Testing and Error Handling

Michael Van Sickle

@vansimke

Overview

Testing handlers

Handling errors

Managing panics

```
import “net/http/httptest”

req := NewRequest(method, target string,
     body io.Reader)

rec := NewRecorder()
rec.Result()

srv := NewServer(handler http.Handler)

client := srv.Client()

srv = NewTLSServer(handler http.Handler)
```

◄ **Package with HTTP testing helpers**

◄ **Create a *http.Request for testing**

◄ **Create an http.ResponseWriter that records response**

◄ **Obtain a *http.Response to request**

◄ **Create server for testing handlers using HTTP over localhost**

◄ **Obtain an *http.Client that is configured to send requests to test server**

◄ **Secure servers can be created with temporary, automatically generated certificates**

Testing Handlers without HTTP

```
func TestWithoutHTTP(t *testing.T) {
  router := gin.New()
  router.GET("/users", func(c *gin.Context) {
    c.JSON(gin.H({
      "user1": "adent",
    }))
  })

  req := httptest.NewRequest("/users", http.MethodGet, nil)
  rec := httptest.NewRecorder()

  router.ServeHTTP(rec, req)

  res := rec.Result()

  // assertions go here
}
```

Testing Handlers with HTTP

```go
func TestWithoutHTTP(t *testing.T) {
  router := gin.New()
  router.GET("/users", func(c *gin.Context) {
    c.JSON(gin.H({
      "user1": "adent",
    }))
  })

  srv := httptest.NewServer(router)
  defer srv.Close()

  client := srv.Client()

  res, err := client.Get(srv.URL + "/users")

  // assertions go here
}
```

```
var c *gin.Context

var err *Error = c.Error(e error)

e.JSON() any
e.SetMeta(obj any) *Error
e.SetType(flags ErrorType) *Error
```

◀ **Attach error to current context**

◀ **Render error as JSON**

◀ **Add metadata to an error**

◀ **Set the error's type**

ErrorTypeBind – Context.Bind() failed

ErrorTypeRender – Context.Render() failed

ErrorTypePrivate – private error

ErrorTypePublic – public error

ErrorTypeAny – other error

- Can be combined with bitwise OR (|)

```
gin.Recovery()
```

◄ **Recover from panic that sends 500 response and records event in log**

```
gin.RecoveryWithWriter(out io.Writer)
```

◄ **Recover from panic that sends 500 response and records event to provided io.Writer**

```
gin.CustomRecovery(handle RecoveryFunc)
```

◄ **Recover from panic with custom behavior and records event in log**

```
gin.CustomRecoveryWithWriter(
    out io.Writer,
    handle RecoveryFunc)
```

◄ **Recover from panic with custom behavior and records event in provided io.Writer**

```
type RecoveryFunc func(
    c *Context,
    err any)
```

