AUTHORS
19K-0207 SARA JAMAL
19K-1333 PRERNA ALIAS PIYA KIRPLANI
19K-0226 MIDHAT KARIM

SECTION: BSCS 7A
INSTRUCTOR: MR. RAHIM HASNANI

FRONT CONTROLLER PATTERN

The front controller design pattern is a behavorial design pattern which is a useful approach for managing requests in an application. It provides a centralized point of access for handling all requests and can be particularly useful in web applications. The front controller acts as a gatekeeper, receiving incoming requests and forwarding them to the appropriate resources for processing. In addition to handling requests, the front controller can also perform authentication and authorization checks to ensure that only authorized requests are processed. This can make it easier to maintain and extend the application, as well as improve performance by allowing common request handling functionality to be implemented once and reused.

INTENT

Provide a centralized point of access for handling all requests in an application

OBJECTIVE

- Improve maintainability, extensibility, and performance by centralizing request handling and allowing common request handling functionality to be implemented once and reused.
- Improve security by providing a single point of access for implementing authentication and authorization checks.
- Provide a central mechanism for performing tasks such as authentication, authorization, logging and tracking, routing, and consistency.
- Particularly useful in web applications with a large number of request types or a complex request handling process.

MOTIVATION

- Centralize request handling in a single handler, which can make it easier to maintain and extend the application.
- Improve performance by allowing common request handling functionality to be implemented once and reused.
- Improve security by providing a single point of access for implementing authentication and authorization checks.
- Provide a consistent and flexible approach to request handling, particularly in applications with a large number of request types or a complex request handling process.
- Allow the application to be easily extended by adding new request handling functionality without changing existing code.

APPLICABILITY

The front controller design pattern is applicable in web, desktop and console applications where there is a need to handle requests in a centralized and consistent way. It is particularly useful in applications with a large number of request types or a complex request handling process.

ALSO KNOWN AS

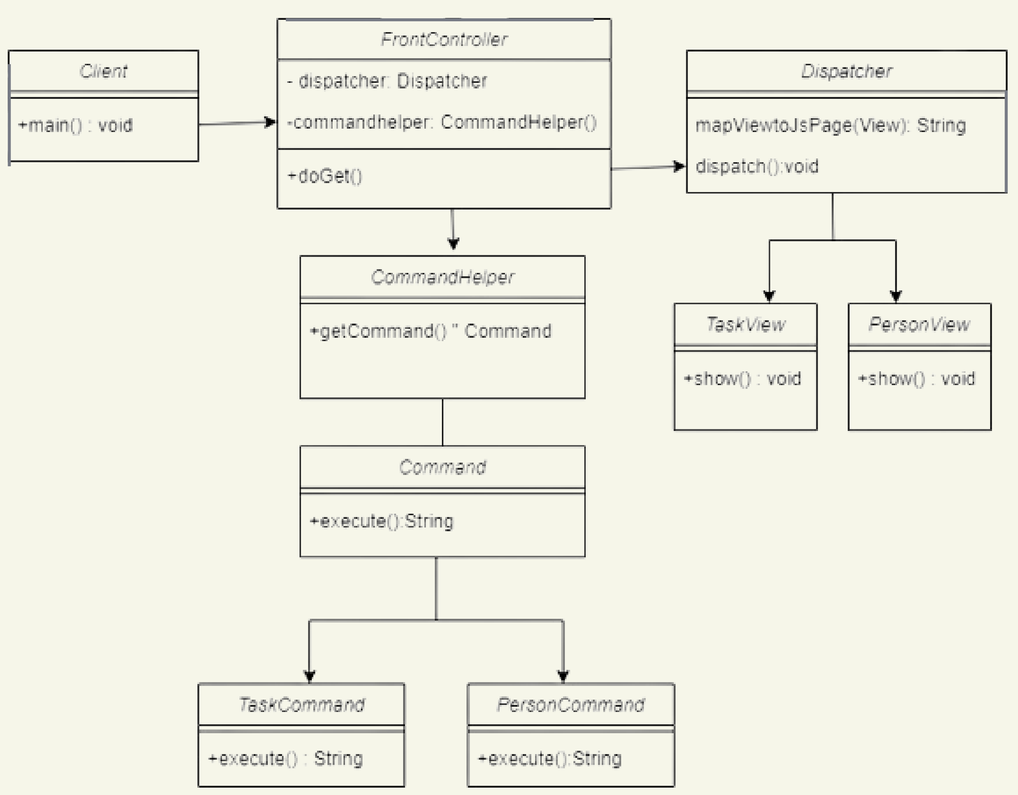
This pattern is also known as the "dispatcher pattern." The term "dispatcher pattern" comes from the fact that the front controller dispatches incoming requests to the appropriate resources for handling.

STRUCTURE

The front controller design pattern typically consists of the following components:

- **Front controller:** This is the central point of access for handling requests. It receives incoming requests and dispatches them to the appropriate resources for processing.
- **Dispatcher:** This component is responsible for managing view output and forwarding requests from front controller to the appropriate request handler.
- **Request handler:** This component is responsible for handling the processing of requests.
- **Helper:** This component can be used to pre-process incoming requests before they are forwarded to the front controller.
- **View:** This component is responsible for rendering the response to the client. It receives data from the request handler and generates the appropriate output

UML DIAGRAM



PROS & CONS

Pros:

1. Improved security: the front controller can act as a security gatekeeper.
2. Improved performance: front controller can cache common resources in data, reducing number of expensive calls.
3. Improved maintainability: changes to the application's request handling logic can be made in a single location.

Cons:

1. Increased complexity due to an additional layer of abstraction between the incoming request and the resources that handle it.
2. Increased development time: as it involves creating a new layer of abstraction in the application.
3. Reduced flexibility: as changes must be made in a single location.

CONSEQUENCES

Provides improved maintainability and extensibility of the application, as well as improved performance due to the reuse of common request handling functionality. However, it can also result in a more complex application structure, as well as a more tightly coupled system.

KNOWN USES

The pattern is commonly used in web application frameworks, such as Java's JSF and Spring MVC.

RELATED PATTERNS

Often used in conjunction with other design patterns, such as the MVC pattern and the command pattern.