DLogger

Manual

|  |  |
| --- | --- |
| **Property** | **Value** |
| Document type | RP |
| Document config | Armatiek |
| Document owner | Armatiek |
| Editor | Name: Arjan Loeffen, Armatiek BV  URL: <http://www.armatiek.nl/> |
| Editor | Name: Maarten kroon, Armatiek BV  URL: <http://www.armatiek.nl/> |
| Draft URI | URL: <http://some.github.repo> |
| Short name | DLogger manual |
| Spec status | GN-WV |
| Version | 1.1.1 |
| Date | 5 dec 2020 |
| Abbreviations | KLM: Koninklijke Luchtvaart Maatschappij  MIM: Metamodel voor Informatiemodellen |
| Section links | True |
| TOC levels | 2 |
| License | cc-by-nd |
|  |  |

# Samenvatting

Id: abstract

DLogger is een XSLweb applicatie waarin ontwikkelaars tussentijdse resultaten van een transformatie kunnen bekijken. Het betreft “logging” op een web applicatie binnen de context van dezelfde XSLweb installatie. Logs van meerdere webapps kunnen dus uitgelezen worden in dezelfde DLogger viewer.

# Over dit document

Id: sotd

Dit document is in ontwikkeling.

# DLogger

DLogger is een xslweb app die de ontwikkelaar een intuïtieve toegang geeft tot tussenresultaten in een run. Bijvoorbeeld, het request document, de atomaire waarde van een variabele, een tussentijds opgebouwd XML fragment, etc.

Deze waarden worden door DLogger getoond in de volgorde waarin deze door de ontwikkelaar zijn gelogged. Dat kan binnen één run (één aanroep van de web applicatie), of over een reeks van runs heen.

De DLogger oplossing bestaat uit een viewer (verplicht), een XSL stylesheet voor de client applicaties (verplicht), een deployment stylesheet (optioneel), en een demo client applicatie (optioneel).

## Dlogging in de client applicatie

In een XSLweb applicatie neem je twee soorten functie calls op naar DLogger: dlogger:init() en dlogger:save(). Beide calls geven altijd een lege sequence terug.

dlogger:init($clear as xs:boolean) as empty-sequence()

Dus bijvoorbeeld:

<xsl:sequence select="*dlogger:init*(true())"/>

Deze call verwijdert alle informatie van een eerdere run uit het geheugen. Deze call plaats je dus typisch in de request-dispatcher, dus wordt aangeroepen voordat de werkelijke verwerking van de request plaatsvindt.

<xsl:sequence select="*dlogger:init*(false())"/>

Deze call laat alle informatie van een eerdere run in het geheugen staan, maar signalleert wel een nieuw proces. Deze call plaats je dus typisch op een plek waar je verwacht dat de viewer opnieuw het geheugen uitleest en toont.

Vanaf dat moment wordt sequentieel iedere aanroep van DLogger toegevoegd aan de lijst:

<xsl:sequence select="*dlogger:save*('string','Hello!')"/>

Hier wordt de string ‘Hello!’ bewaard onder de naam ‘string’. Later kan deze waarde onder die naam weer worden opgevraagd.

Andere mogelijke waarden zijn onder andere:

<xsl:sequence select="*dlogger:save*('string','Hello!')"/>

<xsl:sequence select="*dlogger:save*('hello variable',**$hello**)"/>

<xsl:sequence select="*dlogger:save*('hello variable (native)',**$hello**,'native.xml')"/>

<xsl:sequence select="*dlogger:save*('hello variable (content)',**$hello**/*node*())"/>

<xsl:sequence select="*dlogger:save*('hello variable (attribute)',**$hello**/***@galaxy***)"/>

De hier getoonde waarden worden als 1 t/m 5 (in die volgorde) bewaard en later, in de viewer, getoond.