◄ **Signature for custom recovery function**

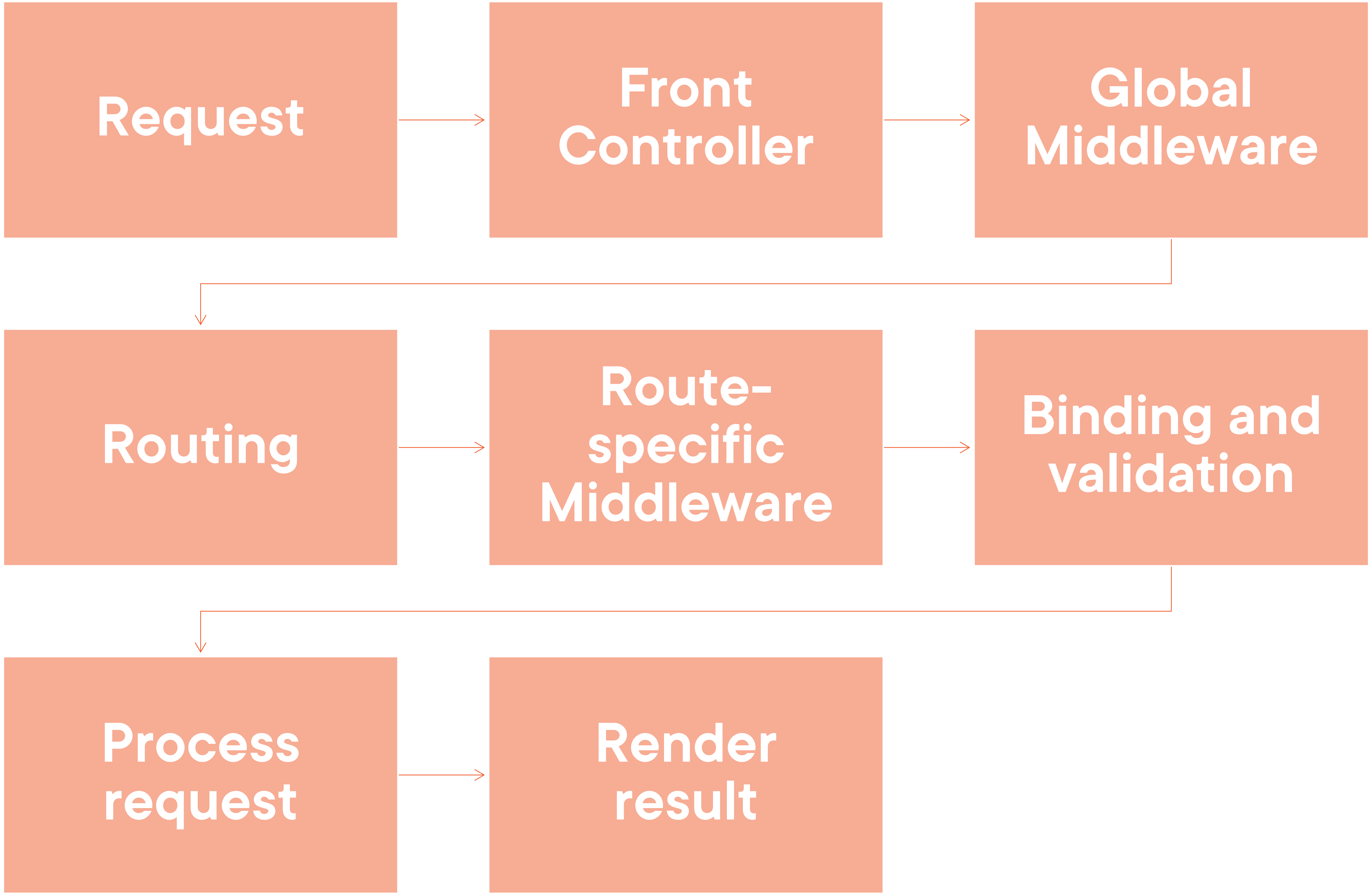
Summary

Testing handlers

Handling errors

Managing panics

