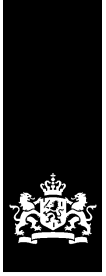


Ministerie van Binnenlandse Zaken en Koninkrijksrelaties

Technisch ontwerp BRP Bijhouden

Versie 0.1

Datum 02-03-2017
Status Concept

Inhoudsopgave

Documenthistorie

Versiehistorie

Versienummer	Datum	Auteur	Aanpassingen
0.1	02-03-2017	Operatie BRP	Eerste opzet.

Reviewhistorie

Versienummer	Datum	Namen
0.1	14-03-2017	Operatie BRP

1 Inleiding

Dit document dient ter onderbouwing en uitleg van de software die is ontwikkeld voor de functionaliteit van de Bijhouding zoals deze is beschreven in het Use Case Model (UCM) [5]. De lezer dient bekend te zijn met de inhoud van het Software Architectuur Document (SAD) [1] en is voornamelijk bedoeld voor software ontwikkelaars.

1.1 Scope

De inhoud van dit document richt zich op het 'gat' tussen het SAD en de broncode. Het is niet de bedoeling om informatie uit het SAD of broncode in dit document de dupliceren. Het is voor de lezer daarom belangrijk om bij het lezen van dit document kennis te hebben van het SAD en toegang te hebben tot de broncode zodat deze drie in samenhang kunnen worden bestudeerd.

Het 'gat' waar dit document zich op richt is het volgende:

- software van alleen de Bijhouding (exclusief bericht ontwerp en database ontwerp);
- het waarom achter de broncode;
- de kaders waarbinnen de software is ontwikkeld;

Het technisch ontwerp van de berichten in de vorm van XSD's [2] en het ontwerp van de BRP database [3] worden elk in een eigen document beschreven en vallen buiten de scope van dit document.

1.2 Referenties

#	Document	Versie	Datum
1	SAD BRP Voorziening		
2	Technisch ontwerp Berichten		
3	Technisch ontwerp Database		
4	Technisch ontwerp BRP Leveren		
5	UCM BY.0 - Bijhouding		
6	UCS-BY.1.ON - Ontrelateren		

Tabel 1 Lijst met referenties naar andere documenten

1.3 Leeswijzer

Deze paragraaf beschrijft de hoofdstukken uit het technisch ontwerp om de lezer te ondersteunen bij het vinden van de juiste informatie.

- Hoofdstuk 1 (Inleiding) beschrijft het doel en scope van dit document en welke informatie men in dit document kan vinden.
- Hoofdstuk 2 (Context) maakt inzichtelijk wat de context van de Bijhouding is en maakt een link met het Software Architectuur Document (SAD).
- Hoofdstuk 3 (Modules) beschrijft de verschillende modules van Bijhouding, gelaagdheid en relevantie classes.
- Hoofdstuk 4 (Model) bevat een toelichting op het domein model van de Bijhouding software.
- Hoofdstuk 5 (Processen) beschrijft de processen die binnen de Bijhouding worden uitgevoerd.
- Hoofdstuk 6 (Ontwerpbeslissingen) somt de ontwerpbeslissingen op voor Bijhouding als aanvulling op het SAD.

2 Context

De bijhouding software zorgt ervoor dat persoonsgegevens worden bijgehouden in de BRP database. Hiervoor biedt de bijhouding twee koppelvlakken, één koppelvlak voor BRP bijhoudingen en één voor GBA bijhoudingen. Het koppelvlak voor BRP bijhoudingen verwerkt berichten van de burgerzaken modulen op synchrone wijze d.m.v. een SOAP webservice. Ter ondersteuning van de verwerking van toevallige gebeurtenissen in de migratievoorziening kunnen zogenaamde GBA bijhoudingen op asynchrone wijze worden verwerkt d.m.v. JMS queues.

De door de bijhouding gemaakte notificatieberichten worden aan de Verzending aangeboden door deze op de daarvoor bestemde JMS queue te plaatsen. Het versturen van deze notificatieberichten wordt vervolgens afgehandeld door het Verzending component [4].

Deze wijze waarop de bijhouding gaat archiveren staat nog ter discussie en is daarom hier niet beschreven.

