Moyo マニュアル

2017 年 1 月 14 日版

改訂履歴

改訂日	改訂者	改訂内容	版
2017 年 1 月 14 日	中川雅三	初版	

1. はじめに

Moyo はリアルタイムに変化する情報を扱えるようにした RDF ストアです。Moyo は、Apache Jena Fuseki 2.4.1（以下 Fuseki と呼びます）を改造したものです。Moyo は、拡張機能を使わなければ Fuseki と互換です。インストールや操作の方法の詳細については Fuseki の資料を参照してください。

Moyo と Fuseki との相違点は以下の 2 点です。

- インメモリの RDF ストアの中に、外部からのデータ送信によってリアルタイムに値が変化するノード、「物理ノード」を構成することができます。
  物理ノードの構成には Moyo が定義する特定の語彙を使います。この語彙を使わなければ Moyo は Fuseki と互換に動作します。

- 「拡張エンドポイント」を持ちます。
  拡張エンドポイントはつぎの機能を持ちます。
  - 簡易な URL 指定による、物理ノードの現在値と値の履歴への HTTP アクセス。
    タグパスと呼ぶ、RDF で表現した木構造のラベルで物理ノードを指定します。物理ノード ID は、物理ノードを定義するときに指定する ID です。これらを指定することで SPARQL を使わずに物理ノードの値を変更したり読み出したりすることができます。
    これらの機能をつかうと、物理ノードの値の履歴を読み出すことができます。
  - WebSocket による物理ノードへのアクセス。
    - WebSocket から、物理ノードの現在値を少ないオーバーヘッドで変更することができます。
    - WebScoket から、物理ノードの値変化通知を受信できます。

Moyo は Window10 と Amazon Linux AMI 2016.09.1 (HVM), SSD Volume Type で動作確認しました。

1.1. ライセンス

Moyo Apache License Version 2.0（「本ライセンス」）に基づいてライセンスされます。あなたがこのファイルを使用するためには、本ライセンスに従わなければなりません。本ライセンスのコピーは下記の場所から入手できます。

http://www.apache.org/licenses/LICENSE-2.0

適用される法律または書面での同意によって命じられない限り、本ライセンスに基づいて頒布されるソフトウェアは、明示黙示を問わず、いかなる保証も条件もなしに「現状のまま」頒布されます。本ライセンスでの権利と制限を規定した文言については、本ライセンスを参照してください。

Copyright 2016, 2017 NAKAGAWA Masami (m-nakagawa)

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.

2. インストール

2.1. 解凍と Moyo サーバの起動

配布パッケージを解凍すると、Fuseki パッケージとサンプルデータのパッケージが入っています。

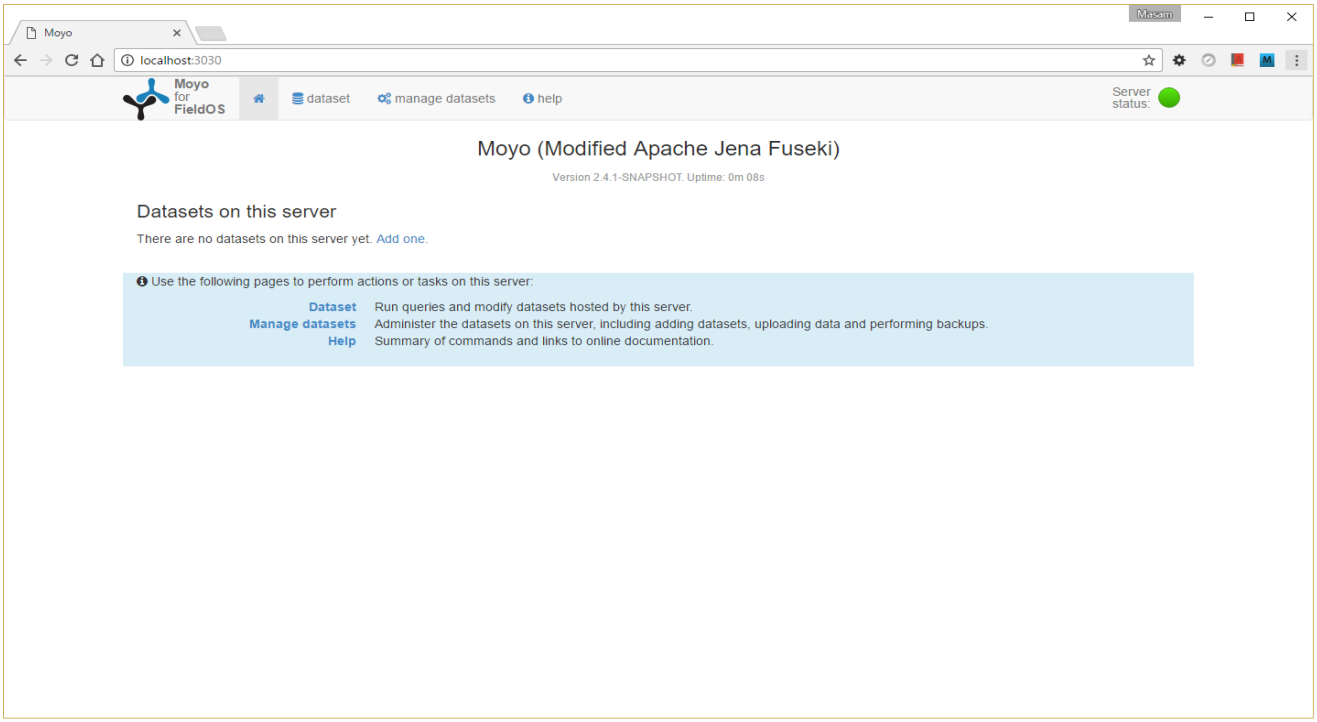
Fuseki パッケージ（apache-jena-fuseki-2.4.1-SNAPSHOT-MoyoYYYYMMDD.zip）は、通常の Fuseki と同じ方法でインストール・実行してください。Windows10 の場合の操作はつぎのとおりです。

