# Uitgangspunten opstellen logische testen

## Eén test bevat één doel

### Beschrijving en rationale

Een logisch testgeval is in principe atomair: het test precies één stukje functionaliteit. Dat is iets anders dan één pagina, waarop meerdere functionaliteiten staan. Zelfs een atomaire actie, zoals het uploaden van een databestand, kan tot meerdere logische testen leiden omdat achter de schermen meerdere acties worden uitgevoerd (verschillende validaties bijvoorbeeld), die elk apart getest dienen te worden.

Door een test precies één stukje functionaliteit te laten testen, voorkom je dat een bepaald testgeval niet uitgevoerd wordt doordat een andere test “eerder in het proces” faalt.

## Testbestanden zijn zo minimaal mogelijk

### Beschrijving en rationale

Het testbestand dient het minimaal aantal triples te bevatten die nodig zijn om de test succesvol uit te voeren. Dit voorkomt dat een test niet slaagt vanwege een andere reden dan waar de test voor bedoeld is.

### Voorbeeld

Een testbestand dat bedoeld is om de controleren of de validatie werkt dat elk concept een rdfs:label moet bevatten, bevat enkelde (overige) verplichte triples en niet meer:

concept:GeneriekBeest a skos:Concept;  
 skos:prefLabel "GeneriekBeest";  
.

## Logische testen zijn onafhankelijk van elkaar

### Beschrijving en rationale

Elke test moet afzonderlijk afgetrapt kunnen worden. Om die reden mag het slagen van een logische test niet afhankelijk zijn van het eerst uitvoeren van een andere logische test.

### Voorbeeld

Een randvoorwaarde als “Testresultaten van stap 4 en 5 zijn nog aanwezig” is niet toegestaan.

## Bij de start van elke test is er geen testdata aanwezig in de testomgeving

### Beschrijving en rationale

De eerste stap van een regressietest is altijd het leegmaken van de (data van de) testomgeving. Dit voorkomt dat een test niet slaagt vanwege mogelijk conflicterende data die al in de omgeving stond.

## Testbestanden in verschillende formaten bevatten exact dezelfde data

### Beschrijving en rationale

Ongeacht het formaat waarin de testdata wordt aangeleverd (XML, TTL, JSON-LD etc.), de data zelf dient exact hetzelfde te zijn. Dit zorgt ervoor dat de datacontroles hergebruikt kunnen worden.

### Voorbeeld

Gegeven het onderstaande testbestand in Turtle:

concept:GeneriekBeest a skos:Concept;  
 skos:prefLabel "GeneriekBeest";  
.

Ziet het overeenkomstige testbestand in JSON-LD er als volgt uit:

{"id":"concept:GeneriekBeest"  
,"rdf:type": "skos:Concept"  
,"skos:prefLabel": "GeneriekBeest"  
}

## Logische testen krijgen een betekenisvol onderwerp en betekenisvolle naam

### Beschrijving en rationale

Het onderwerp wordt bepaald door de overeenkomstige rij in de Matrix Functionaliteit. De naam dient zo kort mogelijk, maar wel beschijvend te zijn.

### Voorbeeld

Voorbeelden van onderwerpen zijn:

* Concepten POST XML
* Conceptenpagina
* Zoekpagina

Voorbeelden van testnamen zijn:

* Uploadcontrole
* Validatie label
* Toeleidingsbegrip

## Databestanden moeten kant en klaar beschikbaar zijn

### Beschrijving en rationale

De databestanden die nodig zijn voor een test, moeten vooraf kant en klaar zijn neergezet, zonder dat daar tijdens de test nog aanpassingen aan gedaan moeten worden. Dit omdat het uitprogrammeren van aanpassingen aan een testbestand technisch lastig is en slechter beheerbaar is dan het creëren van een kant en klaar databestand.

### Voorbeeld

Een “when”-clausule als “Testbestand wordt aangepast rdfs:label veranderd van ‘omgevingswet’ naar ‘dataset omgevingswet’” is niet toegestaan. In plaats daarvan moet er een extra databestand gemaakt worden waarin die aanpassing al gedaan is.

## Overig

* Testbestanden hoeven enkel voor de testomgeving te worden opgeleverd
* De naam van een testbestand komt zoveel mogelijk overeen met het onderwerp van de logische test waar het bijhoort