HTTP Request Pipeline



Request

Front Controller

Global Middleware

Routing

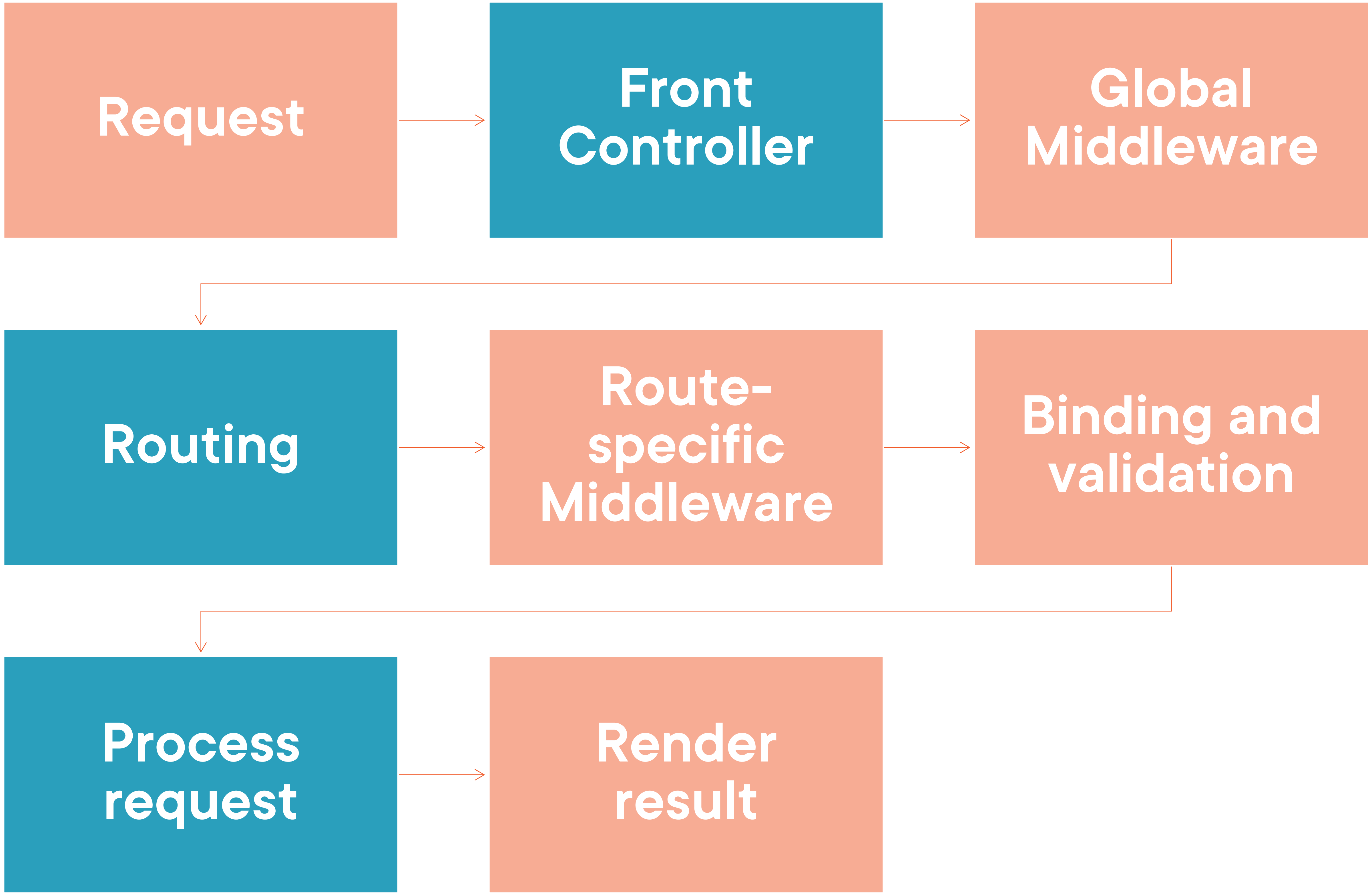
Route-specific Middleware

Binding and validation