- unzip する
- コマンドプロンプトを開き、「cd apache-jena-fuseki-2.4.1-SNAPSHOT」でフォルダを移動する。
- コマンドラインからサーバを起動する
  「fuseki-server.bat」を実行します。

サンプルデータは適当な場所へ解凍してください。このマニュアルでは、「D:/work/」フォルダへ解凍したものとして説明しています。

2.2. Moyo サーバの動作確認

ブラウザから「http://localhost:3030」を開くことで、Fuseki のメインメニューが開きます。



3. サンプルの実行

「moyo-sample.zip」は 2 個のサンプルを含みます。

- 「sample01」: 投票デモ
  複数のブラウザ画面をひらいて操作すると、操作結果をすべての画面へリアルタイムに反映します。

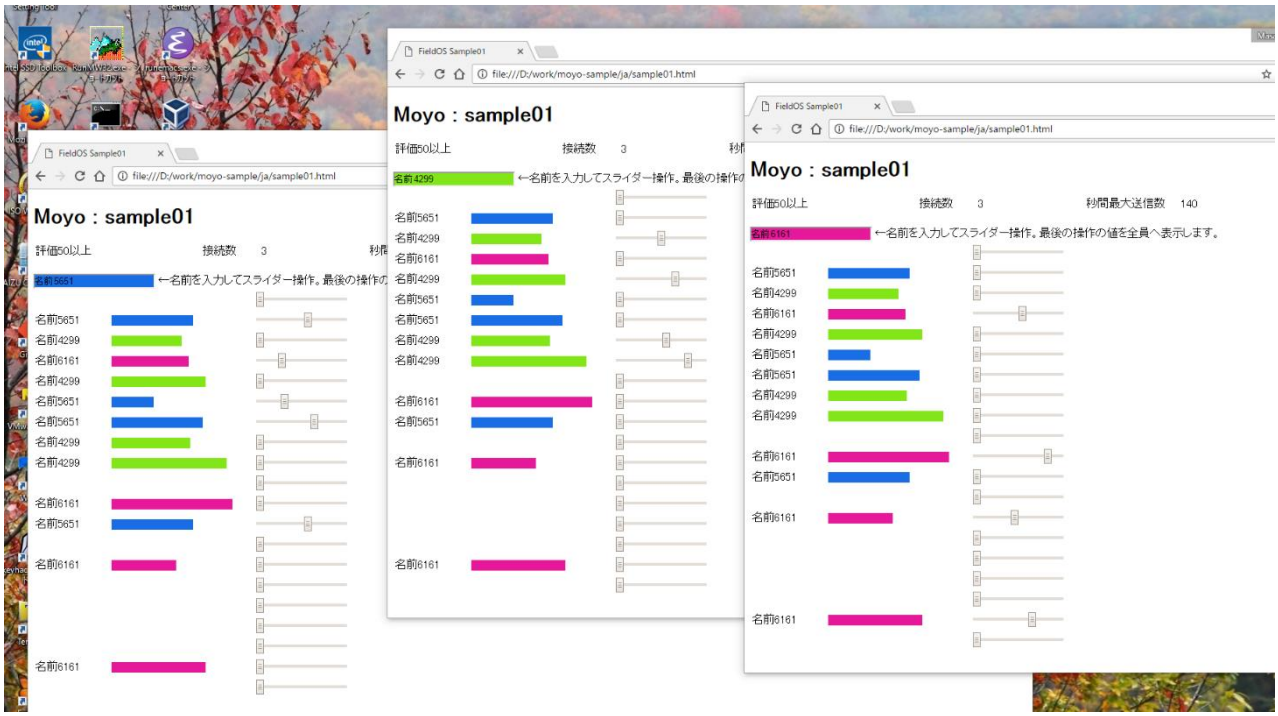
- 「sample02」: Moyo プログラミング説明用サンプル
  次章の説明に使います。

いずれのサンプルもファイルは「D:/work/moyo-sample/ja」ディレクトリにあります。

3.1. 投票デモ

ブラウザで複数の画面を開き、スライダを操作するとすべての画面のスライダが動きます。このサンプルを発展させると、スライダ操作で 0～100 点の値を投票するシステムを作ることができるので投票デモと名付けています。現状では単にスライダの動きを表示するだけのデモプログラムです。

つぎの画面は、3 個のブラウザ画面での実行例です。



それぞれの画面には色と名前が乱数で割り当てられます。スライダを操作すると、他の画面にその色と名前とともに操作した値を反映します。

サンプルはつぎの 3 ファイルで構成しています。

- sample01.ttl
  RDF データです。Fuseki コンソールからアップロードします。
- sample01.html
  HTML ページです。ローカルファイル「file:///...」としてブラウザ表示します。
- sample01.js
  HTML ページから起動する JavaScript のプログラムです。Moyo へ WebSocket 接続して物理ノードの値を読み書きします。（WebSocket には同一オリジンポリシーが適用されないため、HTML ファイルをローカルファイルとして実行しても、JavaScript から Moyo サーバへアクセスすることができます。）