Het formaat van de DLogger call is:

Dlogger:save($label as xs:string, $items as item()\*, $type as xs:string) as empty-sequence()

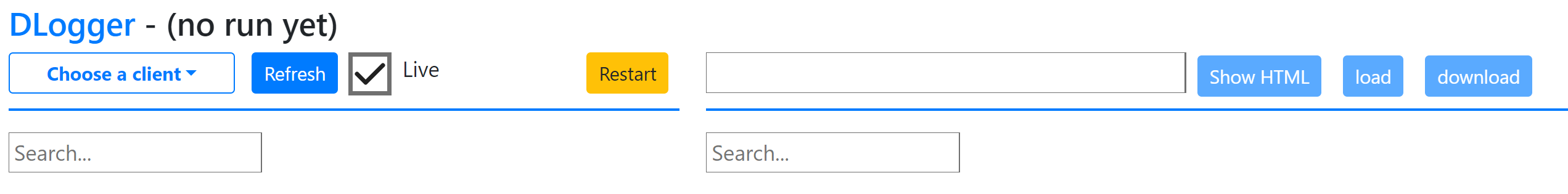
Hierbij geldt het volgende

* $label mag alles zijn, handigste is het een unieke naam te geven waarmee de log positie snel kan worden opgespoord in de XSL code.
* $items mag alles zijn, waarbij geldt dat
  + Als één item wordt weergegeven wordt het getoond zoals verwacht.
  + Als een attribuut of meerdere items worden meegegeven, dat worden deze in een <dlogger-wrapper> opgenomen; deze wordt altijd met inspringing weergegeven zodat de xml struktuur eenvoudiger te volgen is.
* Voor $type geldt het volgende.
  + $type is optioneel.
  + Dlogger is in staat zelf te bepalen dat het XML betreft. Het type wordt dan “xml” tenzij ander opgegeven.
  + Als voor $type de waarde “xml” wordt meegegeven wordt de XML met inspringing getoond.
  + Als “native.xml” wordt meegegeven wordt (als het XML betreft) de XML native getoond, dus zonder formattering. In dat geval wordt de XML via [**Download**] opgeslagen als [$label].native.xml.
  + Als voor $type een andere waarde is opgegeven wordt aangenomen dat dat een valide typering zou zijn van het $item. Typisch voorbeeld is “json” of “html”. Herkend worden: xml, html, xhtml, json, txt.
  + Als DLogger de content niet herkent als xml en er is geen type opgegeven, dat wordt “txt” aangenomen.

## DLogger viewer

De DLogger viewer kan worden opgestart in dezelfde XSLweb context (lees: installatie) als die welke de DLogger -calls doet. Het kunnen dus meerdere XSLweb applicaties zijn.

Bij opstarten wordt een “leeg” scherm getoond.



Door de naam van een XSLweb applicatie te kiezen (bijv. “DLogger-client”) wordt die applicatie “gemonitord”. De client applicatie wordt pas aangeboden als deze applicatie een dlogger:init() heeft uitgevoerd.

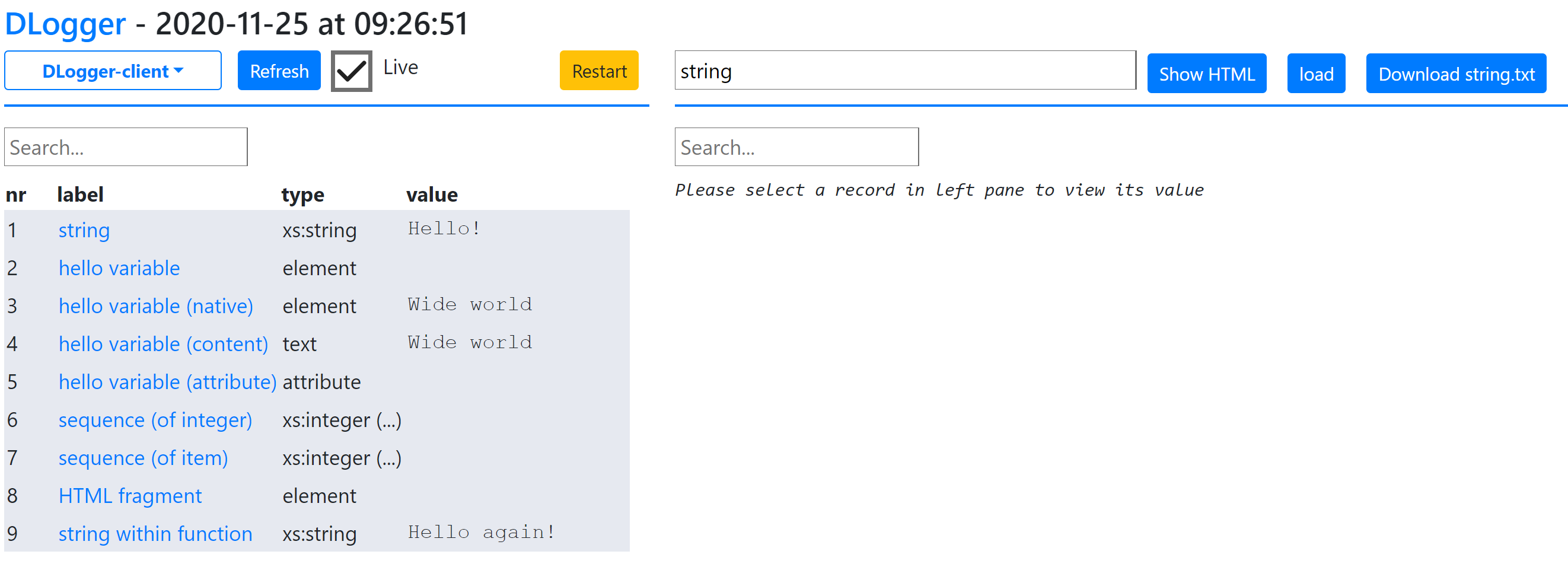
Informatie voor die client komt pas beschikbaar als

