How to create a generic method in Dart?

Asked 4 years, 11 months ago Active 1 year, 6 months ago Viewed 19k times



I'm trying to use generic methods in Dart (1.22.0-dev.10.3). Here is a simple example:









```
abstract class VR<T> {
  VR();
 bool foo<T>(T value);
class VRInt extends VR<int> {
 VRInt();
 bool foo<int>(int n) => n > 0; // Thinks n is Object
}
class VRString extends VR<String> {
 VRString();
 bool foo<String>(String s) => s.length > 0; // Thinks s is Object
}
```

Both subclasses generate errors that say the argument to foo is an Object.

I'm sure this is just a syntactic error on my part, but I've searched the documentation and can't find an answer.

```
generics
         dart
              generic-method
```

Share Improve this question Follow

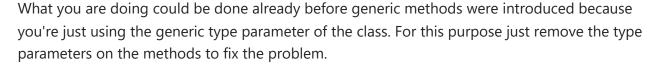
asked Feb 2 '17 at 14:15 **281** 1 2 8

2 Answers





18





Generic methods are to allow to pass a type that specializes a method no the call site

Join Stack Overflow to learn, share knowledge, and build your career.





}

and then use it like

```
new MyClass().myMethod<List<String>>(['a', 'b', 'c']);
```

Share Improve this answer Follow

edited Jul 8 '20 at 2:44

answered Feb 2 '17 at 14:23



- 1 How can we say that T should implement an Interface/Class Siempay Oct 27 '19 at 15:46
- 4 T myMethod<T extends SomeType> . See also <u>stackoverflow.com/questions/18698873/...</u>
 Günter Zöchbauer Oct 27 '19 at 17:19

 ✓
- @GünterZöchbauer May you could correct the output: console.log('javascript') to print('dart')Cassio Seffrin Jul 7 '20 at 20:51
- 1 Thanks @CassioSeffrin! Seems I mixed in some TypeScript: D Günter Zöchbauer Jul 8 '20 at 2:45



Generic methods have type parameters that are not directly related to their class parameter types.

8 You can have a generic method on a class without generic:



```
class MyClass {
  T add<T>(T f()) => f();
}
```

You can also have a generic method on a class with generic:

```
class MyClass<A> {
   A aInstance;
   // here <T> is needed because unrelated to <A>
   T add<T>(T f(A a)) => f(aInstance);
}
```

In your case, you don't need the type parameter on the methods:

```
abstract class VR<T> {
   VR();
   bool foo(T value);
```

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up



```
bool foo(int n) => n > 0; // Thinks n is Object
}

class VRString extends VR<String> {
   VRString();

bool foo(String s) => s.length > 0; // Thinks s is Object
}
```

But there are no concept of generic method needed here.

Share Improve this answer Follow

answered Feb 2 '17 at 14:57



Alexandre Ardhuin

58.7k 12 134 121

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up