'on calcule en faisant la somme des valeurs de votes associés à ce post. Un membre peut changer son vote, soit en le supprimant, soit en passant de `up` à `down` ou inversement.
- La classe `user` représente les utilisateurs enregistrés dans le système. Le champ `role` permet de distinguer les simples `Members` et les `Admins`.
- Pour les posts et les commentaires, le champ `timestamp` sert à stocker la date et l'heure de création de l'entité (par exemple, la date et l'heure d'un [17/12] `vote` commentaire).
- Lorsqu'on supprime une entité, il faut supprimer en cascade les entités qui en dépendent à travers une composition.

Validations métier

- `User` : voir exercice.
- `Post` :
 - `id` est requis et est la clé primaire auto-incrémentée pour la table.
 - `title` est obligatoire pour les questions et doit être nul pour les réponses.
 - `body` est requis.
 - `timestamp` est requis et est initialisé par défaut avec la date système au moment de la création ou de la mise à jour du post.
- `Comment` :
 - `id` est requis et est la clé primaire auto-incrémentée pour la table.
 - `body` est requis.
 - `timestamp` est requis et est initialisé par défaut avec la date système au moment de la création ou de la mise à jour du post.
- `Vote` :
 - La clé primaire pour cette table est la combinaison de `Post.id` et `User.id`.
 - `upDown` est requis et ne peut valoir que `-1` ou `1`. Quand un vote est annulé, on supprime le record correspondant en BD.
- `Tag` :
 - `id` est requis et est la clé primaire auto-incrémentée pour la table.
 - `name` est requis et doit être unique.

Interface utilisateur

Pour l'interface utilisateur, vous partirez des exercices réalisés pendant le cours et vous vous baserez sur l'exemple `prid1920-tuto` pour le canevas général. Par contre, vous êtes libres d'organiser vos composants comme vous le désirez en ayant les objectifs suivants en tête :

- Votre application doit être conviviale et ergonomique.
- Elle doit permettre de réaliser toutes les fonctionnalités décrites dans ce document.

Vous pouvez vous inspirer du *look and feel* de l'interface utilisateur de

`www.stackoverflow.com` (voir copies d'écrans jointes).

Pour la gestion de l'affichage ou de l'édition des champs formatés en *markdown*, nous vous conseillons les librairies suivantes :

- <https://www.npmjs.com/package/ngx-markdown> : propose un composant angular qui permet de convertir du code markdown en HTML et d'afficher celui-ci.
- <https://www.npmjs.com/package/ngx-simplemde> : propose un composant angular qui est un éditeur de code markdown avec une fonctionnalité de prévisualisation du résultat.