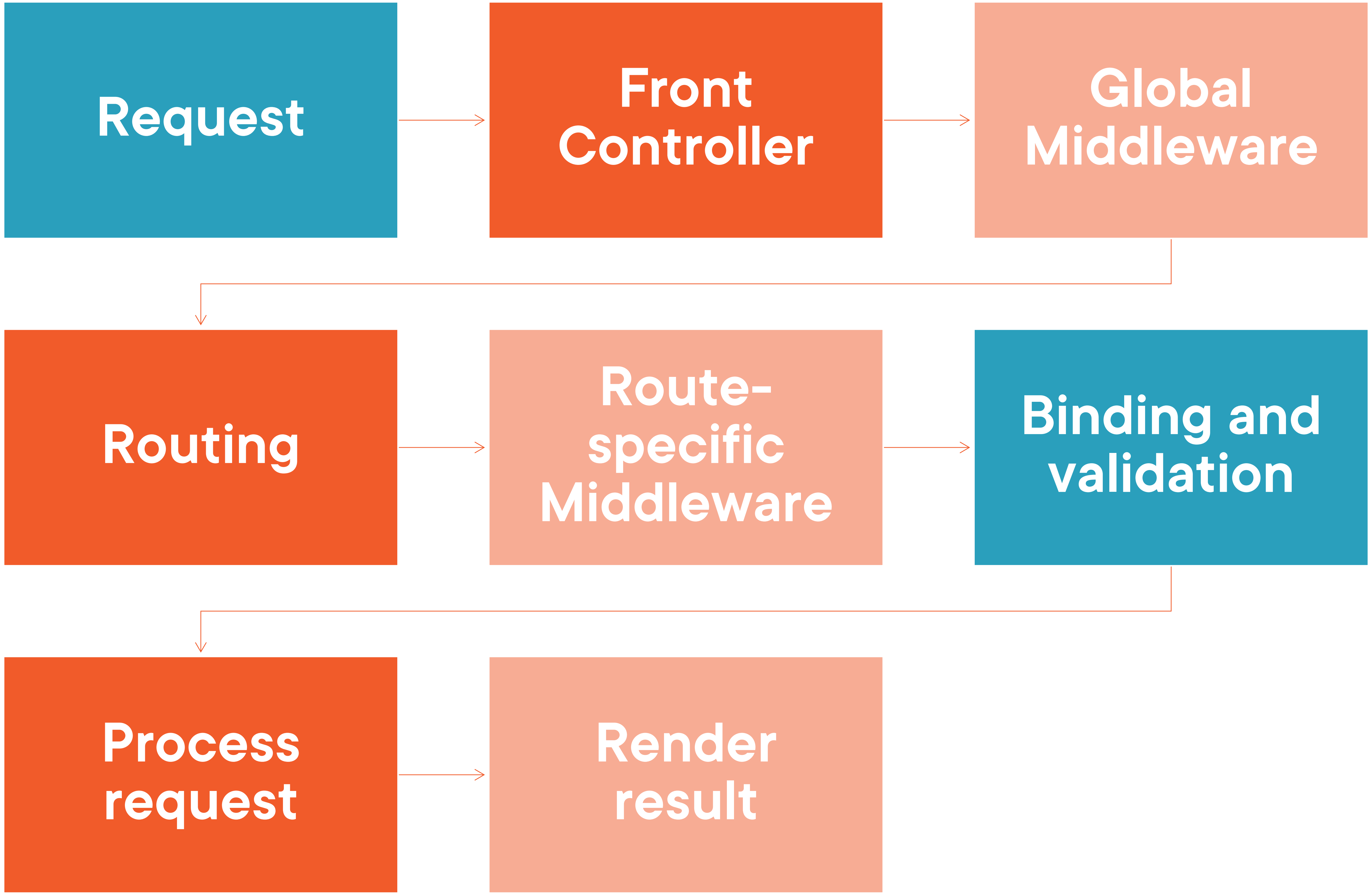
Process request

Render result

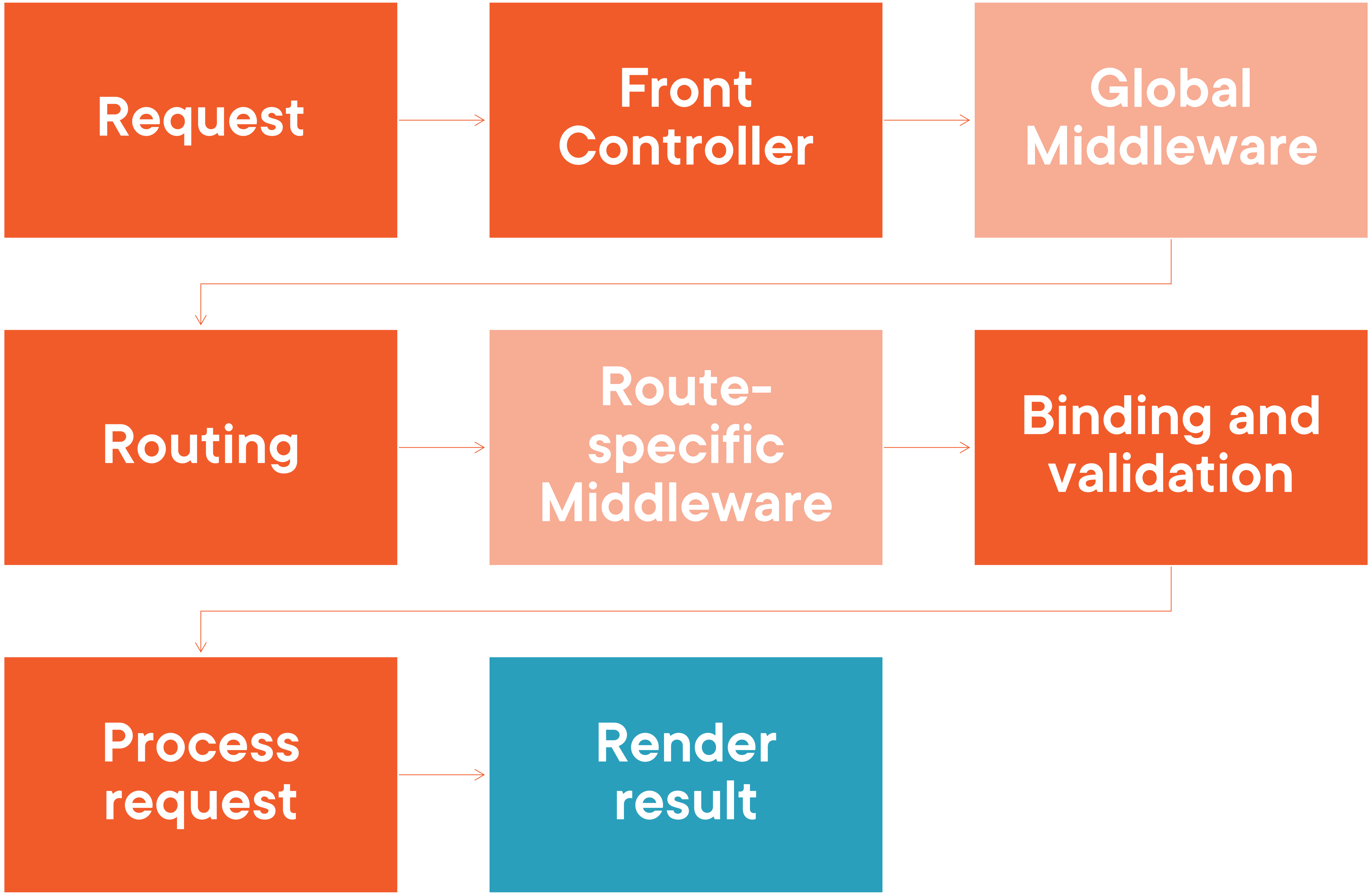
HTTP Request Pipeline