1. De applicatie een log schrijft (via dlogger:save())
2. De naam van de applicatie is ingevoerd in de DLogger viewer, vóór de knop [**Refesh**].
3. De [**Refresh**] knop is ingedrukt, of (elke 3 seconden) wanneer [**Live**] is aangevinkt.

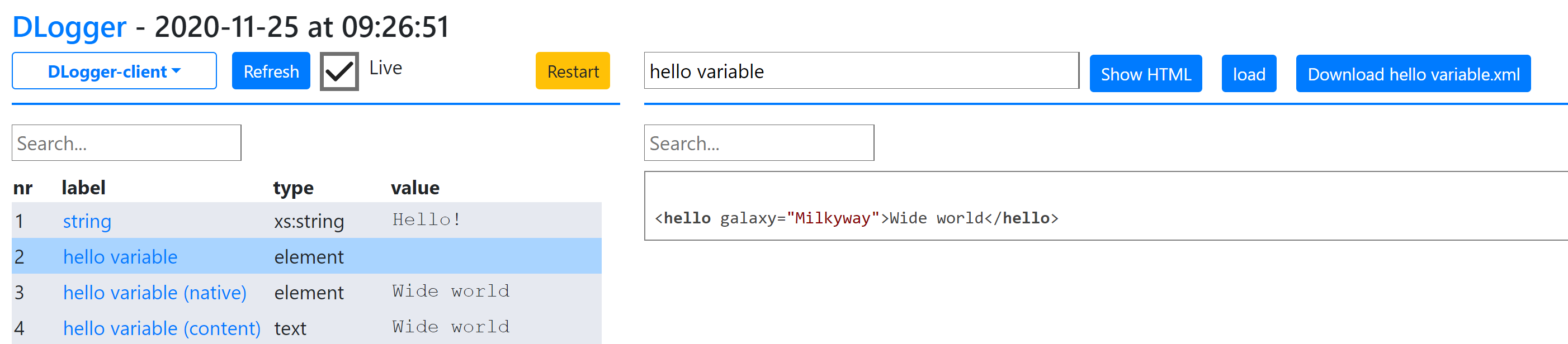
Op dat moment wordt de client applicatie met de naam DLogger-client gemonitord. In eerste instantie gebeurt niets, óf de informatie van de laatste run van de client applicatie wordt getoond.

Start nu de client applicatie (in een ander venster, of in een andere browser). De tabel wordt automatisch gevuld met de logs die zijn uitgestuurd door aanroepen in de client applicatie.

De weergave van deze functie aanroepen is:



Door op de tweede hello variable te klikken laadt de waarde in het rechter scherm. Dit is een weergave van de inhoud van de waarde, in dit geval een enkel XML element.