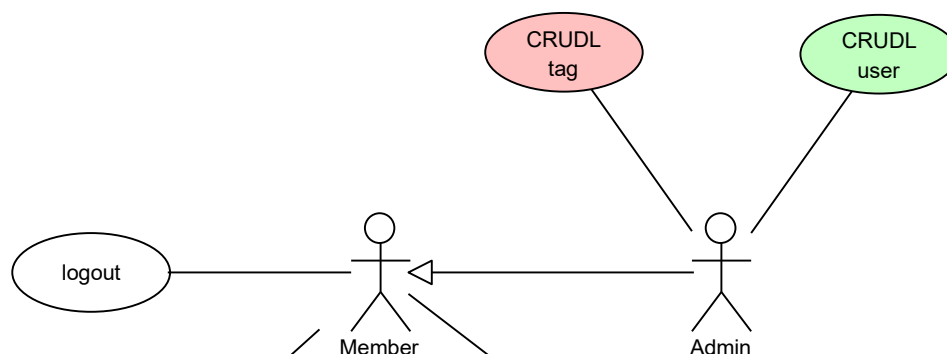
Diagramme de cas d'utilisation

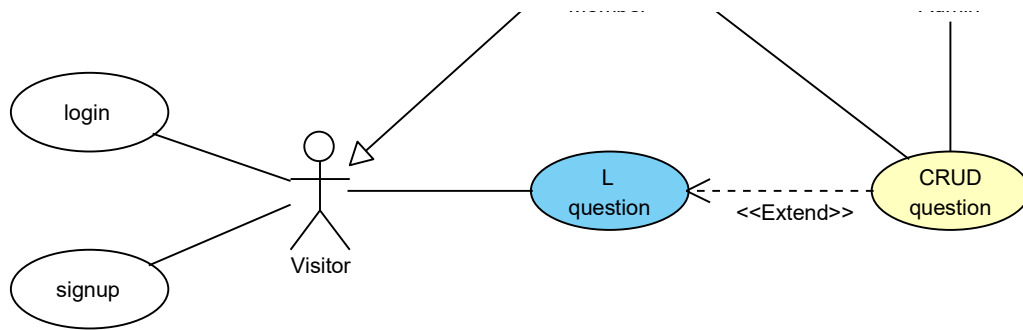
Vue globale

Dans le diagramme ci-dessus, les couleurs indiquent les 4 grosses fonctionnalités de l'application :

- **L question** : parcourir la liste des questions.
- **CRUD question** : création, affichage, mise à jour et suppression des questions et des données liées (réponses, commentaires, votes, ...).
- **CRUDL user** : gestion complète des utilisateurs.
- **CRUDL tag** : gestion complète des tags.

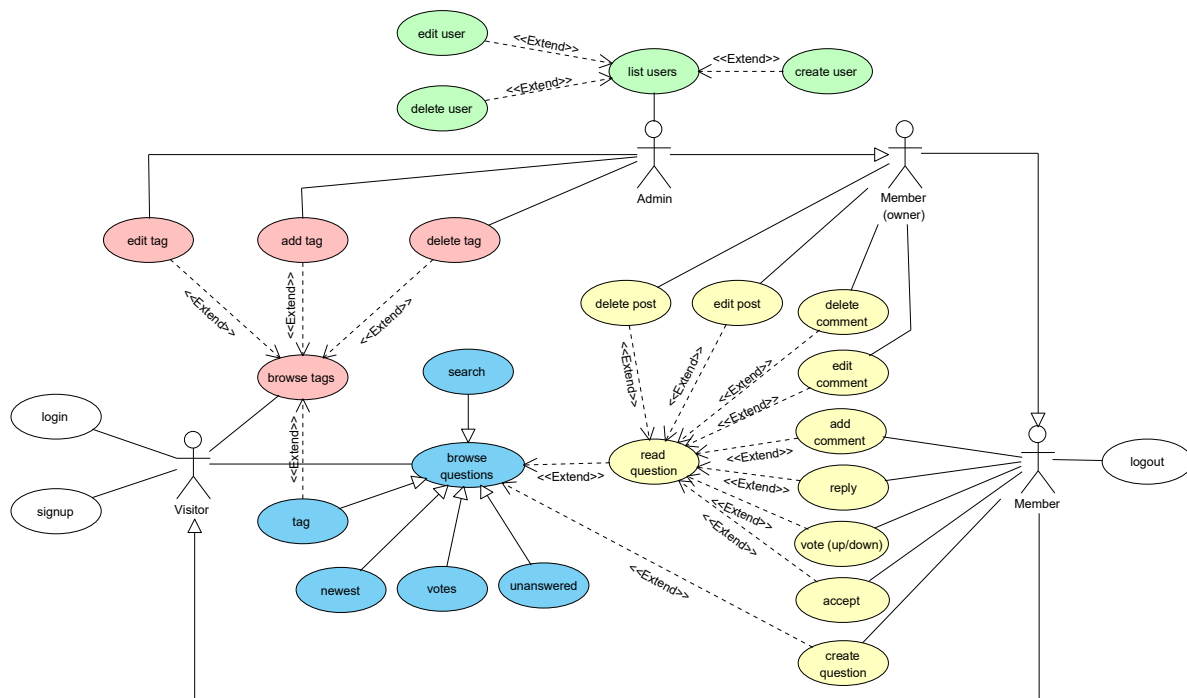
Les fonctionnalités en blanc existent normalement déjà dans votre application suite aux exercices faits durant le cours.





