CreatePaymentRequestHandler

MilMove Code

file: pkg/handlers/primeapi/api.go

```
primeAPI.PaymentRequestCreatePaymentRequestHandler = CreatePaymentRequestHandler{
   ctx,
   paymentrequest.NewPaymentRequestCreator(
      ctx.GHCPlanner(),
      ghcrateengine.NewServiceItemPricer(),
   ),
}
```

Initalize handler in the api.go file

file: pkg/**gen**/primeapi/primeoperations/mymove_api.go

```
o.handlers["POST"]["/payment-requests"] = payment_request.NewCreatePaymentRequest(
        o.context,
        o.PaymentRequestCreatePaymentRequestHandler
)
```

Initialize the Swagger API handler cache with the MilMove Go created handler to implement the Swagger API.

file: pkg/**gen**/primeapi/primeoperations/payment_request/create_payment_request.go

```
// NewCreatePaymentRequest creates a new http.Handler for the create payment request operation
func NewCreatePaymentRequest(ctx *middleware.Context, handler CreatePaymentRequestHandler) *CreatePaymentRequest {
   return &CreatePaymentRequest{Context: ctx, Handler: handler}
}
```

NewCreatePaymentRequest creates a new http.Handler for the create payment request operation

---

file: pkg/**gen**/primeapi/primeoperations/payment_request/create_payment_request.go

```
func (o *CreatePaymentRequest) ServeHTTP(rw http.ResponseWriter, r *http.Request) {
   ...

   res := o.Handler.Handle(Params) // actually handle the request
   o.Context.Respond(rw, r, route.Produces, route, res)

}
```

Calls the MilMove Go handler

Calls

MilMove Code

file: pkg/handlers/primeapi/payment_request.go

```
func (h CreatePaymentRequestHandler) Handle(params paymentrequestop.CreatePaymentRequestParams)
middleware.Responder {
   // this  function does not use a payload to model funtion
   //
   // call the servie object to project the request
   ...
   createdPaymentRequest, err := h.PaymentRequestCreator.CreatePaymentRequest(appCtx,
&paymentRequest)
   ...
   returnPayload := payloads.PaymentRequest(createdPaymentRequest)
   ...
   return paymentrequestop.NewCreatePaymentRequestCreated().WithPayload(returnPayload)
}
```

1. Handler will convert params from Swagger params to Go code base params somewhere near entry of handler funtion. (If it is being used, in most cases it will be).
2. Handler will call service objet to process the request. (If it is being used, in most cases it will be.)
3. Handler will convert from Go ode base params to Swagger params near the end of the handler for return to Swagger call.

Swagger World

file: pkg/mod/github.com/go-openapi/runtime@v0.21.0/middleware/context.go

```
// Serve creates a http handler to serve the API over HTTP
// can be used directly in http.ListenAndServe(":8000", api.Serve(nil))
func (o *MymoveAPI) Serve(builder middleware.Builder) http.Handler {
   o.Init()

   if o.Middleware != nil {
      return o.Middleware(builder)
   }
   if o.useSwaggerUI {
      return o.context.APIHandlerSwaggerUI(builder)
   }
   return o.context.APIHandler(builder)
}
```

file: pkg/mod/github.com/go-openapi/runtime@v0.21.0/middleware/context.go

```
// APIHandler returns a handler to serve the API, this includes a swagger spec, router and the contract defined in the swagger
spec
func (c *Context) APIHandler(builder Builder) http.Handler {
 b := builder
 if b == nil {
  b = PassthroughBuilder
 }

 var title string
 sp := c.spec.Spec()
 if sp != nil && sp.Info != nil && sp.Info.Title != "" {
  title = sp.Info.Title
 }

 redocOpts := RedocOpts{
  BasePath: c.BasePath(),
  Title:    title,
 }

 return Spec("", c.spec.Raw(), Redoc(redocOpts, c.RoutesHandler(b)))
}
```

file: pkg/mod/github.com/go-openapi/runtime@v0.21.0/middleware/context.go

```
// RoutesHandler returns a handler to serve the API, just the routes and the contract defined in the swagger spec
func (c *Context) RoutesHandler(builder Builder) http.Handler {
 b := builder
 if b == nil {
  b = PassthroughBuilder
 }
 return NewRouter(c, b(NewOperationExecutor(c)))
}
```

file: pkg/mod/github.com/go-openapi/runtime@v0.21.0/middleware/router.go

```
// NewRouter creates a new context aware router middleware
func NewRouter(ctx *Context, next http.Handler) http.Handler {
 if ctx.router == nil {
  ctx.router = DefaultRouter(ctx.spec, ctx.api)
 }

 return http.HandlerFunc(func(rw http.ResponseWriter, r *http.Request) {
  if _, rCtx, ok := ctx.RouteInfo(r); ok {
   next.ServeHTTP(rw, rCtx)
   return
  }

  // Not found, check if it exists in the other methods first
  if others := ctx.AllowedMethods(r); len(others) > 0 {
   ctx.Respond(rw, r, ctx.analyzer.RequiredProduces(), nil, errors.MethodNotAllowed(r.Method, others))
   return
  }

  ctx.Respond(rw, r, ctx.analyzer.RequiredProduces(), nil, errors.NotFound("path %s was not found", r.URL.EscapedPath()))
 })
}
```

file: pkg/**gen**/primeapi/primeoperations/payment_request/create_payment_request.go

```
func (o *CreatePaymentRequest) ServeHTTP(rw http.ResponseWriter, r *http.Request) {
   ...

   res := o.Handler.Handle(Params) // actually handle the request
   o.Context.Respond(rw, r, route.Produces, route, res)

}
```