「D:/work/moyo-sample/ja/python」フォルダには、物理ノードの値を読み書きする Python で記述したサンプルプログラムがあります。Python 2.7 の実行環境があれば、このデモも実行することができます。
このサンプルはつぎの手順で実行します。

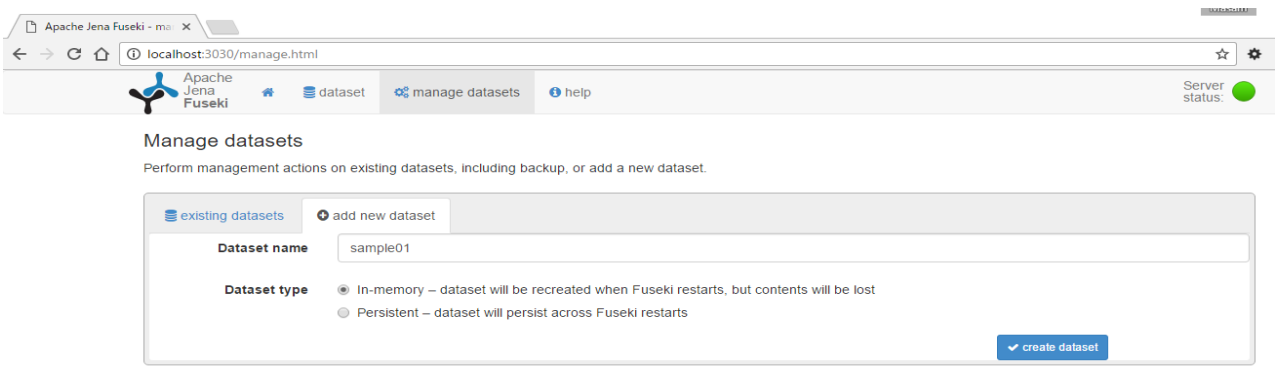
- データセット「sample01」の作成
  データセットは 1 回作成すると Moyo サーバが設定を記録しますので、次回の実行からは作成しなおす必要はありません。

- RDF データ「sample01.ttl」のアップロード
  現在の Moyo はインメモリのデータセットでのみ動作します。Moyo サーバを再起動するたびにこの操作が必要です。Moyo サーバは Fuseki の機能をすべて踏襲していますので、不揮発の RDF ストアとしての動作が必要な場合には、HTTP で接続して再アップロードや更新データのダウンロードを実行するスクリプトを作成することも可能です。

- ブラウザによる表示と操作
  Python スクリプトの実行

3.1.1.データセットの作成

- サーバ画面で「manage datasets」タブを選択し「add new dataset」を選択
- つぎの値を設定
  Dataset name　『sample01』
  Dataset type　『In-memory』 ←　現バージョンは In-memory でのみ動作します。
- create dataset をクリックします。

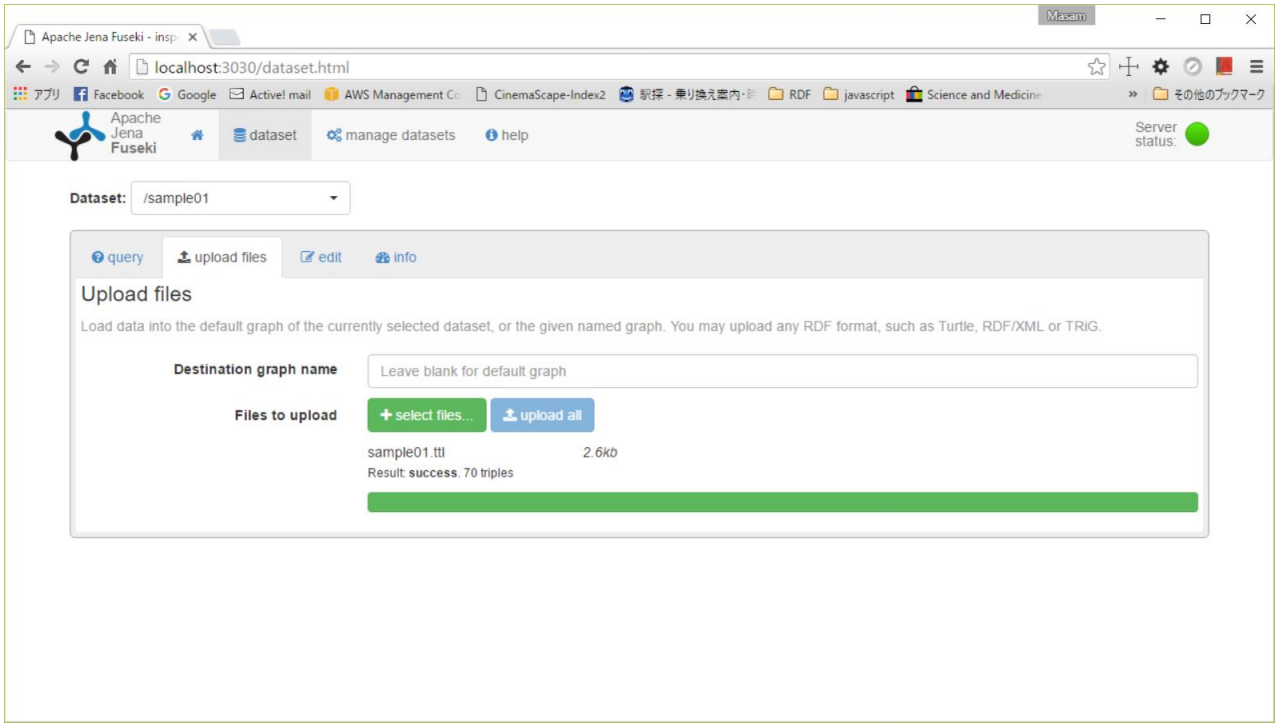


3.1.2.RDF データのアップロード

データセット「sample01」を選択し、「D:/work/moyo-sample/ja/sample01.ttl」をアップロードします。

- dataset　→　upload files を選択
- ファイルを指定

- upload all をクリック



3.1.3. ブラウザによる表示と操作

ブラウザを開いて URL「file:///D:/work/moyo-sample/ja/sample01.html」を表示します。（Google Chrome、Firefox で動作確認しました）この章の最初の図のように、複数の画面を開いて動作させてみてください。

