An Architectural Design pattern proposal

The Router

Proposed by: Hiskel Kelemework

Id: Atr/0872/08

Proposed to: Natnael Argaw

Date: 30-03-18

Inspiration

Internet without a doubt is the biggest invention of the century, and the router is its core component. It interconnects devices and small and medium networks to form the larger internet. This core component, other than routing, can be used to implement security in networks whether in form of firewalls or simple address-based request/response blocking. This is the inspiration behind the ‘Router’ architectural design pattern (pattern not! Yet!). The ‘Router’ architectural design pattern uses the notion of a router to implement connections and tighten security between components.

Description

The ‘Router’ architectural design pattern has 2 component types

1. the router

2. the community

The router is the central component that regulates communication between components. All inter-component communications go through the router, which checks for any security restrictions about the communication and takes appropriate actions. The action could be to grant or forbid the communication.

The community is a set of components. The components could be client components or provider components. These components publish their security rules to the router about which other components they allow communications with. Client components can request services from providers and also communicate with other client components, whereas the provider components provide services for client components and also communicate with other providers. The pattern works by arranging components around the router to form the system or subsystem (if multiple routers used). Communications between components can happen in three ways:

1. Client-Client: can be used to pass messages between components

2. Client-Provider: can be used to request and receive services from providers

3. Provider-Provider: can be used to share common or complementary services

Advantages

1. allows for a logical flow of handling of cases

2. coding becomes relatively easier as whole system boils down to handling cases

3. centralized communication allows for ease of testing

4. allows easy grouping of components with complementary tasks

5. good scalability, by joining routers which have their own communities

6. handling security is easier, as components only have to specify which components they wish to communicate to (one way or both ways) and listen for requests or responses.

Component 1 (sender receiver) Component 2 sender only component Router Receiver only component