# SecureWebServer.Core

## Entities

Bevat de *User* entiteit. Deze wordt gebruikt voor het inloggen.

## Error

Bevat de *IErrorHandler* interface voor een klasse die, in het geval dat er een exception optreed, de juiste HTTP response terug moet sturen. De implementatie van deze interface zit in de *SecureWebServer*.*Service*.

## Extensions

Niet heel belangrijk. In deze map staan extension methods die het leven wat makkelijker maken. Hier staat dus geen business logic in.

## Helpers

Net als bij de *Extensions* bevat deze map algemene klassen om het programmeren wat makkelijker te maken. Deze konden alleen niet als extension method gebruikt worden.

## Request

De *RequestMessage* is het request wat door de client is gestuurd. Door met dit object te werken hoef je niet heel de tijd met sockets en stream te werken maar gewoon met deze message.

De *IRequestHandler* interface is voor een klasse die een request moet afhandelen nadat deze is geparsed naar een *RequestMessage* object. De IRequestHandler geeft uiteindelijk eenresponse terug die naar de client gestuurd moet worden. De implementatie hiervan zit in de *SecureWebServer.Service*.

De *RequestListener* luistert op een aparte thread naar nieuwe connecties. Deze worden dan omgezet naar een *RequestMessage* en afgehandeld op een aparte thread door de *IRequestHandler*.

De *RequestException* wordt gegooid op momenten waarin je bewust een foutmelding met status code naar de client wilt sturen, bijvoorbeeld als een pagina niet kan worden gevonden of een gebruiker geen toegang heeft.

## Response

De *ResponseMessage* is het object wat door een *IRequestHandler* of *IErrorHandler* implementatie wordt teruggestuurd. De *RequestListener* zorgt er vervolgens voor dat deze naar de client wordt gestuurd.

# SecureWebServer.DataAccess

Bevat code voor het ophalen en opslaan van data naar de database.

## Repositories

Bevat de *UserRepository* voor het ophalen en opslaan van de users in een SQL database. Deze users worden gebruikt voor het inloggen.

# SecureWebServer.Service

Dit is de daadwerkelijke webservice. Dit is een console applicatie.

## De RequestHandler implementatie

Dit is het begin punt van de afhandeling van een request. Voor normale request waarbij bestanden worden opgehaald of directories worden uitgelezen zal de dat door deze klasse zelf gedaan worden. Andere request waarbij meer logica komt kijken worden afgehandeld door command handlers om het overzichtelijk te houden.

## De ErrorHandler implementatie

Deze stuurt de juiste error pagina terug als deze bestaat voor de status code. Bestaat deze niet, dan wordt er gewoon een foutmelding in plain text teruggestuurd.

## De Logging

Logging wordt gedaan d.m.v. log4net. Deze logt naar een bestand toe. De logging staat geconfigureerd in *log4net.config*.

## De database

De database die gebruikt wordt is een fysiek bestand die bij het builden wordt mee gekopieerd. Dit is het *securewebserver\_db.mdf* bestand.

## CommandHandlers

In het *CommandHandlers* mapje staan de volgende command handlers:

* ConfigCommandHandler  
  Toont de configuratie pagina en zorgt voor het verwerken van wijzigingen aan de configuratie.
* IndexCommandHandler  
  Zorgt voor het opbouwen en tonen van de HTML.
* LogCommandHandler  
  Toont het logbestand en geeft de mogelijkheid deze te wissen.
* LoginCommandHandler  
  Toont de login pagina en handelt het inloggen af.
* LogoutCommandHandler  
  Handelt het uitloggen af.
* UserEditCommandHandler  
  Toont de bewerk pagina voor een gebruiker en handelt wijzigingen aan een gebruiker af.
* UserOverviewCommandHandler  
  Toont een overzichtpagina met gebruikers die in de database staan.

De *RequestHandler* roept de *CommandHandlerFactory* aan om de juiste handler aan te maken. De factory kijkt simpelweg naar de URL die wordt aangeroepen en maakt daarvoor de juiste handler aan. Als de *RequestHandler* geen command handler terug krijgt zal deze het request op de standaard manier afhandelen: terugsturen van een bestand of het browser van een directory (afhankelijk van de URL).

## Config

De *ServerConfiguration* klasse is de configuratie die door de webserver wordt gebruikt. Deze configuratie is vanuit je webbrowser aan te passen via de config pagina, als je daar de rechten voor hebt natuurlijk.

Settings worden uitgelezen en opgeslagen in een JSON bestand. Wanneer de settings worden opgeslagen wordt er een callback gedaan die er voor zorgt dat de server herstart met de nieuwe configuratie (dus ook de nieuwe port, als deze is gewijzigd).

## ErrorPages

Hier staan statische error pagina’s in die door de *ErrorHandler* worden teruggestuurd. Je kunt hier simpelweg een pagina toevoegen en deze wordt automatisch gebruikt, als de naam overeenkomt met de statuscode van een error.

## Security

De *SecurityProvider* bevat alle logica voor de authenticatie en autorisatie van een gebruiker. De rollen kunnen per URL geconfigureerd worden in de *app.config* van de webserver. De *RequestHandler* zal altijd eerst kijken of de client toestemming heeft om het request uit te voeren door van de *SecurityProvider* gebruik te maken.

## WebRoot

Dit is de standaard webroot folder met pagina’s en bestanden die gebruikt wordt. Als de applicatie start zal deze standaard van deze webroot gebruik maken. Dit is aan te passen in je browser door naar de configuratie pagina te gaan.

## Info

In dit mapje staat wat info, waaronder dit document.

**TestUsers.txt** bevat een lijstje met standaard users en wachtwoorden die nu in de database zitten.

**OWASPtop10.txt** legt uit hoe de OWASP top 10 is toegepast op de webserver.