- バーを操作すると
- バーの色はブラウザ側の乱数によって選択していますので、リロードすると色を変えることができます。
- 名前も乱数で生成していますが、入力欄に任意の文字列を指定できます。

3.1.4.SPARQL によるアクセス

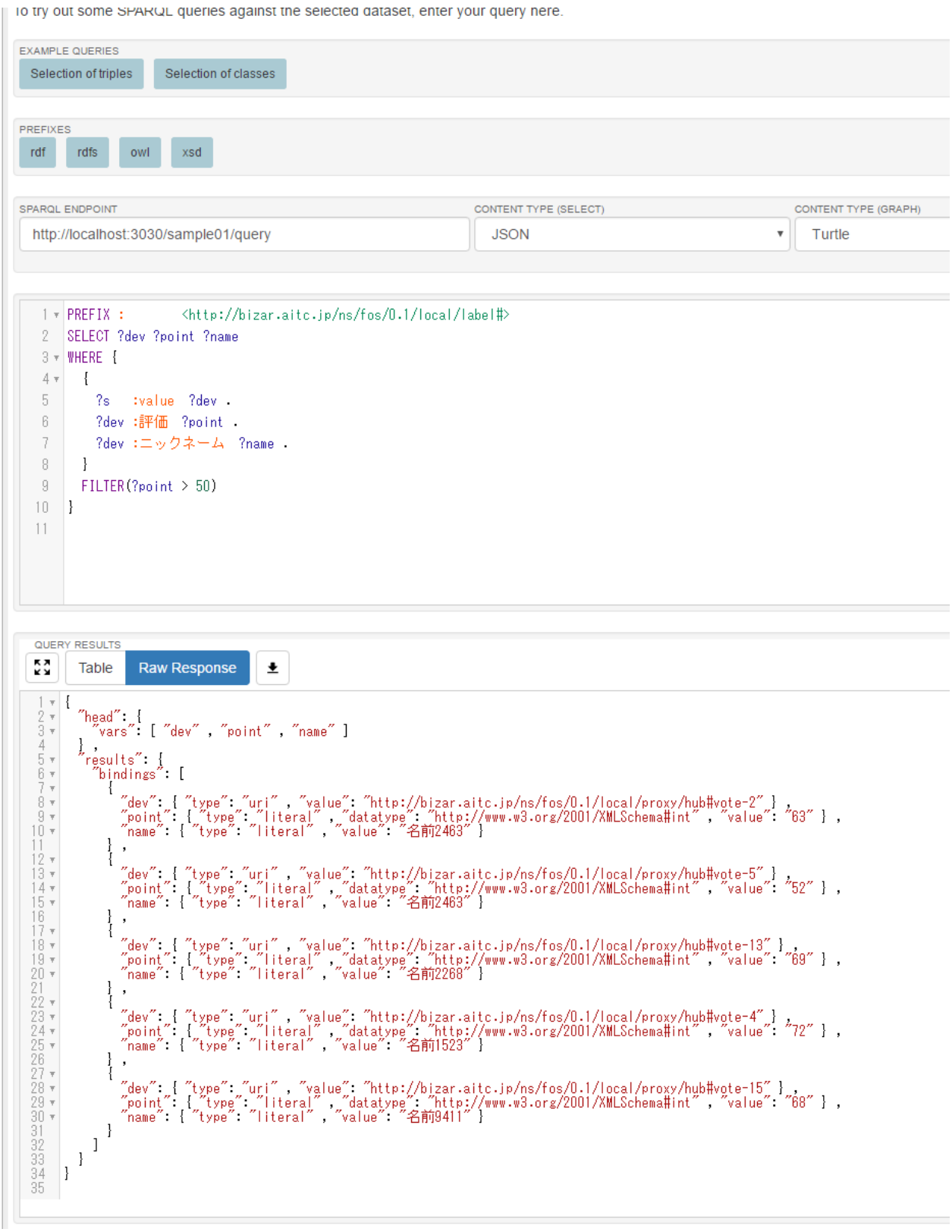
サーバから入力した値は RDF ストア内にただちに反映されますので、SPARQL で検索できます。

サーバ画面で query タブを開き、つぎのクエリを実行します。

```
PREFIX :       <http://bizar.aitc.jp/ns/fos/0.1/local/label#>
SELECT ?dev ?point ?name
WHERE {
  {
    ?s   :value  ?dev .
    ?dev :評価  ?point .
    ?dev :ニックネーム  ?name .
  }
  FILTER(?point > 50)
}
```

このクエリは、評価値が 50 より大きい物理ノードの評価値と入力者の名前を表示します。

前項に示した図に対応する結果がつぎの図です。



Moyo は、SPARQL クエリで「?point」などの変数へ値を代入するときに、現在値を代入します。クエリの中のパターンマッチで現在値を使うことができないことに注意してください。このような用途には Filter 文を使います。