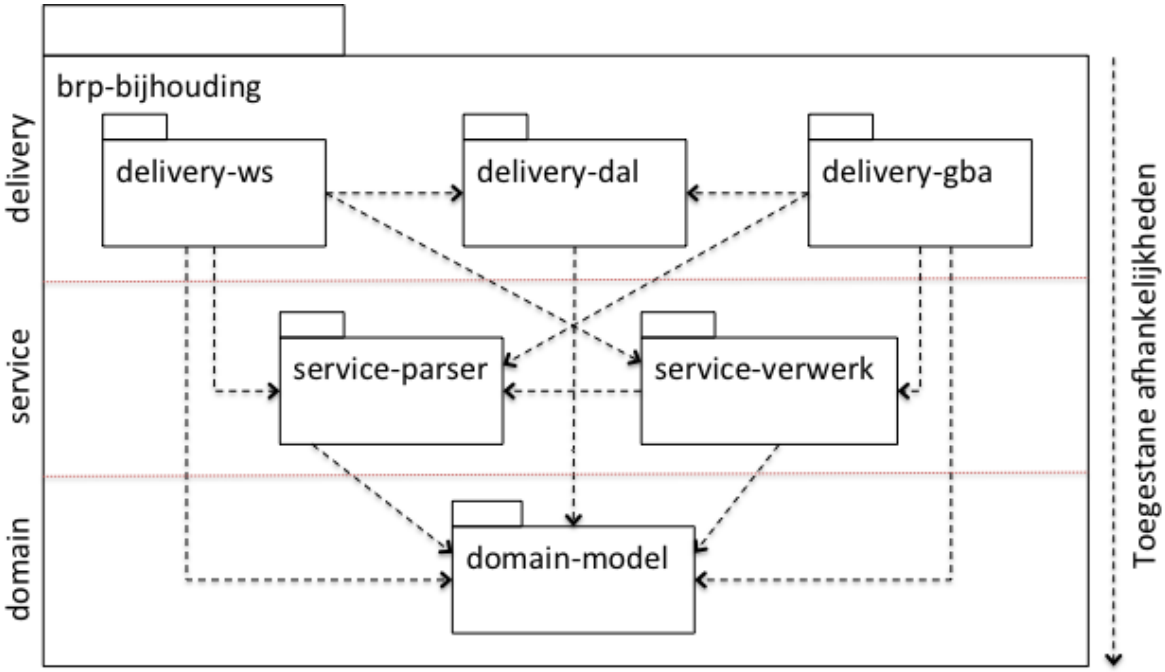
Onderstaande afbeelding toont de bijhouding in de context zoals deze hierboven beschreven is.

Afbeelding 1: Context Bijhouding Software

Deze afbeelding is een vereenvoudigde weergave van de bijhouding software in haar context en maakt geen onderscheidt tussen modules (logische groepering van broncode) en componenten (runtime gedrag). Deze worden in meer details beschreven in hoofdstuk 3 (Modules).

3 Modules

De bijhouding software is verdeeld over verschillende modules op een zodanige wijze dat de broncode die bij elkaar hoort, gegroepeerd en gescheiden is van de andere broncode. De modules zijn verdeeld in verschillende lagen waarbij modules binnen één laag van elkaar afhankelijk mogen zijn. Daarnaast mag een module ook afhankelijk zijn van een module in de laag eronder. Dit is schematisch weergegeven in afbeelding 2.



Afbeelding 2: Lagen architectuur van de Bijhouding software

De *delivery* laag bevat modules met afhankelijkheden van specifieke technologieën waar onderliggende lagen niet van afhankelijk willen zijn. Bijvoorbeeld Apache CXF voor JAX-WS, Hibernate als object relational mapping (ORM) technologie en Apache ActiveMQ als middleware implementatie. De *service* laag bevat de applicatie services van de bijhouding. De onderste laag bestaat uit de *domain* laag. Hierin bevinden zich de modules met de kern logica van het systeem. Voor de bijhouding bestaat deze logica uit validatie van berichten en verwerking in de database.

Het doel van deze opsplitsing in modules is het beteugelen van de complexiteit van de software door het groot stuk software op te splitsen in kleinere (minder complexe) stukken. Ook wel bekend als "separation of concerns". Door bovendien de extra gelaagdheid te introduceren ontstaat een nog betere scheiding van verantwoordelijkheden en worden afhankelijkheden met externe technologieën beter beheersbaar. Zo wordt het bijvoorbeeld erg eenvoudig om bedrijfsregels te unit testen in de domain modules zonder dat er kennis nodig is van JAX-WS, ORM of andere specifieke technologieën.

Van de verschillende modules volgt hieronder een beschrijving van hun verantwoordelijkheden.

3.1 BRP koppelvlak (brp-bijhouding-delivery-ws)

Deze module is verantwoordelijk voor het ontvangen en verwerken van SOAP berichten met BRP XML als body. Deze webservice is gebouwd met JAX-WS en Apache CXF. Er wordt gebruik gemaakt van de brp-bijhouding-service-parser module om binnenkomende BRP XML berichten om te zetten in objecten uit het domein model en omgekeerd. Verdere verwerking van het

bijhoudingsvoorstel wordt afgehandeld door de brp-bijhouding-service-verwerk module die door deze module wordt aangeroepen. Het bouwen van deze module met Apache Maven levert een WAR-bestand op dat gedeployed kan worden binnen een Apache Tomcat server.

3.2 GBA koppelvlak (brp-bijhouding-delivery-gba)

Het GBA koppelvlak lijkt erg veel op het BRP koppelvlak maar biedt geen SOAP interface. In plaats daarvan worden de BRP XML berichten van een queue gelezen en wordt het antwoord bericht op een antwoord queue weggeschreven.

3.3 Data Access (brp-bijhouding-delivery-dal)

Deze module bevat een JPA implementatie op basis van Hibernate voor de bijhouding entiteiten uit het model. Deze implementatie is gescheiden van het model zodat deze geen kennis heeft van hoe entiteiten gelezen en opgeslagen worden.