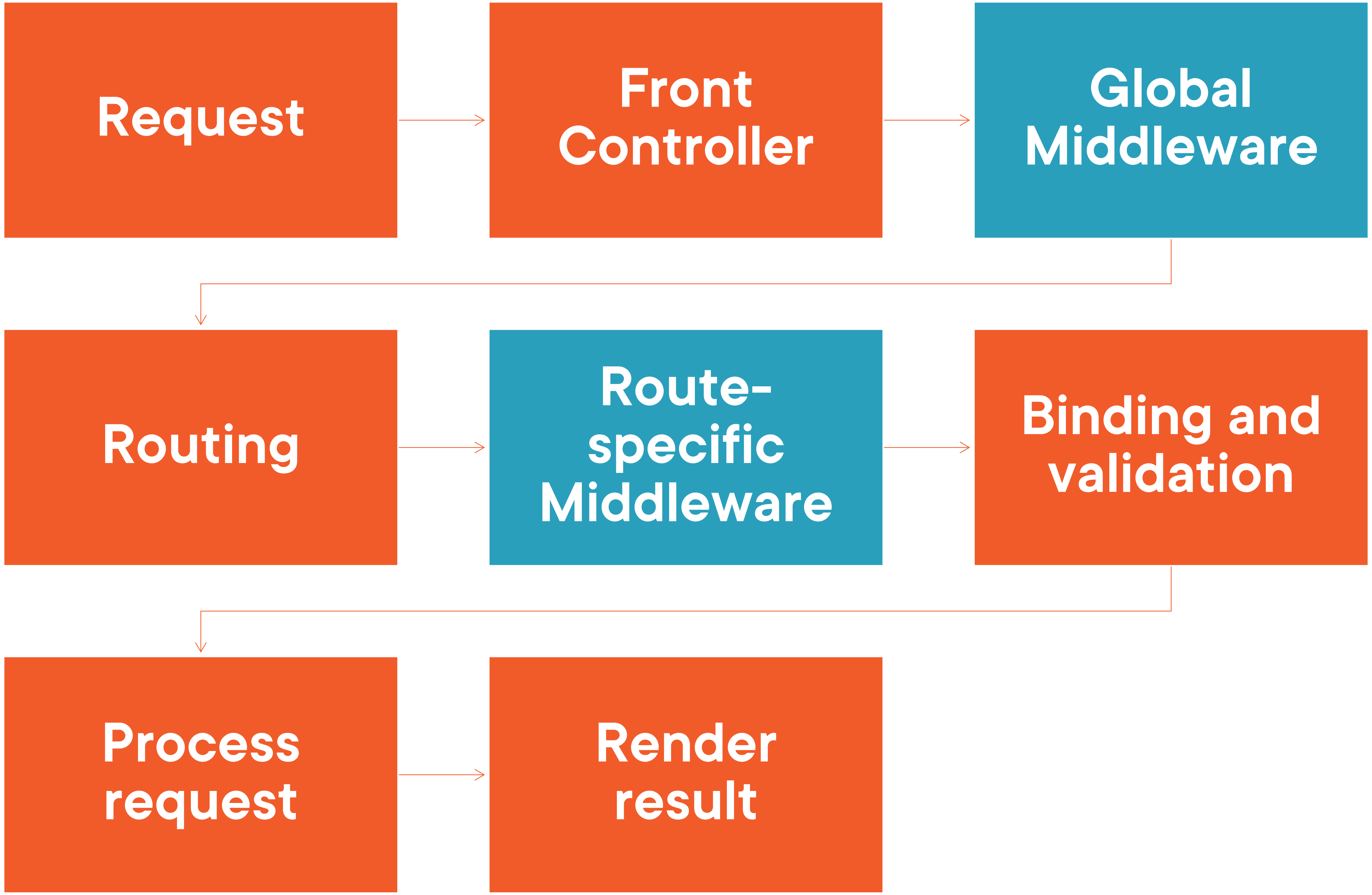
HTTP Request Pipeline



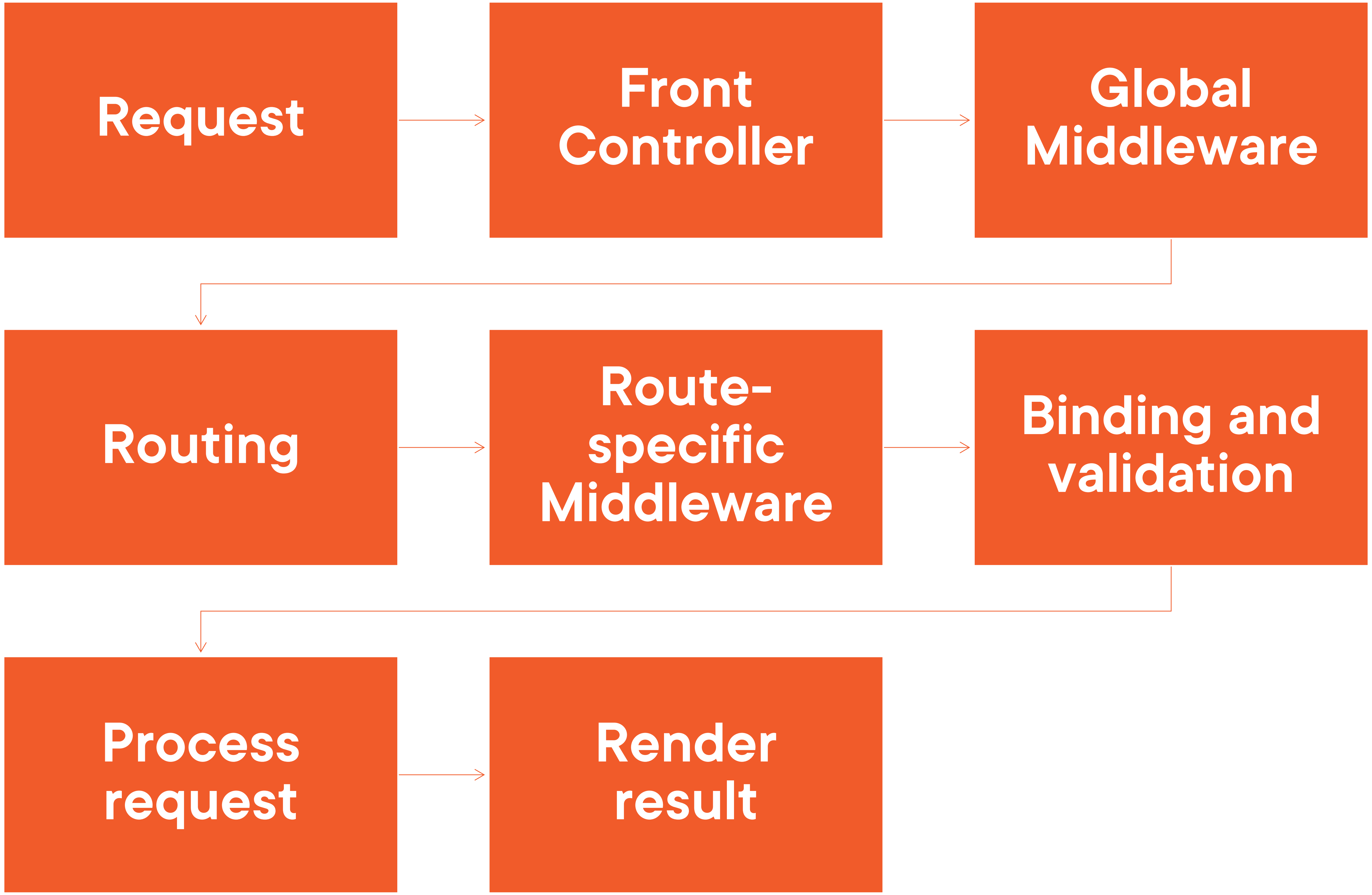
HTTP Request Pipeline



HTTP Request Pipeline



HTTP Request Pipeline



Request

Front Controller

Global Middleware

Routing

Route-specific Middleware

Binding and validation

Process request

Render result