Vue détaillée

Dans ce diagramme, chacune des grosses fonctionnalités est détaillée de manière plus fine.



Descriptions :

- **login** : permet de se connecter.
- **signup** : permet à un utilisateur de s'enregistrer dans le système en tant que membre. Les administrateurs sont prédéfinis dans la base de données (au minimum le pseudo `admin` avec le mot de passe `admin`).
- **logout** : permet de se déconnecter de l'application.
- **list users** : permet d'afficher la liste des utilisateurs.
- **create user** : permet de créer un nouvel utilisateur.
- **edit user** : permet de modifier les données d'un utilisateur.
- **delete user** : permet de supprimer un utilisateur.
- **browse questions** : cas d'utilisation abstrait représentant le fait qu'il faut pouvoir

parcourir les différentes questions pour identifier celles qui nous intéressent. Ce cas d'utilisation se décline en différents cas d'utilisation concrets correspondant à différentes manières de parcourir les questions :

- **newest** : affichées dans l'ordre chronologique inverse (plus récente au début) de création de la question.
- **votes** : affichées dans l'ordre décroissant de score (le score pris en compte est le meilleur score de la question et de ses réponses).
- **unanswered** : n'affiche que les questions pour lesquelles aucune réponse n'a été acceptée ; elles sont triées comme dans **newest**.
- **tag** : n'affiche que les questions associées à un tag ; elles sont triées comme dans **newest**.
- **search** : permet de faire une recherche *full text* sur les données des posts (auteur, commentaires et tags compris), afin de retrouver une question ou une réponse ; ce qui est affiché, c'est une liste de questions qui satisfont au critère(s) de recherche.
- **read question** : permet d'afficher une question ainsi que toutes les réponses correspondantes. Dans cet affichage, on doit voir également les *timestamps*, les commentaires, les scores, les *tags* et les membres correspondants (voir copies d'écrans). Les posts sont affichés dans l'ordre suivant :
 - en premier lieu on montre la question ;
 - ensuite, si elle existe, on montre la réponse acceptée ;
 - finalement, on montre le reste des réponses dans l'ordre décroissant de leur score et, à score identique, dans l'ordre chronologique inverse (le plus récent en premier).
- **reply** : tout en bas de la page présentée dans **read question**, on trouve un formulaire de réponse (voir copies d'écrans).
- **add comment** : permet d'ajouter un commentaire sur un post (questions ou réponses). Les commentaires sont affichés en dessous du post auquel ils se rattachent, dans l'ordre chronologique de leur `Timestamp` (voir copies d'écrans).
- **edit comment** : permet de modifier un commentaire sur un post. Seul un utilisateur ayant un rôle `Admin` ou l'auteur du commentaire peuvent l'éditer.
- **delete comment** : permet de supprimer un commentaire sur un post. Seul un utilisateur ayant un rôle `Admin` ou l'auteur du commentaire peuvent le supprimer.
- **vote (up/down)** : chaque post (questions et réponses) doit pouvoir faire l'objet d'un vote *Up* ou *Down* par un simple click. Un vote *Up* augmente le score du post de `1`. Un vote *Down* réduit le score du post de `1`. Il doit pouvoir être possible d'annuler ou de changer son vote.
- **accept** : permet à l'auteur d'une question de marquer une des réponses comme étant la réponse acceptée. [6/12] L'auteur de la question doit pouvoir annuler ou changer l'acceptation d'une réponse.
- **edit post** : permet de modifier un post (question ou réponse). Seul un utilisateur ayant

un rôle `Admin` ou l'auteur du `Post` peuvent l'éditer.

