Client-Bibliothek

Versionen

Datum	Notiz	Version
30.06.2021	Erste Veröffentlichung	1.0

Autoren

Name	Institution	Email
Fabian Bruckner	Fraunhofer ISST	fabian.bruckner@isst.fraunhofer.de

D° verwendet intern als Typsystem Nukleus. Aus diesem Grund erfolgen Ein- und Ausgaben an und von D°-Applikationen im Format von Nukleus. Da aber die Verwendung von Nukleus nicht vom Aufrufer vorausgesetzt werden soll, werden Nukleus Instanzen als JSON-String kodiert und in einem weiteren JSON-Objekt gekapselt um mit D°-Applikationen zu interagieren. Diese Struktur wird im Kontext von D° Scope genannt.

Das nachfolgende Beispiel zeigt einen mögliche Ein- bzw. Ausgabe Scope für eine D°-Applikation. Man kann sehen, dass die Werte der Attribute `payload` und `anotherParameter` String Repräsenatationen von JSON-Objekten sind.

Beispiel Ein-/Ausgabe einer D°-Applikation

```
{
  "payload": "{\"Text\":\"\"}",
  "creationDate": "{\"time.DateTime\":{\"day\":{\"time.Day\":\"1\"},\"month\":{\"time.Month\":\"1\"},\"year\":
{\"UnsignedInt\":\"1980\"},\"hour\":{\"time.Hour\":\"0\"},\"minute\":{\"time.Minute\":\"0\"},\"second\":{\"time.
Second\":\"0\"},\"utcOffsetPositive\":{\"Boolean\":\"true\"},\"utcOffsetHours\":{\"time.Hour\":\"0\"},\"
utcOffsetMinutes\":{\"time.Minute\":\"0\"}}}"
}
```

An dem Beispiel zeigt sich, dass die Erstellung von validen Eingaben bzw. die Verarbeitung von Ausgaben von D°-Applikationen eine gewisse Komplexität besitzt. Dies ist an den Systemgrenzen, an denen D°-Applikationen von Anwendern oder anderen Applikationen verwendet werden, problematisch. In einem Prozess, in denen die Ausgabe einer D°-Applikation als Eingabe für die nächste Verwendet wird, tritt diese Problematik nicht auf, da jede D°-Applikation entsprechende Scopes verstehen und erzeugen kann.

Diese Problematik ist die ursprüngliche Motivation für die Entwicklung einer D° Clientbibliothek. Darüber hinaus werden weitere Funktionen in die Clientbibliothek integriert, welche die Verwendung von bzw. Interaktion mit D°-Applikationen vereinfachen. Dabei kann die Clientbibliothek in Kombination mit jeder D°-Applikation verwendet werden und erweitert somit die Aufrufmöglichkeiten/Schnittstellen, welche die App zur Verfügung stellt. Ebenso ist es möglich die Clientbibliothek als Abhängigkeit in andere Software zu integrieren und direkt auf die enthaltenen Funktionalitäten zuzugreifen.