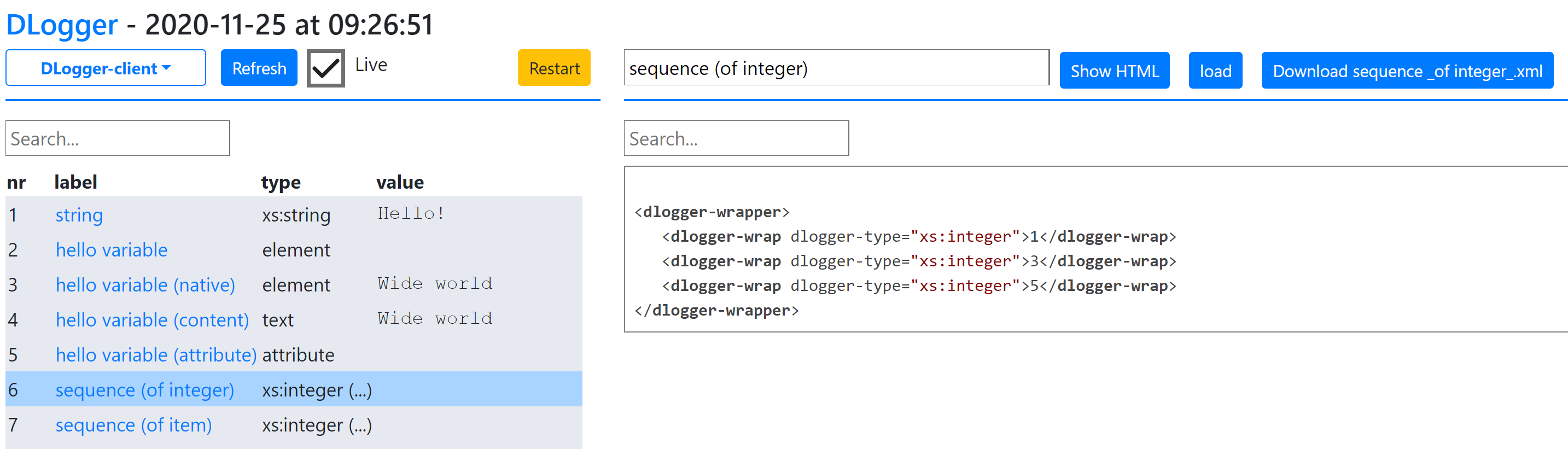
****

De weergave van sequences van méér dan een item wordt duidelijk gemaakt middels ellipsis:

<xsl:sequence select="*dlogger:save*('sequence A',(1, 3, 5))"/>

<xsl:sequence select="*dlogger:save*('sequence B',(1, **$hello**))"/>

Levert twee ingangen met een ellipsis. Het type is het type van het éérste item in de sequence. Om de waarden te zien klik je op het linkje:



Er wordt hier gebruik gemaakt van een “XML wrapper” om de inhoud expliciet te maken.

Kort even de zichtbare velden op de pagina.

* Middels [**Refresh**] haal je de laatste info op, zoals beschreven.
* Vink [**Live**] aan om automatisch (elke 3 seconden) de laatste situatie op te halen.
* Met [**Restart**] maak je alle geheugen vrij en kun je nieuwe DLogs lezen. Dit is vooral van belang als je niet initialiseert met de “clean” optie (en de logs dus langzaam aangroeien).
* Middels [**Show HTML**] kan HTML content getoond worden als HTML. Natuurlijk kan alleen een complete HTML pagina (met CSS en JS daarbinnen) alleen een werkelijk sluitende weergave van de HTML opleveren.
* Middels [**Load**] kun je de inhoud in een tweede browser venster bekijken, wat soms handig is bij grotere bestanden, en als de browser de document-boom interactief laat benaderen.
* Middels de knop [**Download**] kun je de waarde als bestand opslaan.
* Het veld [**Search**] links in het scherm: Je kunt snel zoeken in een mogelijk lange lijst van gelogde waarden. Meer karakters beperkt de lijst, verwijderen breidt deze weer uit.
* Het veld [**Search**] rechts in het scherm: highlight de ingetikte tekst in het venster.

Omdat meerdere dloggers op je systeem kunnen draaien kan het verwarrend zijn: naar welke dlogger zit je te kijken? Daarom kun je een parameter meegeven aan de dlogger: ?visual= green | blue | orange | yellow | red. Op basis daarvan wordt een gekleurde achtergrond gegenereerd. Als je het niet opgeeft is de achtergrond wit.