- **delete post** : permet de supprimer un post (question ou réponse). Seul un utilisateur ayant un rôle `Admin` ou l'auteur du post peuvent le supprimer, moyennant le respect des règles suivantes :
 - Un `Admin` peut supprimer n'importe quel post. S'il s'agit d'une question, les réponses associées, ainsi que tous les commentaires liés à la question et aux réponses, seront supprimés. S'il s'agit d'une réponse, tous les commentaires associés seront également supprimés.
 - L'auteur d'un post peut le supprimer s'il n'y a pas de commentaires associés et s'il s'agit d'une réponse. S'il s'agit d'une question, il peut la supprimer à condition qu'elle n'ait fait l'objet d'aucun commentaire ni d'aucune réponse.
- **create question** : un membre peut créer une nouvelle question (voir copies d'écrans).
- **browse tags** : permet de parcourir les différents tags existants et de voir pour chacun d'eux combien de questions lui sont associées. En cliquant sur le nombre de questions on accède directement à **tag**.
- **add tag** : permet d'ajouter un nouveau tag (rôle `Admin` uniquement).
- **delete tag** : permet de supprimer un tag (rôle `Admin` uniquement).
- **edit tag** : permet d'éditer le nom d'un tag (rôle `Admin` uniquement).

Gestion de la réputation

La réputation permet de faire certaines actions sur les *posts*. Voici un tableau qui indique, pour chaque action, la réputation nécessaire pour l'effectuer ainsi que l'augmentation du score de réputation qu'elle procure :

action	réputation nécessaire pour entreprendre l'action	gain/perte de réputation
ajouter une question	0	0
ajouter une réponse	0	0
ajouter un commentaire	0	0
accepter une réponse	0	+15 pour le répondant et +2 pour l'auteur de la question
voter positivement pour une question ou une réponse	15	+10 pour l'auteur de la question ou de la réponse
voter négativement contre une question ou une réponse	30	-2 pour l'auteur de la question ou de la réponse et -1 pour le votant

Remarque : les gains et les pertes de réputation sont acquis définitivement au moment de l'action. Si l'objet de l'action est supprimé ou modifié par après, le gain ou la perte reste acquis.

Implémentation des règles métier

Important : Gardez à l'esprit que les règles métier doivent être vérifiées à la fois du côté front-end et du côté back-end.

Organisation du travail

Nous vous conseillons d'organiser votre travail dans l'ordre suivant :

1. Mise en place du modèle dans EF.
2. Création de vos DTO.
3. Création d'une API de base permettant de récupérer une liste de posts ou un post spécifique.
4. Tests de votre API avec Postman.
5. Use cases **browse questions** (en version basique) et surtout **read question**.
6. Ajout progressif des UC qui étendent **read question** en ajoutant à chaque fois la partie back-end et la partie front-end nécessaire.

7. UC **create question**.
8. Variantes du UC **browse questions**.
9. UC's de gestion des tags.
10. UC's de gestion des utilisateurs.

FAQ

Tout au long du projet, nous publierons des réponses à des questions ou des précisions pour le projet. N'oubliez pas de vérifier régulièrement son contenu (voir moodle).

Modalités pratiques et consignes

Le projet sera réalisé par groupe de deux étudiants (un seul groupe de trois sera autorisé si le nombre d'étudiants est impair) conformément à l'**architecture et aux technologies vues au cours** (Angular / ASP.NET Core / EF Core / MySQL).

Le développement sera réalisé de manière **collaborative** par **tous les étudiants** du groupe. Outre l'évaluation du projet chaque membre du groupe sera **interrogé individuellement** pour valider sa maîtrise des technologies et architecture utilisées dans le cadre du projet.

L'utilisation d'un **repository GIT BitBucket** est obligatoire (il sera créé et rendu accessible par le professeur). Son utilisation **correcte** et **régulière** (au minimum un commit par session de travail, libellés clairs pour les commits permettant de suivre l'historique du projet, ...) par tous les membres de l'équipe fera partie des critères d'évaluation. L'utilisation de branches est laissée libre et à l'appréciation de chaque étudiant (sauf la branche `master` qui est obligatoire).