Umwandlung von Ein- und Ausgaben

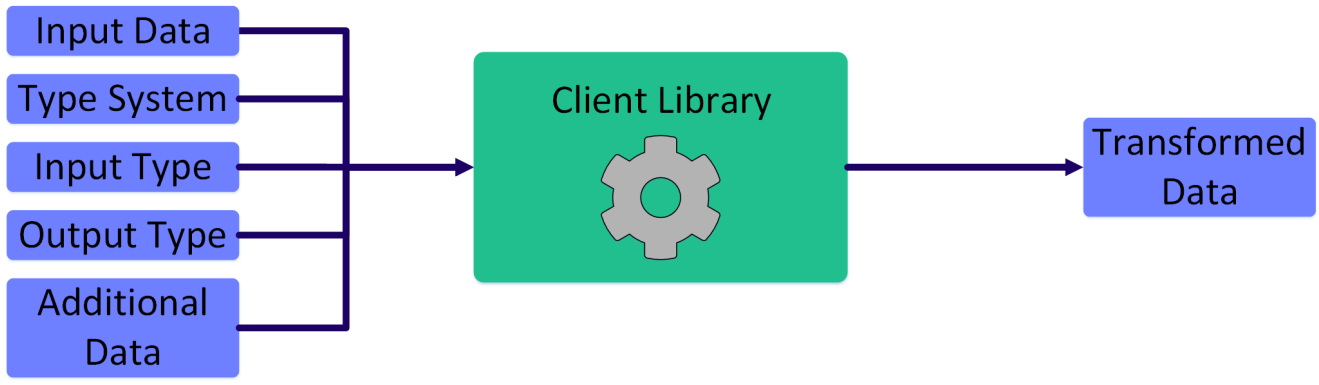
Die entwickelte Funktionalität der Clientbibliothek unterstützt unterschiedliche Arten der Transformation. Es können D°-Scopes, Nukleus-Objekte und normale JSON-Objekte als Eingaben und Zielformat verwendet werden. Dabei ist eine Transformation von jedem Eingabeformat zu jedem Zielformat möglich. Unter Umständen ist es notwendig neben den Eingabedaten zusätzliche Daten bereitzustellen, da diese in den Eingabedaten nicht vorhanden sind, aber für die Transformation notwendig sind. Beispielsweise ist es notwendig, bei der Transformation von einem JSON-Objekt zu einem D°-Scope, anzugeben, welche Nukleus-Typen verwendet werden sollen und wie das transformierte Nukleus-Objekt im D°-Scope benannt ist.

Nachfolgend die in der D°-Clientbibliothek enthaltene Nutzungshilfe für Formattransformationen zu sehen. Im Anschluss werden einige Beispielaufrufe aufgezeigt.

Nutzungshilfe für den Transformationsmodus

```
Usage: declib transform [-hpV] -d=DATA [-i=IN_FORMAT] [-j=IDENTIFIER] [-k=KEY]
                        [-n=NUKLEUS_TYPE] [-o=OUT_FORMAT] [-t=FILE...]...
Transform JSON nukleus instances and input/output-scopes for/from D°
-applications to plain JSON and vice-versa.
  -d, --data=DATA            The data to transform.
  -h, --help                 Show this help message and exit.
  -i, --input=IN_FORMAT      Format of the data that will be transformed.
                             Allowed values: DEGREE, NUKLEUS, JSON
                             Default: DEGREE
  -j, --jsonKey=IDENTIFIER   The key that is used to extract primitive values
                             during all transformations with input format
                             JSON.
  -k, --key=KEY              The key that is added to the input data if the
                             input is nukleus and one of the following
                             applies:
                             - output format is D°
                             - nukleus instance is a primitive type
                             Usage for transforming composite instance to JSON
                             is optional.
                             A key is also mandatory for JSON -> D°
                             transformations
  -n, --nukleus=NUKLEUS_TYPE For transformations from plain JSON to other
                             formats it is mandatory to provide the
                             targetnukleus type identifier. This parameter is
                             mandatory for IN_FORMAT JSON and not used for
                             all other cases.
  -o, --output=OUT_FORMAT    Target format for transformation.
                             Allowed values: DEGREE, NUKLEUS, JSON
                             Default: JSON
  -p, --pretty               Target format for transformation.
                             Default: false
  -t, --types=FILE...        Nukleus type system files which will be used.
  -V, --version              Print version information and exit.
```

Die nachfolgende Abbildung zeigt eine schematische Darstellung der Ein- und Ausgaben, welche von die D°-Clientbibliothek während der Transformation verwendet, bzw. erzeugt werden.