3.1.5.SPARQL による値操作

SPARQL から物理ノードの値を操作することもできます。
つぎのクエリは、評価値が 50 より大きい物理ノードの値を、評価＝100、色=黒、ニックネーム＝「SPARQL から」へ書き換えます。（読出しのエンドポイントが、http://localhost:3030/sample01/sparql ですが、書き込みのエンドポイントは http://localhost:3030/sample01/update であることに注意してください）

```
PREFIX :       <http://bizar.aitc.jp/ns/fos/0.1/local/label#>
INSERT {
    ?dev :評価  100 .
    ?dev :色   "#000000" .
    ?dev :ニックネーム "SPARQL から" .
}
WHERE {
  {
    ?s   :value ?dev .
    ?dev :評価  ?point .
    ?dev :ニックネーム  ?name .
  }
  FILTER(?point > 50)
```

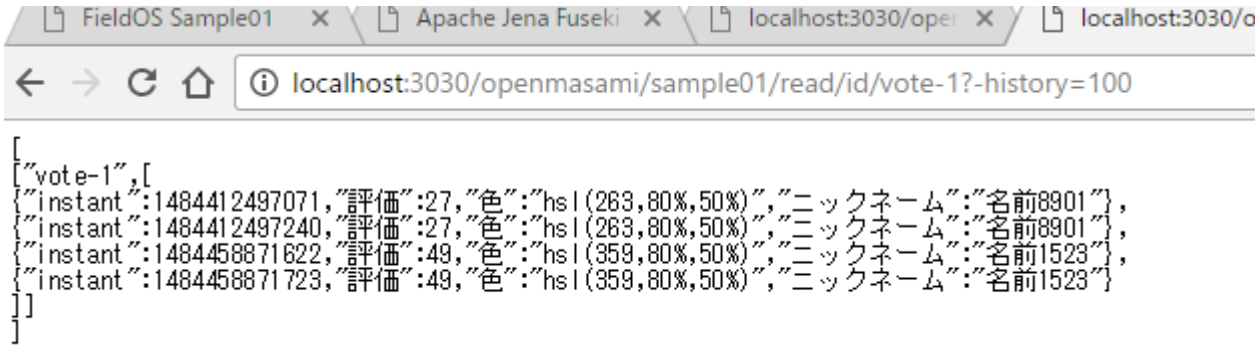
このクエリによって、前項の画面はつぎのように変わります。

3.1.6.履歴の表示

Moyo の拡張エンドポイントへの HTTP アクセスで、物理ノードの現在値や履歴を取得できます。
ブラウザから、つぎの URL を入力すると図のような応答が返ります。

```
http://localhost:3030/openmasami/sample01/read/id/vote-1?-history=100
```

このクエリは、「vote-1」という名前の物理ノードの値履歴を最大 100 個まで取得します。



vote-1 という物理ノードに 4 回設定された値を返しています。”instant”の項目は EPOC からのミリ秒で時刻を表現しています。
URL の記述にはいくつかの方法があり、またパラメータとして SPARQL クエリを渡すこともできます。

3.1.7.プログラムによる値の処理

「D:/work/moyo-sample/ja/python」フォルダに Python 2.7 で動作するスクリプトがあります。このフォルダにある pip.txt に必要なモジュールの記述がありますので、「pip –r pip.txt」コマンドによって環境をインストールします。

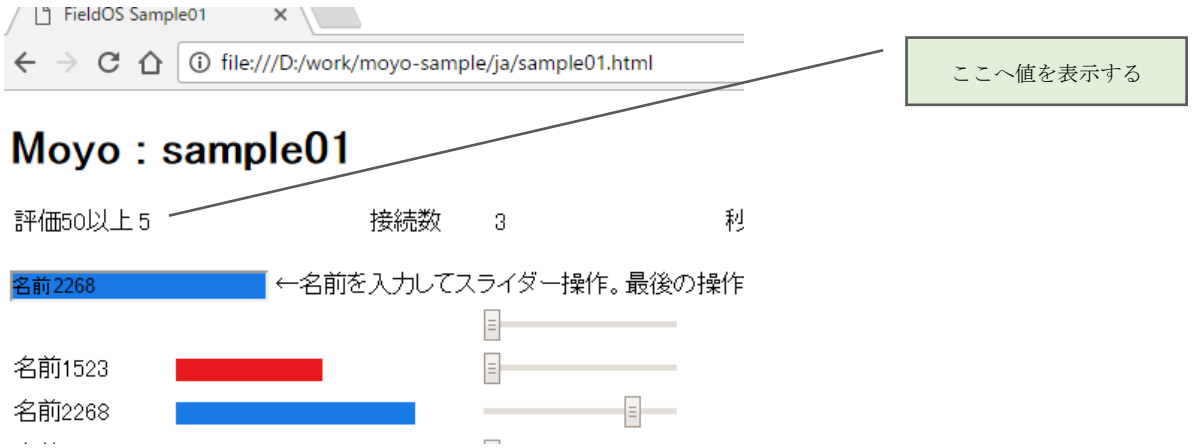
Positive.py

このプログラムは、評価値が 50 より大きい投票の数を、画面に表示します。バーの値変化をリアルタイムに検出して、数が変化すればただちに表示を書き換えます。

コマンドラインからつぎのように起動します。

```
> python Positive.py
```

実行結果は画面左上の「評価 50 以上」というフレーズの右側に表示します。



Heartbeat.py

このプログラムは、指定された個数のバーそれぞれを毎秒 100 サンプル程度の速度でデータを更新します。書き込む値は心電図を模したものです。「投票」とは無関係な動作ですが、Moyo が毎秒千個オーダーのイベントを扱えることを示すためのプログラムです。

