Frameworks

What are frameworks?

All programs are different just like snowflakes, yet they have similarities in code that we want to draw your attention to, or rather, how these similarities can be used to the developer's advantage.

In programming, it is common practice to reuse code packed into libraries in order to simplify the development and avoid making the same errors over and over. Such libraries exist for most programming languages; they provide good documentation and well-tested code used by many people.

Large applications like Internet stores, online banks or social networks often need the same typical components and functionality such as user authorization, database interaction, sending notifications and so on.

To reuse them, developers created a special kind of software called **frameworks** for all popular programming languages.

A **framework** is a universal, reusable piece of software that facilitates the development of typical applications or their parts.

It consists of structured code templates and provides generic functionality which can be easily extended for the needs of a specific application.

To relinquish control on low-level tasks and focus on the high-level problems, you should use the API provided by the framework. It can significantly reduce total development time.

Some frameworks are so large that they are in fact a union of different frameworks and libraries under a single name.

Frameworks vs libraries

Applications that use frameworks are built on top of them and extend their code to get specific functionality. In a sense, a framework serves as the skeleton of an application or its parts and sets "the rules of the game".

A library, on the other hand, only provides some specific operations without having such a global influence.

This is the key difference between frameworks and libraries. However, libraries can be provided as parts of frameworks.

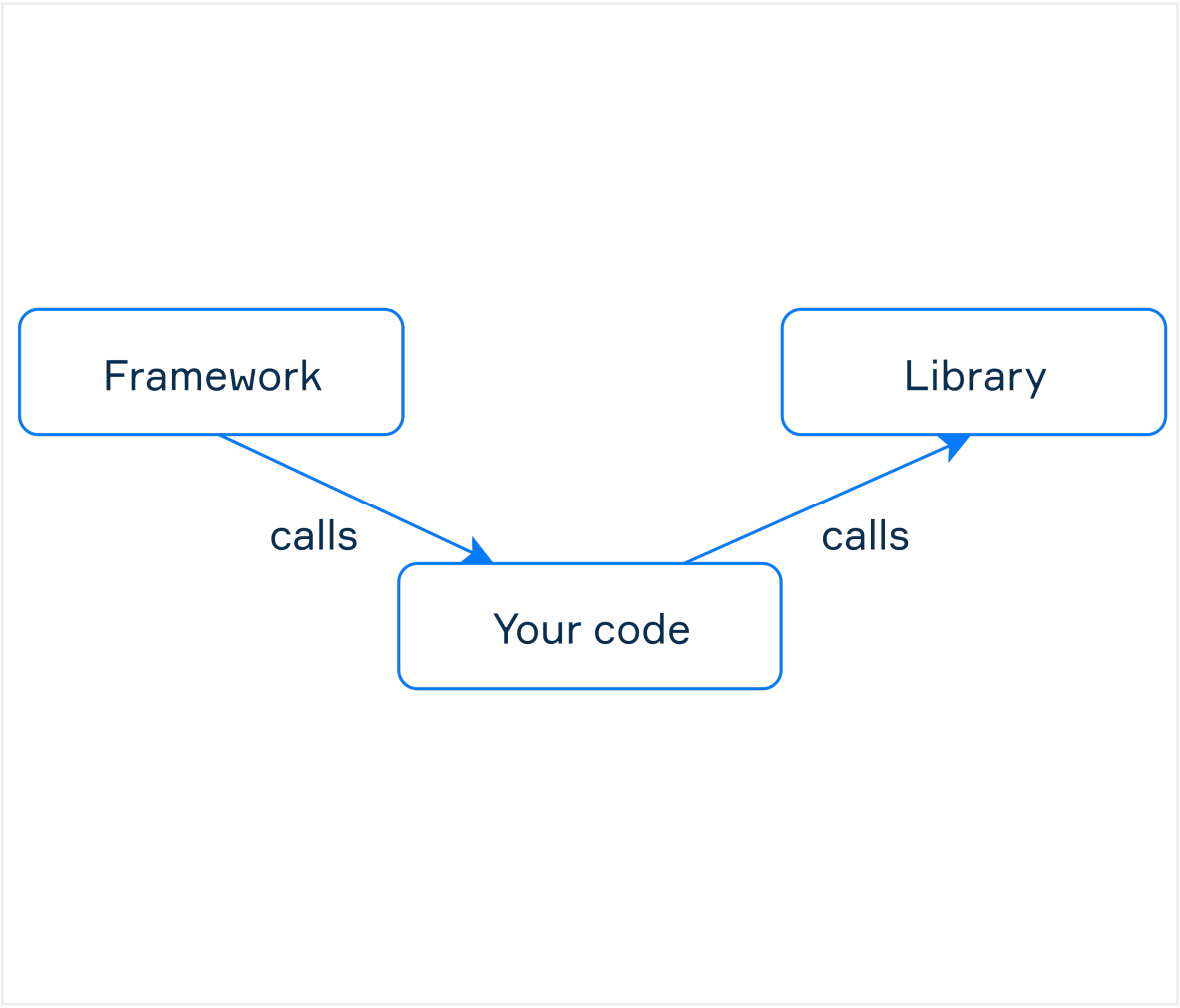
Of course, there's no escape from evident similarities between frameworks and libraries. The programmer who uses a framework does not modify its source

code, acting only as its user.

Inversion of Control

The most common principle that comes with frameworks is **Inversion of Control (IoC)**.

In a framework, unlike in libraries or standard user applications, the overall program's flow of control is dictated not by the caller but by the framework. It means the framework calls your code, and not vice versa:



This happens because a framework provides templates for solving possible tasks and the interaction between the templates has been defined by developers of the framework.

The user of a framework just takes the templates and extends them with application-specific code.

Time to weigh everything. To start on a positive note, the use of frameworks has a number of strong advantages:

- Rapid prototyping and development;

- Standardization of project structures: it is easier to understand similar projects with the same structure;
- Wide use in companies around the world;
- Bug fixes and security updates by the authors;
- A well-designed skeleton: as a rule, frameworks use up-to-date practices and patterns to provide a firm skeleton for applications.

Despite the advantages, there are a number of common drawbacks:

- Selecting an unsuitable framework can make an application harder to implement;
- Application slowdown: frameworks often do a lot of heavyweight things hidden from programmers;
- It is difficult to replace a no longer suitable framework with another one while libraries can be easily replaced;
- You may encounter a bug in the framework which may affect your work.

How to choose frameworks

As a rule, each programming language has several frameworks to choose from.

Of course, if you come to a company where some framework is already being used, there may be no choice for you. But if you do have a choice, try to take into account all possible benefits and problems when making a decision.

Here is some general advice for choosing a suitable framework:

- Pay attention to well-known frameworks with good documentation. This will greatly simplify the use and allow you to easily find developers already familiar with this framework. Some popular frameworks even become a **de facto standard** for developing specific types of applications. Such frameworks should be considered first.
- If you write a small application that will most probably never change, you can develop it without frameworks. Moreover, for such an application they can introduce unnecessary additional complexity. But you may also consider the use of the so-called **lightweight frameworks** or choose a framework only for some part(s) of your application.

This is general information; as we said, programs are different, and so are the possible frameworks out there. The best part is getting to know specific frameworks and working with them closely.