La version finale de votre projet fera l'objet d'un commit dans la branche `master` dans BitBucket et sera marquée par un Tag. Elle sera constituée des éléments obligatoires suivants :

- le **code source** de l'application, sous forme d'un projet respectant la structure de fichiers vue au cours et compatible avec l'environnement de développement Visual Studio Code ;
- tous les fichiers externes utilisés par votre application ;
- la base de données doit pouvoir être créée avec la commande `dotnet ef database update` ;
- **l'application sera responsable d'insérer les données de tests fournies dans la base de données** lors de son premier lancement ;
- un fichier de **notes de livraison** destiné à fournir au professeur des renseignements que vous jugerez utile de porter à son attention : bugs connus, fonctionnalités

manquantes, fonctionnalités supplémentaires, ...

Afin de rendre l'installation de votre application sur une autre machine la plus simple possible, vous respecterez les règles suivantes :

- Le nom de votre projet sera `prid1920_gxx`, où `xx` représente le numéro de votre groupe exprimé sur deux chiffres (ex: groupe 2 => `prid1920_g02`).
- La base de données MySQL doit avoir le même nom que votre projet, soit `prid1920_gxx`.
- Vous respecterez le canevas de projet vu au cours.
- Le projet doit pouvoir être exécuté sur une machine de l'école simplement en lançant les commandes `npm start` pour le front-end et `dotnet run` pour le back-end.

Planning

La version finale de votre projet doit être livrée pour le mercredi 8 janvier 2020 avant 23h59.

Copies d'écrans de `www.stackoverflow.com`

Pour rappel, les copies d'écrans qui suivent sont reprises à titre d'information pour illustrer l'ergonomie mise en place par StackOverflow. Seules les fonctionnalités et les règles métier décrites plus haut dans le document doivent être implémentées.

Affichage d'une question

What does 'initialization' exactly mean?

Asked yesterday Active yesterday Viewed 126 times



My csapp book says that if global and static variables are initialized, than they are contained in .data section in ELF relocatable object file.

3

So my question is that if some `foo.c` code contains

▼

★

```
int a;
int main()
{
    a = 3;
}
```

and `example.c` contains,

```
int b = 3;
int main()
{
...
}
```

is it only `b` that considered to be initialized? In other words, does initialization mean declaration and definition in same line?

c++ c linker

share edit flag

asked yesterday

 Jinwoo Park
42 ● 5

▲ Global "uninitialized" variables typically end up in a "bss" segment, which will be initialized to zero. – Some programmer dude yesterday ✎

▲ [stackoverflow.com/questions/1169858/...](https://stackoverflow.com/questions/1169858/) This might help. – Amal John yesterday ✎

▲ All definitions are declarations in C and C++, and a definition of a variable may optionally include an initialiser. If no initialiser is provided for a global/static, then the default is zero-initialisation (or, in C++, for a struct/class type, calling an appropriate default constructor if one exists). What that translates to in terms of where variables are located in memory, or represented in a segment of an object or executable file, depends on the implementation. – Peter yesterday ✎

▲ Related, if not a dup: stackoverflow.com/q/58791133/6699433 – klutt yesterday

▲ Here's a summary with examples: electronics.stackexchange.com/a/237759/6102 – Lundin yesterday

show 1 more comment

5 Answers

active

oldest

votes

3

It means exactly what it says. Initialized static storage duration objects will have their init values set before the main function is called. Not initialized will be zeroed. The second part of the statement is actually implementation dependant, and implementation has the full freedom of the way it will be archived.

✓

When you declare the variable without the keyword `extern` you always define it as well

share edit flag

answered yesterday

 P_J
14.9k ● 2 ● 8 ● 28

1 ▲ To complicate matters: C has the messy concept of 'tentative definition'. – pmg yesterday

add a comment



More ways to Q&A: Try Stack Overflow for Business with secure single sign-on for your entire team



▲

Both are considered initialized

3

They get [zero initialized](#) or constant initialized (in short: if the right hand side is a compile time constant expression).