Beispiel 1: Transformation von D°-Scope zu Nukleus-Objekt

Beispiel für eine Formattransformation

```
java -jar declib.jar transform -d="{\"date\":\"{\\\"time.DateTime\\\":{\\\"day\\\":{\\\"time.Day\\\":\\\"1\\\"},
\\\"month\\\":{\\\"time.Month\\\":\\\"1\\\"},\\\"year\\\":{\\\"UnsignedInt\\\":\\\"1980\\\"},\\\"hour\\\":{\\\"
time.Hour\\\":\\\"0\\\"},\\\"minute\\\":{\\\"time.Minute\\\":\\\"0\\\"},\\\"second\\\":{\\\"time.Second\\\":\\\"
0\\\"},\\\"utcOffsetPositive\\\":{\\\"Boolean\\\":\\\"true\\\"},\\\"utcOffsetHours\\\":{\\\"time.Hour\\\":\\\"
0\\\"},\\\"utcOffsetMinutes\\\":{\\\"time.Minute\\\":\\\"0\\\"}}}\"}" -t="typeDefinitions/primitive.types.yaml"
-t="typeDefinitions/composite.types.yaml" -o=NUKLEUS -p

> {
  "date": {
    "time.DateTime": {
      "day": {
        "time.Day": "1"
      },
      "month": {
        "time.Month": "1"
      },
      "year": {
        "UnsignedInt": "1980"
      },
      "hour": {
        "time.Hour": "0"
      },
      "minute": {
        "time.Minute": "0"
      },
      "second": {
        "time.Second": "0"
      },
      "utcOffsetPositive": {
        "Boolean": "true"
      },
      "utcOffsetHours": {
        "time.Hour": "0"
      },
      "utcOffsetMinutes": {
        "time.Minute": "0"
      }
    }
  }
}
```

Beispiel 2: Transformation von D°-Scope zu JSON-Objekt

Beispiel für eine Formattransformation

```
java -jar declib.jar transform -d="{\"date\":\"{\\\"time.DateTime\\\":{\\\"day\\\":{\\\"time.Day\\\":\\\"1\\\"},
\\\"month\\\":{\\\"time.Month\\\":\\\"1\\\"},\\\"year\\\":{\\\"UnsignedInt\\\":\\\"1980\\\"},\\\"hour\\\":{\\\"
time.Hour\\\":\\\"0\\\"},\\\"minute\\\":{\\\"time.Minute\\\":\\\"0\\\"},\\\"second\\\":{\\\"time.Second\\\":\\\"
0\\\"},\\\"utcOffsetPositive\\\":{\\\"Boolean\\\":\\\"true\\\"},\\\"utcOffsetHours\\\":{\\\"time.Hour\\\":\\\"
0\\\"},\\\"utcOffsetMinutes\\\":{\\\"time.Minute\\\":\\\"0\\\"}}}\"}" -t="typeDefinitions/primitive.types.yaml"
-t="typeDefinitions/composite.types.yaml"

> {"date":{"day":"1","month":"1","year":"1980","hour":"0","minute":"0","second":"0","utcOffsetPositive":"true","
utcOffsetHours":"0","utcOffsetMinutes":"0"}}
```

Beispiel 3: Transformation von Nukleus-Objekt zu D°-Scope

Beispiel für eine Formattransformation

```
java -jar declib.jar transform -d="{\"time.DateTime\":{\"day\":{\"time.Day\":\"1\"},\"month\":{\"time.Month\":\"
1\"},\"year\":{\"UnsignedInt\":\"1980\"},\"hour\":{\"time.Hour\":\"0\"},\"minute\":{\"time.Minute\":\"0\"},\"
second\":{\"time.Second\":\"0\"},\"utcOffsetPositive\":{\"Boolean\":\"true\"},\"utcOffsetHours\":{\"time.Hour\":
\"0\"},\"utcOffsetMinutes\":{\"time.Minute\":\"0\"}}}" -t="typeDefinitions/primitive.types.yaml" -t="composite.
types.yaml" -i=NUKLEUS -o=DEGREE -k=date


> {"date":"{\"time.DateTime\":{\"day\":{\"time.Day\":\"1\"},\"month\":{\"time.Month\":\"1\"},\"year\":{\"
UnsignedInt\":\"1980\"},\"hour\":{\"time.Hour\":\"0\"},\"minute\":{\"time.Minute\":\"0\"},\"second\":{\"time.
Second\":\"0\"},\"utcOffsetPositive\":{\"Boolean\":\"true\"},\"utcOffsetHours\":{\"time.Hour\":\"0\"},\"
utcOffsetMinutes\":{\"time.Minute\":\"0\"}}}"}
```

