Architectuur thywin

Inhoud

Thywin Protocol

Voor de interne communicatie tussen onze applicaties hebben we ons eigen protocol ontworpen. Dit is een protocol gebaseerd op laag 4 van het OSI model.

Het Thywin protocol is een verzameling van afspraken over hoe informatie tussen het Master, Crawler en Parser proces wordt uitgewisseld. Dit is samengevoegd in een Thywin pakket.

Het Thywin protocol maakt deel uit van de Thywin library, wat een gemeenschappelijke basis bied voor de cliënt & servers binnen de processen.

Thywin packet

Een Thywin packet is het pakket dat overgezonden wordt binnen het Thywin protocol.

Dit pakket bevat de volgende informatie:

- Methode
- Type
- Content

Aan de hand van deze informatie kan er worden bepaald wat een cliënt/server wil en wat voor een soort informatie over wordt gestuurd.

Een Thywin pakket is als volgt opgebouwd:

METHODE | HEADER_SEPERATOR | TYPE | HEADER_SEPERATOR | CONTENT | END_OF_PACKET

De header is afgeschermd met HEADER_SEPERATOR's. Het einde van een pakket is gemarkeerd met een END_OF_PACKET. Dit zijn speciale ASCII karakters.

Methode

Er zijn drie verschillende methode voor een Thywin packet:

- GET
- PUT
- RESPONSE

Aan de hand van deze methodes, wordt gekeken wat de sender van dit pakket wil.

Type

Er zijn 6 verschillende types voor een Thywin packet:

- URI
- DOCUMENT
- RELEVANCE
- DOCUMENTVECTOR

- URIVECTOR
- SEARCH_RESULTS

Door middel van het type kan worden bepaald wat voor een informatie wordt overgestuurd of opgevraagd wordt. Dit is gelinkt aan de methode.

Content

De content is een klasse die ThywinPacketContent implementeert. Binnen de Thywin Lib zijn er 4 verschillende soorten die overgestuurd worden:

- URIPacket
- DocumentPacket
- DocumentVectorPacket
- URIVectorPacket

Al deze classes hebben een implementatie voor een functie Serialize. Deze methode wordt aangeroepen op het moment dat er een Thywin packet wordt verstuurd via de library.

Verdere uitleg over de verschillende content classes binnen het Thywin Protocol is beschreven in het hoofdstuk Thywin Library.

Thywinlib library

Alle applicaties maken gebruik van een gedeelde library. Deze library bevat onderdelen die in meerdere applicaties nodig zijn waaronder de implementatie van het protocol, dat hierboven is beschreven.

Communicator

De communicator binnen de thywinlib verzorgt het versturen en ontvangen van Thywin pakketten voor cliënt. Deze communicator gebruikt het Thywin protocol om te communiceren via sockets.

URIPacket

Een URI packet bestaat uit:

- Relevantie van de URI
- De URI zelf.

De relevantie binnen dit pakket is de relatieve relevantie. Dit betekend dat het de relevantie is van de pagina waarop de URI is gevonden.

Aan de hand hiervan wordt de prioritiet binnen de URI queue bepaald. Meer uitleg hierover is te vinden in het hoofdstuk Master.

DocumentPacket

Het document pakket is het pakket dat overgestuurd om een opgehaald document over te sturen naar een andere applicatie. Het bestaat uit:

- De URI van waar het document vandaan komt
- Het document zelf

DocumentVectorPacket

Een document vector pakket is een verzameling pakket voor een index, relevantie en de locatie hiervan.

- URI van de index
- Relevantie
- Index

In tegenstelling tot een URI pakket, is de relevantie hierbij de relevantie van het document zelf.

De index is een Documentvector class. Dit zijn de woorden die voorkomen op het document met een aantal hoe vaak dit woord voorkomt.

URIVectorPacket

Dit pakket is een verzameling van URI pakketten. Het is bedoelt om meerdere URIs tegelijkertijd over te sturen naar een ander proces. Het bevat geen extra informatie.

Document Vector

Een ander onderdeel in de library is de class documentvector. De documentvector class wordt gebruikt voor het berekenen van de gelijkheid van 2 documenten.

De gelijkheid berekening wordt aan de hand gedaan van een cosine similarity algoritme, zoals hieronder beschreven.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Dit algoritme levert een getal op, dit is de relevantie van het document tegenover het onderwerp. Als de relevantie 1 is het document 100% gelijk aan het onderwerp en als de relevantie 0 is komt het onderwerp niet voor in het document.

Crawler applicatie

De crawler heeft 1 hoofdtaak namelijk: documenten in de documentqueue zetten. Dit doet de crawler door na het starten aan de master een URI te vragen en die te crawlen. Nadat het document is binnengekomen wordt het naar de documentqueue gestuurd.

Tijdens het crawlen wordt er gebruikt gemaakt van CURL. CURL is een proces dat gestart wordt en de http header en de content kan downloaden.

Het crawler proces verwerkt deze data vervolgens. Eerst wordt de http header uitgelezen. Er wordt gekeken of er geen 3xx Redirection http code terug komt. Is dit wel het geval dan wordt er gezocht naar de redirect URI.