If permitted, Constant initialization takes place first (see Constant initialization for the list of those situations). In practice, constant initialization is usually performed at compile time, and pre-calculated object representations are stored as part of the program image. If the compiler doesn't do that, it still has to guarantee that this initialization happens before any dynamic initialization.

For all other non-local static and thread-local variables, Zero initialization takes place. In practice, variables that are going to be zero-initialized are placed in the .bss segment of the program image, which occupies no space on disk, and is zeroed out by the OS when loading the program.

To sum up, if the implementation cannot constant initialize it, then it must first zero initialize and then initialize it before any dynamic initialization happens.

share edit flag

answered yesterday



darune

5,357 1 10 33

add a comment

Your Answer

B *I*

☐ community wiki

Post Your Answer

Création d'une question

Title

Be specific and imagine you're asking a question to another person

test test test

Body

Include all the information someone would need to answer your question

B *I*

Show formatting tips

bla bla :

- * fkfkkf
- * ffkfkf

Et un peu de code :

```

csharp
public class Dummy {
    public void Foo() {}
}
                    
```


...

draft saved code bold italic quote

bla bla :

- fikfik
- ffikkf

Et un peu de code :

```
public class Dummy {  
    public void Foo() {}  
}
```

Tags
Add up to 5 tags to describe what your question is about

c# class

☐ Answer your own question – [share your knowledge, Q&A-style](#)

Parcourir les questions

All Questions

[Ask Question](#)

18,508,468 questions

Newest

Active

Bountied **321**

Unanswered

More ▾

 Filter

0

votes

0

answers

22 views

BadPaddingException with Cipher class

I'm coding a program that generates a simmetrical AES key, encodes it using an assimetrical public key, writes that encoded AES key in a plain text file, and then it recovers the encoded AES key and ...

javaencryptioncryptography

modified 39 secs ago



CRONOS.LXIII

89 ● 8

0

votes

1

answer

11 views

how to cache images paths in angular 2

I have created an image slider. To fetch next and previous images I have created 2 functions. component.ts: ngOnInit(){ this.length = this.images.length; } getNextImage(){ this.imageUrl = "...

angularcaching

answered 44 secs ago



Ajaz

48 ● 1 ● 1 ● 14

21

votes

4

answers

17k views

How do I update an existing document inside ElasticSearch index using NEST?

I am trying to update an existing indexed document. I have indexed tags, title and owners field. Now when the user changes the title, I need to find and update the document inside the index. Should I ...

c#elasticsearchnest

answered 47 secs ago



Ali Bayat

1,694 ● 2 ● 26 ● 26

1

vote

2

answers

28 views

Angular 8: Run Component Class Methods Synchronously/Sequentially

How do I run methods synchronously in Angular Typescript? These are not functions, but methods. First one calls a service, and then second saves into an array. runTwoMethods() { this....

javascriptangulartypescriptangular8

modified 48 secs ago



SailorLuvBoat

76 ● 9

Top Questions

Ask Question

Interesting

321

Bountied

Hot

Week

Month

59

votes

9

answers

6k

views

Is this a pure function?

javascript

function

functional-programming

modified Nov 9 at 8:32 Shiva 7

39

votes

8

answers

3k

views

Object Oriented Programming - how to avoid duplication in processes that differ slightly depending on a variable

c#

oop

modified 2 days ago Michał Turczyn 23.2k

43

votes

6

answers

4k

views

How to get the address of a C++ lambda function within the lambda itself?

c++

c++11

lambda

c++14

c++17

modified Nov 8 at 8:18 majkrzak 156

30

votes

3

answers

2k

views

Equality operator does not get defined for a custom spaceship operator implementation in C++20

c++

c++20

spaceship-operator

answered 2 days ago Oktalist 11.3k

18

votes

6

answers

2k

views

Why is this claimed dereferencing type-punned pointer warning compiler-specific?

c

pointers

casting

answered 14 hours ago supercat 60.9k

33

votes

3

answers

5k

views

App archived with Xcode 11.2 (11B52) rejected: ITMS-90534: Invalid Toolchain [duplicate]

ios

xcode

app-store

toolchain

macos-catalina

modified Nov 8 at 5:33 Muhamed Bajramović 1

Liste des tags

Tags

A tag is a keyword or label that categorizes your question with other, similar questions. Using the right tags makes it easier for others to find and answer your question.