Beispiel 4: Transformation von Nukleus-Objekt zu JSON-Objekt

Beispiel für eine Formattransformation

```
java -jar declib.jar transform -d="{\"time.DateTime\":{\"day\":{\"time.Day\":\"1\"},\"month\":{\"time.Month\":\"
1\"},\"year\":{\"UnsignedInt\":\"1980\"},\"hour\":{\"time.Hour\":\"0\"},\"minute\":{\"time.Minute\":\"0\"},\"
second\":{\"time.Second\":\"0\"},\"utcOffsetPositive\":{\"Boolean\":\"true\"},\"utcOffsetHours\":{\"time.Hour\":
\"0\"},\"utcOffsetMinutes\":{\"time.Minute\":\"0\"}}}" -t="typeDefinitions/primitive.types.yaml" -t="
typeDefinitions/composite.types.yaml" -i=NUKLEUS

> {"day":"1","month":"1","year":"1980","hour":"0","minute":"0","second":"0","utcOffsetPositive":"true","
utcOffsetHours":"0","utcOffsetMinutes":"0"}
```

Beispiel 5: Transformation von JSON-Objekt zu Nukleus-Objekt

Beispiel für eine Formattransformation

```
java -jar declib.jar transform -d="{\"day\":\"1\",\"month\":\"1\",\"year\":\"1980\",\"hour\":\"0\",\"minute\":\"
0\",\"second\":\"0\",\"utcOffsetPositive\":\"true\",\"utcOffsetHours\":\"0\",\"utcOffsetMinutes\":\"0\"}" -t="
typeDefinitions/primitive.types.yaml" -t="typeDefinitions/composite.types.yaml" -i=JSON -o=NUKLEUS -n=time.
DateTime

> {"time.DateTime":{"day":{"time.Day":"1"},"month":{"time.Month":"1"},"year":{"UnsignedInt":"1980"},"hour":
{"time.Hour":"0"},"minute":{"time.Minute":"0"},"second":{"time.Second":"0"},"utcOffsetPositive":{"Boolean":"
true"},"utcOffsetHours":{"time.Hour":"0"},"utcOffsetMinutes":{"time.Minute":"0"}}}
```

