SOFTWARE FRAMEWORKS – AN OVERVIEW

PROGRAMMING APPLICATIONS AND FRAMEWORKS (IT3030)

CONTENTS

- Preliminaries
- Programming vs Software Engineering
- Generic functionalities in Software Development
- Frameworks: what are they?
- How does Frameworks help with the Software Engineering Approach?
- Framework vs. Libraries
- Why use frameworks?
- Advantages of using frameworks
- Limitations of using frameworks

PRELIMINARIES

- What is this course?
  - **PAF (IT3030)** introduces the fundamentals of full stack web development with an emphasis on application of good software engineering practices and Industry standards.
- Learning Outcomes of this course?
  - After completing this module, you will,
    - Know the basics of Software Frameworks
    - Apply Industry standard software development best practices
    - Understand the REST architectural style
    - Develop Java based REST web services (Back-end)
    - Develop JavaScript based Interactive web applications (Front-end)
- Module Outline
- Assessments

PROGRAMMING VS SOFTWARE ENGINEERING

- What is Programming?
  - Programming simply means writing Code to create an application.
- What is Software Engineering?
  - Software engineering is the end-to-end process of applying good engineering principles and software development practices to produce quality software.
- Programming vs Software Engineering
  - Software Engineering is **NOT** just coding. Coding is just one part of a large process.
  - **Q: What other practices come under software engineering?**

PROGRAMMING VS SOFTWARE ENGINEERING

- What other practices come under software engineering?
  - Analyzing user requirements and designing the software to meet those requirements.
  - Communicating with all the stakeholders to resolve issues and update them on the progress.
  - Choosing the software architectures, patterns, algorithms, coding techniques, technologies used to realize the software design based on sound design considerations.
  - Quality Assurance (QA) of the developed software to ensure that the software meets the client requirements and acceptable reliability, performance and security considerations.
  - Deploying/ releasing the software for actual use.
  - Maintaining the developed software as needed after release/ deployment.

GENERIC FUNCTIONALITIES IN SOFTWARE DEVELOPMENT

common features

- Generic functionalities in software development
  - Most (web) applications have a lot of features in common.
    - E.g.: User Authentication, routing, database connectivity, scheduled jobs etc.
  - There is little point in rewriting all the functionalities (including the common ones) from scratch every time new software is developed.
  - What if the generic functionalities can be collected and made reusable?

FRAMEWORKS: WHAT ARE THEY?

- A framework is an integrated set of software artifacts (such as classes, objects, and components) that collaborate to provide a reusable architecture for a family of related applications [1].
- A framework enables a coding environment that contains low-level libraries to address conventional coding issues. The objective of a framework is to deliver faster development of an application. This includes everything we need to build large-scale applications, such as templates based on best practices [4].
- A framework is a foundation for developing software applications. Software engineers and developers use a framework as a template to create websites and applications. Developers do this by adding code to a framework, then personalizing it for their specific purpose [5].
- A software framework is an abstraction in which software, providing generic functionality, can be selectively changed (**extended**) by additional user-written code, thus providing application-specific software [6].

HOW DO FRAMEWORKS HELP WITH THE SOFTWARE ENGINEERING APPROACH?

- How do Frameworks help with the Software Engineering Approach?
  - Architectural Patterns (e.g.: MVC, MVVM)
  - Design Patterns
    - Inversion of Control
  - Standards/ Best Practices (Naming conventions, Coding standards, Directory structures etc.)
  - Tooling Support (Package managers, Dependency managers, Boilerplate code generation)
  - Testing Support

FRAMEWORK VS. LIBRARIES

- Certain features make a framework different from other library forms, including the following [7]:
  - **Default Behavior**: Before customization, a framework behaves in a manner specific to the user's action.
  - **Inversion of Control**: Unlike other libraries, the global flow of control within a framework is employed by the framework rather than the caller.
  - **Extensibility**: A user can extend the framework by selectively replacing default code with user code.
  - **Non-modifiable Framework Code**: A user can extend the framework, but not modify the code.

WHY USE FRAMEWORKS?

- The designers of software frameworks aim to facilitate software developments by allowing designers and programmers to **devote their time to meeting software requirements rather than dealing with the more standard low-level details of providing a working system, thereby reducing overall development time** [6].
- The aim of frameworks is to provide a common structure so that developers don't have to redo it from scratch and can reuse the code provided. In this way, frameworks allow us to cut out much of the work and save a lot of time.
- To summarize: there's **no need to reinvent the wheel**.

ADVANTAGES OF USING FRAMEWORKS

- Improved coding, easy code reusability and ease of debugging code compared with from-scratch developments.
- Community support.
- Accelerated development (if the programmer is familiar with the framework).
- Often provides caching and optimized processes for resource intensive tasks.
- Enables faster methods for development with less code (boilerplate code etc.).
- Better compliance to accepted security standards.

LIMITATIONS OF USING FRAMEWORKS

- Learning the Framework and not the programming language prevent programmers from gaining an in-depth understanding of the programming language.
- Users of a framework are forced to respect the limitations and conventions imposed by its design.
- Options to tweak functionalities are limited.
- A framework will come with everything required to satisfy a wide range of use cases, not all these features will be used for a given project.
- The right framework for the application should be chosen, or else performance and user experience may be impacted negatively.
- We must be up-to-date with new/deprecated features in every version.

SOME EXAMPLES

- Spring/ Spring Boot (Java)
- Express.js (with Node.js)
- Laravel (PHP)
- .NET/ .NET Core (C# etc.)
- Django (Python)
- Angular (Typescript)
- Vue.js (JavaScript)

SUMMARY

- Programming vs Software Engineering
- Generic functionalities in Software Development
- Frameworks: what are they?
- How do Frameworks help with the Software Engineering Approach?
- Framework vs. Libraries
- Why use frameworks?
- Advantages and limitations of using frameworks
- Some examples

REFERENCES

1. Software Engineer vs. Programmer: What's the Difference?
2. Frameworks: Why They Are Important and How to Apply Them Effectively
3. What is a Web Framework, and Why Should I use one?
4. The Difference Between a Framework and a Library
5. What Is a Framework? (Definition and Types of Frameworks)
6. Software framework
7. Techopedia Explains Software Framework

THANK YOU

VISHAN.J@SLIIT.LK

NELUM.A@SLIIT.LK