Q Filter by tag name

Popular

Name

New

javascript

× 1900150

JavaScript (not to be confused with Java) is a high-level, dynamic, multi-paradigm, object-oriented, prototype-based, weakly-typed programming language.

813 asked today, 4914 this week

java

× 1607168

Java (not to be confused with JavaScript, JScript or JS) is a general-purpose, platform-independent, statically typed, object-oriented programming language.

609 asked today, 3410 this week

c#

× 1359025

a high level, statically typed, multi-paradigm programming language developed by Microsoft. C# code usually targets .NET.

458 asked today, 2559 this week

php

× 1316820

a widely used, high-level, dynamic, object-oriented and interpreted scripting language primarily designed for server-side web development.

346 asked today, 2091 this week

python

× 1284571

a multi-paradigm, dynamically typed, multipurpose programming language, designed to be quick (to learn, to use, and to write).

1094 asked today, 6372 this week

android

× 1233311

Google's mobile operating system, used for programming or developing digital devices (Smartphones, Tablets, Automobiles, TVs, Wearables).

398 asked today, 2282 this week

jquery

× 969968

a JavaScript library, consider also adding the jQuery tag. jQuery is a popular cross-browser JavaScript library that facilitates DOM manipulation.

144 asked today, 858 this week

html

× 859691

the standard markup language used for structuring web pages and formatting content. HTML describes the structure of a document.

330 asked today, 2048 this week

c++

× 641615

a general-purpose programming language. It was originally designed as an extension to C, and keeps a similar syntax, but is now a general-purpose programming language.

223 asked today, 1435 this week

ios

× 613618

the mobile operating system running on the Apple iPhone, iPod touch, and iPad. Use this tag [ios] for questions related to iOS.

161 asked today, 921 this week

css

× 609809

a representation style sheet language used for describing the look and formatting of HTML (HyperText Markup Language), XML (eXtensible Markup Language), and SVG (Scalable Vector Graphics).

195 asked today, 1214 this week

mysql

× 577464

a free, open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL). DO NOT use this tag for questions about MySQL.

192 asked today, 1059 this week


Liste des utilisateurs

14 sur 1526-06-20 à 18:53

Users

Reputation	New users	Voters	Editors	Moderators
------------	-----------	--------	---------	------------

week month quarter year all




Gordon Linoff

New York, United States

1,361

sql, mysql, sql-server




jezrael

Bratislava, Slovakia

1,260


pandas, python, dataframe



GMB

1,208

sql, mysql, database




kaya3

United Kingdom

955


python, java, algorithm



akrun

883

r, dplyr, list




Tim Biegeleisen

Singapore

875

regex, python, sql




Ronak Shah

Pune, India.

875

r, dplyr, dataframe




T.J. Crowder

United Kingdom

871

javascript, jquery, html




CertainPerformance

Austin, TX, USA

870

javascript, function, functional-programming




VonC

France

870

git, github, go




Willem Van Onsem

Ypres, Belgium

858

haskell, django, python




Nicolas

Québec, QC, Canada

800

javascript, html, arrays




Martijn Pieters

Cambridge, UK

779

python, python-3.x, string




Jon Skeet

Reading, United Kingdom

765

c#, java, .net




Gerardo Furtado

Melbourne, Australia

752

d3.js, javascript, svg



Frank van Puffelen

San Francisco, CA

746

firebase, javascript, firebase-realtime-database

15 sur 15

26-06-20 à 18:55