Deze redirect URI wordt terug gestuurd naar de master met een PUT URI pakket. De content van dit pakket bevat de redirect URI en de relevantie van de initiële URI die verkregen is van het GET URI pakket.

Als er geen 3xx redirection http code terug komt, wordt er gecontroleerd of de http header het goede content type bevat. Er is voor gekozen om alleen “text/html” te accepteren.

Indien het content type hier niet gelijk aan is, wordt er niks gedaan met de URI. Wanneer het content type wel correct is wordt er een thywinpacket met de waardes PUT DOCUMENT gemaakt. Dit pakket wordt verstuurd naar de document queue.

Parser applicatie

De parser heeft 2 hoofdtaken namelijk: URI’s zoeken in een document en deze geven aan de master en een documentvector maken van het document en ervoor zorgen dat deze wordt opgeslagen in de database. De parser begint door aan de document queue een document te vragen. Uit dit document worden dan eerst de URI’s gehaald daarna word de tekstuele content uit het document gehaald en wordt hiervan een documentvector gemaakt en deze wordt opgeslagen.

Searchengine applicatie

De searchengine heeft 1 hoofdtaak namelijk: het beantwoorden van zoek requests met een lijst met documentvectoren. De searchengine dient vooral als tussenlaag tussen de webapplicatie (geschreven in PHP) en de Postgresql database. De searchengine wacht op een verzoek en zal deze beantwoorden met de meest relevante resultaten bij dat verzoek.

De master

De master is het brein van de webcrawler. De master communiceert met de crawler(s) en de parser(s) en slaat alle data op die binnenkomt in een database.

Voor het opstarten van de Master zijn er de volgende argumenten:

- Port
- Blacklist file (optioneel)

Dit is de port waar de Master server bereikbaar op is voor de Parser en Crawler. De blacklist is een lijst met URI's of delen van URI's die ingelezen wordt. Alle binnenkomende URIs worden tegen deze blacklist gehouden en als er een match is zal deze niet worden mee genomen naar de queue. De gedetailleerde werking van de blacklist wordt in een later hoofdstuk beschreven.

Server Interface

De master heeft een multi-threaded server. Wanneer een Parser of Crawler zich meld, zal er een verbinding opengehouden worden.

Er kunnen maximaal 127 verbindingen worden opgezet en opgehouden tegelijkertijd. Dit houdt in dat er maximaal 127 Crawlers/Parsers tegelijkertijd informatie kunnen uitwisselen met het Master proces.

De volgende requests worden afgehandeld door de Master:

- GET URI
- PUT URI
- GET Document
- PUT Document
- PUT DocumentVector
- PUT URIVector

GET URI

Wanneer de Master een GET URI request binnen krijgt, wordt er als eerste gekeken of de database leeg is. Indien deze leeg is, wordt deze aangevuld met de default URI's, die tevens het start punt zijn.

Hierna wordt er gekeken of de in memory queue leeg is. Indien deze queue leeg is, worden er maximaal 50 URI's uit de database gehaald. Deze worden daarna in de memory queue gezet.

Vervolgens wordt er een URI Packet uit de queue gehaald en wordt deze terug gestuurd naar de client via het Thywin Protocol.

Het terug gestuurde Thywin packet is als volgt opgebouwd:

- Method: RESPONSE

- Type: URI
- CONTENT: URIPacket

GET Document

Wanneer de Master een GET Document request binnen krijgt, wordt er als eerste gekeken of de DocumentQueue Semaphore tenminste 1x een request door kan laten. Dit is het geval als er tenminste 1 document in de queue staat.

Als dit niet het geval is, zal de request blocking zijn tot er een nieuwe document in de queue staat.

Na de semaphore wordt er een document uit de database/queue gehaald en wordt deze terug gestuurd naar de client via het Thywin Protocol.

Het terug gestuurde Thywin packet is als volgt opgebouwd:

- Method: RESPONSE
- Type: URI
- CONTENT: DocumentPacket

PUT URI

Bij een PUT URI request wordt er gekeken of de URI al voorkomt in de URI lijst op de database. Indien dit niet het geval is, wordt deze toegevoegt aan de URI Queue.

Als de URI al in de lijst voor komt, wordt er genoteerd dat de URI is voor gekomen. Dit is een teller die elke keer wordt opgehoogd wanneer de URI binnen komt.

Er wordt geen response gegeven aan de client.

PUT Document

Bij een PUT Document request wordt er gekeken of de URI waar het document vandaan komt al voorkomt in de URI lijst op de database. Indien dit niet het geval is, wordt er niks gedaan met het document.

Als de URI al in de lijst voor komt, wordt het document opgeslagen in de database. Ook wordt er een semafoor opgehoogd met 1, om zo aan te geven dat er een nieuw document beschikbaar is voor de GET Document requests.

PUT DocumentVector

Bij een PUT DocumentVector wordt de data van een document opgeslagen in de database. De DocumentVector bevat keywords met een getal van het aantal keren dat het woord voorkomt in een document, ook wel de occurrence . Elk keyword met bijbehorende uri_id en occurrence wordt opgeslagen in de indices tabel.

PUT URIVector

Bij een PUT URIVector worden er meerdere URI's opgeslagen in de database. Hierbij wordt hetzelfde het proces voor een PUT URI herhaalt voor elke URI en in een pakket verstuurd.