コマンドライン引数として、書き込む対象とするバーの個数を渡します（1～10 程度の値が扱えます）。大きな値を渡すとブラウザの表示が間に合わなくなって JavaScript からの WebSocket 接続がきれるなどしますが、Moyo が記録するデータの更新履歴はほぼ 10ms 間隔を保っています。

コマンドラインからつぎのように起動します。

```
> python Heatbeat.py 3
```

この例を実行すると、上から 3 個のバーが脈動します。

3.2. Moyo プログラミング説明用サンプル

Moyo で物理ノードのデータを表現する RDF を説明するためのサンプルです。

このサンプルは、住宅での IoT 制御への適用例です。説明のために単純なデータ構造にしています。2 フロアにいくつかの部屋があり、部屋には、照明スイッチ、温度センサー、顔検出センサーがあるものとしています。

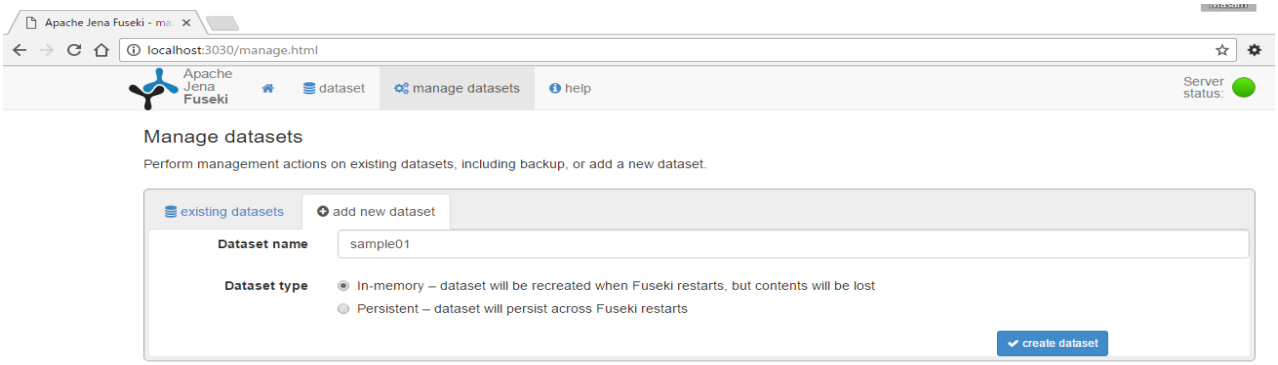
サンプルはつぎの 3 ファイルで構成しています。

- sample02.ttl
  RDF データです。Fuseki コンソールからアップロードします。
- sample02.html
  HTML ページです。ローカルファイル「file:///...」としてブラウザ表示します。
- sample02.js
  HTML ページから起動する JavaScript のプログラムです。Moyo へ WebSocket 接続して物理ノードの値を読み書きします。(WebSocket には同一オリジンポリシーが適用されないため、HTML ファイルをローカルファイルとして実行しても、JavaScript から Moyo サーバへアクセスすることができます。)

3.2.1. データセットの作成

- サーバ画面で「manage datasets」タブを選択し「add new dataset」を選択
- つぎの値を設定
  Dataset name　『sample02』
  Dataset type　『In-memory』 ←　現バージョンは In-memory でのみ動作します。
- create dataset をクリックします。

(図は Sample01 のものです。Sample01 を Sample02 に替えて操作してください)



