Using generics vs returning dynamic in Dart

Asked 2 years ago Active 1 year, 7 months ago Viewed 303 times

3

I have an abstract class that stores different settings for the user so I can reference them from the same location. Then, whenever I need to see what their setting is I call a function that maps the name (string) of the setting to the value of it. For example:

```
abstract class UserPrefs {

  static bool auto_create;
  static String favorite_location;
  // etc

}
```

Then I have a function (also inside `UserPrefs`) to get the associated values like so:

```
static dynamic getSettingFromString(String name) {
  switch (name) {
    case 'auto_create': return auto_create;
    case 'favorite_location': return favorite_location;
    // etc
  }
  throw UserPrefNotFound('User preference "$name" does not exist.');
}
```

This works fine, but so does this:

```
static T getSettingFromString<T>(String name) {
  switch (name) {
    case 'auto_create': return auto_create as T;
    case 'favorite_location': return favorite_location as T;
    // etc
  }
  throw UserPrefNotFound('User preference "$name" does not exist.');
}
```

I'm wondering what the potential advantages of either may be (in speed, efficiency, etc) and ultimately what difference is under the hood. Basically, why is one (or is one) better than the other and are there unforeseen consequences I'm not aware of for using one or the other. Thanks!

generics dart dynamic

Share Improve this question Follow

1 Answer

Active Oldest Votes

0

I think this is a good question, the main difference between generics and `dynamic` to me is:

Generics are restricted to only 1 type, while dynamic is not because it stops the compiler to perform "type checking".

Real example:

```
class Foo<T> {
  var bar = List<T>();
  var foo = List<dynamic>();

  void addBar(T elem) {
    bar.add(elem);
  }

  void addFoo(dynamic elem) {
    foo.add(elem);
  }
}

var bar = Foo<String>()
  ..addBar("hello")
  ..addBar(123); //compile error, you can't add an integer to <string> list

var foo = Foo<String>()
  ..addFoo("hello")
  ..addFoo(123); //OK because dynamic accept any type
```

Dynamics enable you to use mixed maps `Map<String, dynamic>` or "emulate" union types logic like:

```
dynamic foo = //some type;
if (foo is bool) {
  //do somethings
} else if (foo is String) {
  //do something else
}
```

without using interfaces or inheritance for types.

So the next question should be, what is the difference between `Object` and `dynamic`?

You can find the answer here: https://stackoverflow.com/a/31264980/2910520