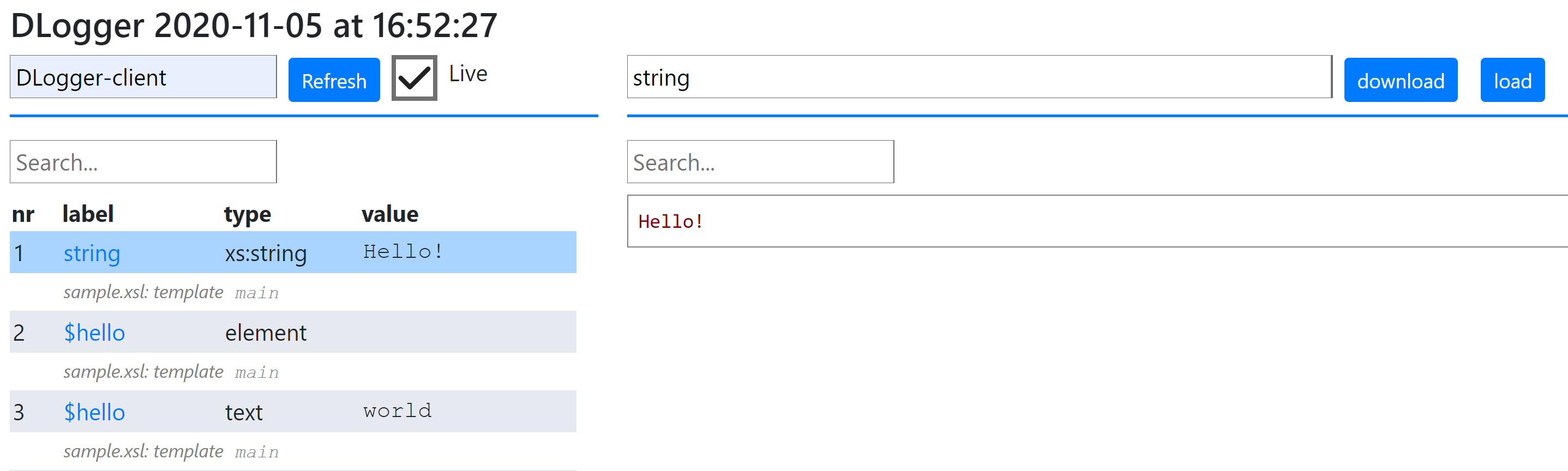
## Reflexion

Omdat het soms handig is te weten wáár precies de betreffende informatie was gelogged, en omdat het niet handig is om dat steeds zelf expliciet mee te geven bij iedere dlogger:save() aanroep, is het mogelijk de stylesheets te deployen via een reflexion stylesheet. Deze transformatie kun je het beste aanroepen op iedere (nieuwe, aangepaste) stylesheet, dus tijdens deployment naar een ontwikkelomgeving (dus typisch vanuit je GIT ontwikkelomgeving naar je lokale xslweb omgeving).

Deze stylesheet past de dlogger:save($name,$item) call aan, en plaatst drie parameters:

dlogger:save-reflexion($stylesheet-pad,$container-type,$container-name,$name,$item)

Hierdoor wordt de context uitgebreid met deze informatie, en de weergave wordt in dat geval:



De functie dlogger:save-reflexion() is onderdeel van de standaard dlogger-common.xsl stylesheet.

Let op, de reflection stylesheet laat de indentatie van de oorspronkelijke stylesheet niet intact. Hierdoor is het niet altijd mogelijk een foutmelding van Saxon (met regelnummering) goed te interpreteren. De keuze om reflexion in te zetten is dus afhankelijk van je prioriteiten: behoefte precies na te gaan waar een Dlog is geschreven, of behoefte altijd een exacte foutlocatie van Saxon door te krijgen.

# Hoe zelf inzetten?

Om DLogger te gebruiken moet de ontwikkelaar het volgende doen.

1. De webapp installeren (volgens de standaard methode, door deze in een XSLweb webapps folder een plek te geven).
2. Een enkele XSLT van DLogger includeren in de eigen app (namelijk dlogger.xsl).
3. DLogger initialiseren door de dlogger:init() functie aan te roepen. Dit gebeurt typisch in de request-dispatcher, omdat deze altijd als eerste wordt geëvalueerd.

DLogger is actief wanneer expliciet is aangegeven dat deze modus actief moet zijn: $config:dlogger-mode = true(). Terwijl het echt als een ontwikkelaars-tool wordt gepositioneerd, en dus typisch actief is in development-mode, is het soms handig de deze logs ook beschikbaar te hebben op een OTAP server. Soms wil je toch kunnen kijken naar sommige tussenresultaten. Daarom hebben we een aparte setting hiervoor geintroduceerd.

Functie calls naar dlogger:\*() functies leveren de empty sequence op; in productie “doen de functies niets”.