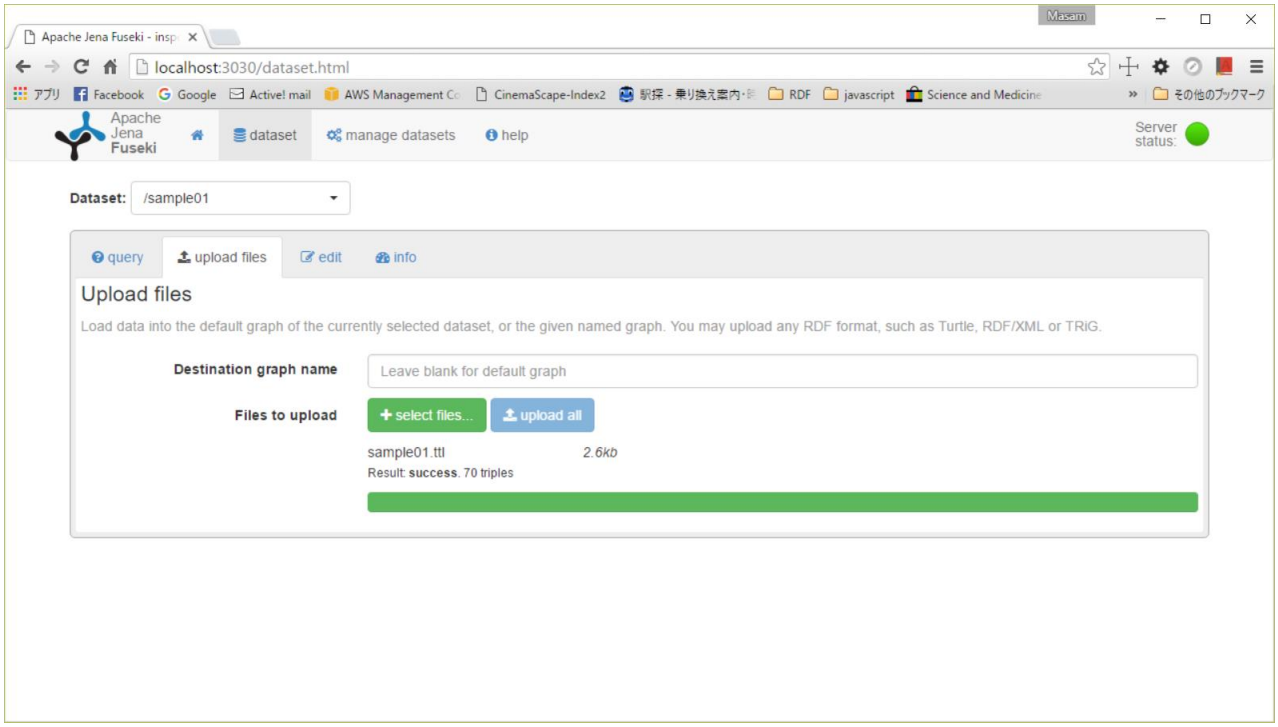
3.2.2. RDF データのアップロード

データセット「sample02」を選択し、「D:/work/moyo-sample/ja/sample02.ttl」をアップロードします。

- dataset　→　upload files を選択

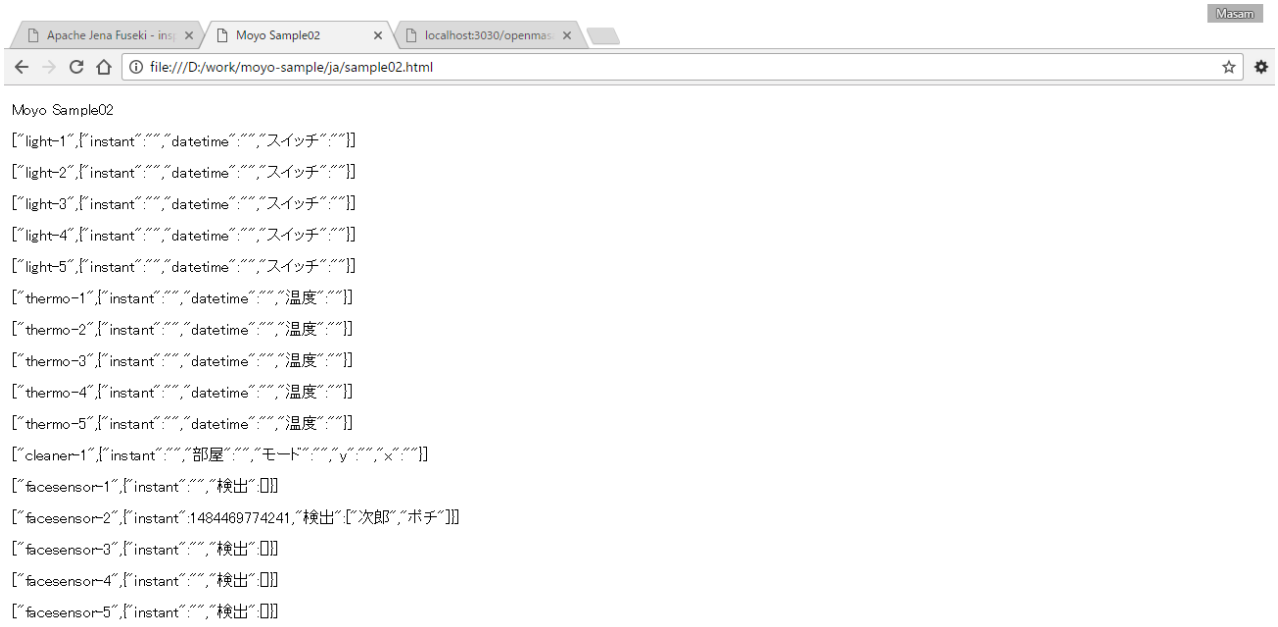
- ファイルを指定
- upload all をクリック

（図は Sample01 のものです。Sample01 を Sample02 に替えて操作してください）



3.2.3. ブラウザによる表示と操作

ブラウザを開いて URL「file:///D:/work/moyo-sample/ja/sample02.html」を表示します。



- 別のブラウザウインドウを開き、つぎの URL をインプットします。

```
http://localhost:3030/openmasami/sample02/update/path/1F/居間/気温?温度=20
```

- sample02.html の表示で、「thermo-1」の「温度」が 20 になります。

4. 解説

4.1. データ構造

4.1.1.物理ノードの表現

sample02.ttl で 1F 居間の気温センサーを表す物理ノードは、つぎのように表現しています。

```
@prefix :       <http://bizar.aitc.jp/ns/fos/0.1/local/label#>
@prefix fos:     <http://bizar.aitc.jp/ns/fos/0.1/>
@prefix leaf:   <http://bizar.aitc.jp/ns/fos/0.1/local/proxy/leaf#>
@prefix thermo:     <http://bizar.aitc.jp/ns/fos/0.1/local/proxy/hub#thermo->
thermo:1
      :温度            leaf:110;
      fos:instant     leaf:111;
      fos:datetaime   leaf:112;
      .
```

物理ノードは複数の値をもつことができます。この例では、「温度」「時刻(instant)」「日時(datetime)」の値を持ちます。
「http://bizar.aitc.jp/ns/fos/0.1/local/proxy/hub#<ID>」形式の IRI は、物理ノードを表します。
この例では「termo-1」を、この物理ノードを示す ID として設定しています。この ID は、Moyo が物理ノードの値を返すときに「id」という名前の要素の値として使います。Moyo のクライアントは、この id を使ってデータがどのノード由来かを区別することができます。また「id」を直接指定して、拡張エンドポイントから物理ノードへ値を書き込むことができます。