3.4 XML Parser (brp-bijhouding-service-parser)

Voor het mappen van BRP XML berichten op objecten uit het bijhouding model is een parser gemaakt o.b.v. de StAX parser. Deze module bevat ook een writer die de bijhouding objecten kan terug vertalen naar XML.

3.5 Verwerking (brp-bijhouding-service-verwerk)

De verschillende applicatie services van de bijhouding horen in deze module thuis. De bijhouding software kent de volgende applicatie services:

3.5.1 AutorisatieService

Controleert of er een geldige toegang bijhoudingsautorisatie bestaat voor de verzendende partij die het bijhoudingsvoorstel heeft ingediend.

Input: Het BijhoudingVerzoekBericht.
Output: Een lijst met meldingen. Een lege lijst betekent dat er geen autorisatie fouten zijn opgetreden. Een gevulde lijst betekent dat de verwerking niet door mag gaan.

3.5.2 BijhoudingAntwoordBerichtService

Maakt het antwoord bericht voor de bijhouding service.

Input: Het BijhoudingVerzoekBericht, de lijst met meldingen, administratieve handeling en bijhoudingsplan context
Output: Het BijhoudingAntwoordBericht.

3.5.3 BijhoudingService

Deze service biedt twee methodes voor het verwerken van een BijhoudingVerzoekBericht, één methode voor het verwerken van een bericht dat is binnengekomen op het BRP koppelvlak en de andere methode voor een bericht dat is binnengekomen op het GBA koppelvlak.

Input: Het BijhoudingVerzoekBericht.
Output: Het BijhoudingAntwoordBericht.

3.5.4 BijhoudingsplanService

Deze service maakt het bijhoudingsplan o.b.v. het ontvangen verzoek bericht.

Input: Het BijhoudingVerzoekBericht.
Output: De BijhoudingsplanContext.

3.5.5 MutatieNotificatieService

Deze service verstuurt het bijhouding notificatiebericht.

Input: Het BijhoudingsplanNotificatieBericht.
Output: Geen output.

3.5.6 OntrelateerService

Deze service voert de ontrelateer functionaliteit[6] uit.

Input: De BijhoudingsplanContext en BijhoudingVerzoekBericht.
Output: Geen output.

3.5.7 ValidatieService

Deze service wordt gebruikt om het BijhoudingVerzoekBericht te valideren. Daarnaast biedt deze service functionaliteit om aan de hand van een lijst meldingen te bepalen of de verwerking door mag gaan en wat het hoogste melding niveau is.

Input: Het BijhoudingVerzoekBericht.
Output: Een lijst met 0, 1 of meerdere meldingen.

3.6 Model (brp-bijhouding-domain-model)

Deze module bevat de kern functionaliteit van de bijhouding. Deze domein classes bestaan vooral uit representaties van de bericht elementen en de bijbehorende functionaliteit om de bericht elementen te valideren en te verwerken.

Vanwege het grote aantal bedrijfsregels is gekozen voor een model waarin deze bedrijfsregels makkelijk kunnen worden uitgedrukt. Mede om deze reden heeft de bijhouding een specifiek model waarin elke element bij een bijhouding verzoek bericht wordt gemapped op een specifieke Java class waarin vervolgens de functionaliteit m.b.t. dit element kan worden toegevoegd. Op deze manier zit de controle van gegevens letterlijk op de gegevens zelf. In hoofdstuk 4 (Model) wordt het model in detail toeglicht.

4 Model

TODO: Class model van de bijhouding toelichten.

5 Processen

TODO: component & connector view.

6 Ontwerpbeslissingen

Hier staan ontwerpbeslissingen gedocumenteerd die gelden voor de alle modules binnen de bijhouding software. Deze ontwerpbeslissingen kunnen per module worden aangevuld met additionele ontwerpbeslissingen.

6.1 Minimaliseer afhankelijkheden met specifieke implementaties

Voorkom onnodige afhankelijkheden met implementaties. Declareer velden bijvoorbeeld als List in plaats van ArrayList en gebruik javax.inject.Inject in plaats van org.springframework.beans.factory.annotation.Autowired.

6.2 Gebruik resources niet langer dan nodig

TODO

6.3 Transactie management

De scope van een transactie is gedefinieerd op de services. TODO Atomikos / etc.

6.4 Hibernate

TODO

6.5 Implementatie keuzes

6.5.1 Spring

Als basis framework voor het opzetten van de componenten wordt Spring gebruikt.

6.5.1.1 Spring IoC (dependency injection)

Implementatie van services en repositories wordt m.b.v. Spring IoC geïnjecteerd.

6.5.1.2 Spring AOP voor transaction management

Transaction management is geregeld m.b.v. Spring; methodes die in een transactie moeten worden uitgevoerd zijn geannoteerd met de @Transactional annotatie van Spring.

6.5.1.3 Spring core (jdbc template)

Voor de interactie met de database gebruikt deze sub-module het JDBC template van Spring.

6.5.1.4 Spring JMS (jms)

Voor het versturen van berichten naar de berichtenmakelaar wordt gebruik gemaakt van een JMS template.

6.5.2 Configuratie

De module is op de volgende manieren te configureren: TODO