Wanneer de DLogger app is geïnstalleerd kan de DLogger viewer in iedere internet browser worden benaderd via:

<http://localhost:8080/xslweb/DLogger>-viewer

Om te zien hoe e.e.a. werkt is een voorbeeld DLogger-client beschikbaar. Dit is een client app, die gebruik maakt van DLogger-common. Zie de screenshots hierboven.

# DLogger aanspreken buiten XSLweb om

Het is mogelijk DLogger aan te spreken vanuit een XSLT die in een applicatie die zelf niet onder XSLweb draait wordt aangesproken; we noemen dit hieronder de *non-XSLweb applicatie*. Een voorbeeld is een Saxon XSLT binnen een zelfstandige Java applicatie. Omdat de DLogger viewer de eigen XSLweb context attributen uitleest om de informatie op het scherm te toveren is het noodzakelijk dat er een “proxy” web applicatie bestaat die logs ontvangt (via REST) en deze “alsof het eigen logs waren” doorgeeft aan de viewer.

Om dit spelletje te kunnen spelen moet de non-XSLweb applicatie in staat zijn vanuit de XSLT’s HTTP-requests uit te voeren. Deze XSLT’s moeten twee functies implementeren:

dlogger-impl:put($atts as element(atts)) as empty-sequence()

Het element atts wordt door het framework opgebouwd, en met door de client naar de DLogger-proxy worden gestuurd (via POST).

dlogger-impl:get($key as xs:string) as xs:string?

De key wordt door het DLogger framework gespecificeerd, en de waarde moet door deze functie worden teruggegeven.

Iedere non-XSLweb applicatie moet dus haar eigen implementatie realiseren van deze twee XSL functies. Daarbij moet de proxy webapp worden aangesproken. De proxy is beschikbaar als:

/xslweb/DLogger-proxy

De proxy implementeer de volgende URL’s:

/xslweb/DLogger-proxy?key=\* (HTTP GET)

/xslweb/DLogger-proxy (HTTP POST)

Deze twee REST interfaces vormen een non-XSLweb equivalent van de context:set-attribute() en context-get-attribute() functies. De POST is een POST omdat dan meerdere attributen in één klap kunnen worden doorgegeven, en er geen serieuze grens is aan de omvang van de waarden van de attributen.

## XSLweb voorbeeld van een non-XSLweb applicatie

Er is een voorbeeld beschikbaar van een XSLweb applicatie die *in een andere context* draait dan die van de DLogger-viewer. Dit is (als illustratie) een voorbeeld van een non-XSLweb applicatie. Deze is te vinden in:

/xslweb/DLogger-proxy-client

Deze applicatie doet hetzelfde als de DLogger-client applicatie, maar geheel over HTTP.

# Technische achtergrond

Dlogger schrijft en leest naar de XSLweb context via context attributen. Iedere run van de client webapp verwijdert alle eerder opgeslagen waarden uit het geheugen.

De context attributen hebben de naam

{client-webapp-naam}\_{log-nr}\_{log-type} = {waarde}

De client webapp naam is de naam van de client applicatie (dus zoals opgenomen in $config:webapp-path), bijv. “cvdr”. Log-nr is een oplopend nummer, toegekend door DLogger. {log-type} is een aspect waarvan informatie wordt opgeslagen, zie hieronder. De waarde is, in geval van de registratie van een opgegeven item, de serialisatie van dat item.

Bijvoorbeeld:

<xsl:sequence select="dlogger:save('tekst: start','Hallo!')"/>

Levert de volgende context attributen:

|  |  |
| --- | --- |
| **Context attribuut** | **Waarde** |
| Mijnapp\_3 = ‘Hallo!’ | Zonodig geserialiseerde waarde. |
| Mijnapp\_3\_label = ‘tekst: start’ | Label zoals opgegeven en integraal getoond in browser. |
| Mijnapp\_3\_file = ‘tekst\_start’ | File naam, geschikt voor download. |
| Mijnapp\_3\_type = ‘txt’ | Extensie voor download, zoals xml en txt. |
| Mijnapp\_3\_datatype = ‘xs:string’ | Xpath 3.0 datatype, wanneer meerdere het datatype van het eerste item, voorzien van (…) |
| Mijnapp\_3\_value = ‘Hallo!’ | Toonbare waarde voor snelle weergave. |

Deze worden door de DLogger viewer uitgelezen en aangeboden.