Beispiel 6: Transformation von JSON-Objekt zu D°-Scope

Beispiel für eine Formattransformation

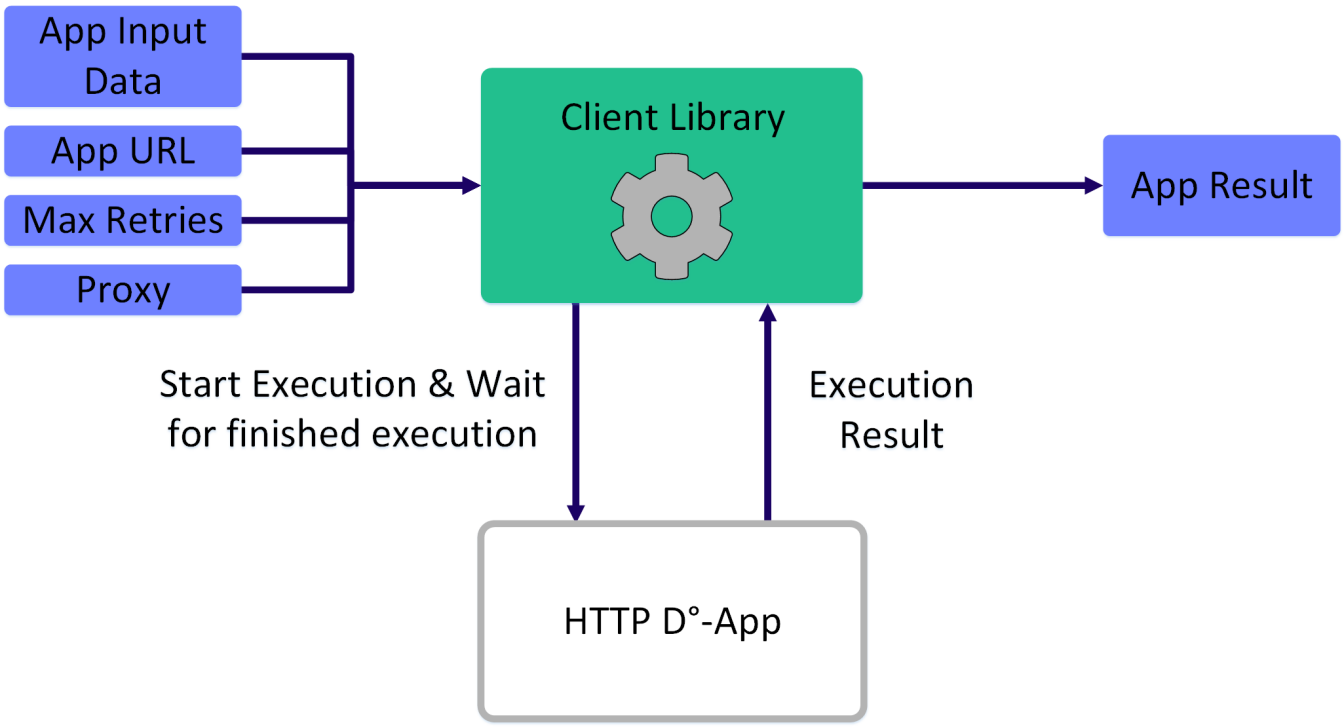
```
java -jar declib.jar transform -d="{\"day\":\"1\",\"month\":\"1\",\"year\":\"1980\",\"hour\":\"0\",\"minute\":\"
0\",\"second\":\"0\",\"utcOffsetPositive\":\"true\",\"utcOffsetHours\":\"0\",\"utcOffsetMinutes\":\"0\"}" -t="
typeDefinitions/primitive.types.yaml" -t="typeDefinitions/composite.types.yaml" -i=JSON -o=DEGREE -n=time.
DateTime -k=date

> {"date":"{\"time.DateTime\":{\"day\":{\"time.Day\":\"1\"},\"month\":{\"time.Month\":\"1\"},\"year\":{\"
UnsignedInt\":\"1980\"},\"hour\":{\"time.Hour\":\"0\"},\"minute\":{\"time.Minute\":\"0\"},\"second\":{\"time.
Second\":\"0\"},\"utcOffsetPositive\":{\"Boolean\":\"true\"},\"utcOffsetHours\":{\"time.Hour\":\"0\"},\"
utcOffsetMinutes\":{\"time.Minute\":\"0\"}}}"}
```

Aufruf von D°-Applikationen

D°-Applikationen, welche nicht über die Kommandozeile aufgerufen werden, können aktuell nur asynchron angesprochen werden. Ein Endpunkt wird verwendet, um die Ausführung zu starten und ein weiterer Endpunkt wird verwendet um die Ergebnisse abzufragen.

Die Clientbibliothek wrappt diese Endpunkte und stellt eine synchrone Aufrufmöglichkeit für D°-Applikationen bereit. Hierdurch wird es möglich D°-Applikationen sowohl synchron als auch asynchron zu verwenden. Die Nachfolgende Abbildung stellt den Ablauf schematisch dar.



Nachfolgend wird die Nutzunfshilfe für diese Funktionalität gezeigt. Diese ist ein Bestandteil der D°-Clientbibliothek.

Nutzungshilfe für den Aufrufmodus

```
Usage: declib call [-hV] -d=DATA [-p=PROXY] [-r=RETRIES] -u=URL
Perform a synchronous call of a D°-application.
  -d, --data=DATA       The data which will be used as input.
  -h, --help            Show this help message and exit.
  -p, --proxy=PROXY     The that will be used
  -r, --retry=RETRIES   Max number of retries, -1 for infinite retries.
                        Default: -1
  -u, --url=URL         The application's endpoint which will be used
  -V, --version         Print version information and exit.
```