物理ノードの値は「http://bizar.aitc.jp/ns/fos/0.1/local/proxy/leaf#<LID>」形式の IRI で指定されたプレースホルダと結び付けられます。上記の例では、「:温度」「fos:instant」「fos:datetime」というプレディケートで、この物理ノードが 3 つの値を持つことをあらわしています。
この形式の IRI や<LID>は、データ構造の定義と Moyo の内部処理で使われ、通常のクライアントプログラムが直接扱うことはありません。
物理ノードの IRI を「http://bizar.aitc.jp/ns/fos/0.1/local/proxy/array#<LID>」形式の IRI で指定すると、配列値と結び付けられます。配列の要素数は 0 個以上で、値の順番は保存します。

現バージョンの Moyo では、物理ノードが指すプレディケートには、
「http://bizar.aitc.jp/ns/fos/0.1/local/label#<ラベル>」形式の IRI のみが使えます。

上記のように定義された物理ノードの値変化を WebSocket で受信したときには、つぎのような JSON 形式でデータが渡されます。

```
[["thermo-1",
{"温度":20,"instant":1472377713073,"datetime":"2016-08-28T09:48:33.073Z"}]]
```

サンプルページ「sample02.html」は、この値を表示しています。

プレースホルダを結びつける「:温度」「fos:instant」「fos:datetime」などのプレディケートの「温度」「instant」「datetime」の部分を値タグと呼びます。

- 物理ノードは、すべてのデータセットが共有します。
  同じ ID を持つ物理ノードは、すべてのデータセットに対して同じ値を読み書きします。
- 書きこみ操作で、データ構造の定義（sample01.ttl）にない値タグを使った場合には、その値を無視します。
- 値タグ「fos:instant」「fos:datetime」は予約語です。
  - 「fos:instant」には、Moyo が、物理ノードの値を設定した時刻をエポックからのミリ秒を表す整数値で設定します。
  - 「fos:datetime」には、Moyo が、物理ノードの値を設定した UTC 時刻を ISO8601 形式で設定します。
  - データ構造定義にこれらのタグがないときには、値は設定しません。
  - クライアント（センサー）側で測定時刻を付加するときには、「fos:instant」「fos:datetime」以外の値タグを使います。
- 物理ノードに測定値を設定するとき、データに指定されなかった値タグのデータは更新しません。古いデータが再度設定されたものとして記録します。
- 異なる物理ノードに同一の LID を持つプレースホルダを設定したときには、それらの物理ノードは同じ値を共有します。
- 書き込み操作のとき、物理ノードのすべての値はその時刻の値の組みにして記録します。
  - SPARQL で検索したとき、同一物理ノードに属する値は同時刻に記録された組み合わせのものを使います。（異なる物理ノードの値は、離れた時刻のものが使われることがあります）
  - 履歴には値の組み合わせ単位で記録します。
- 現バージョンの Moyo では、物理ノードの値として他の物理ノードを指定することはできません。

物理ノードは Moyo のすべてのデータセットで共有します。複数のデータセットで同じ ID の物理ノードを使うと、他のデータセットで設定された値を使うことができます。あるいは、あるデータセットでは日本語の RDF 表現、他のデータセットでは英語の RDF 表現で同一の物理ノードを取り扱うといった使い方ができます。

4.1.2.sample01.ttl の構成

sample01.ttl の概要は以下のとおりです。

- ＃＃デバイス定義
  物理ノードの定義です。
  - 5 個の温度計、5 個の照明スイッチ、1 個の掃除ロボット、5 個の顔センサーを定義しています。

- 無名ノード「_:device_index」は、sample01.html がすべてのデバイスの物理ノードを選択するのに使うインデックスです。

- ＃＃部屋
  部屋と、そこある備品を定義しています。
  - ＃家の構造
    2 つのフロアと、そこにある部屋を定義しています。階段は両方のフロアの構成要素になっています。

4.1.3.タグパス

拡張エンドポイントの動作確認で使ったつぎの URL は、「タグパス」と呼ぶ形式です。

```
http://localhost:3030/openmasami/sample02/update/path/1F/居間/気温?温度=20
```

タグパスは、RDF のデータ内にある木構造のパターンを探して物理ノードを特定します。
上記の URL で「/1F/居間/気温」の部分が、タグパスです。この探索のためにノードにタグをつけます。
タグは IRI「< http://bizar.aitc.jp/ns/fos/0.1/tag>」で指定します。

4.2. Fuseki 標準エンドポイントからのアクセス

Fuseki 標準エンドポイントから、SPARQL で物理ノードの読み書きができます。

4.2.1. 物理ノードの読出し

物理ノードの値を読み出すと、プレースホルダの IRI のかわりに現在値を返します。

```
PREFIX :       <http://bizar.aitc.jp/ns/fos/0.1/local/label#>
SELECT ?s1 ?s2
WHERE {
     ?s1 :温度 ?s2 .
}
```

Moyo は、SPARQL クエリで「?s2」などの変数へ値を代入するときに、現在値を代入します。クエリの中のパターンマッチで現在値を使うことができないことに注意してください。このような用途には、Samploe01 で示したような Filter 文を使います。

4.2.2. 物理ノードへの書き込み

プレースホルダへ値を書き込むと、現在値として記録します。（読出しのエンドポイントが、http://localhost:3030/sample02/sparql ですが、書き込みのエンドポイントは http://localhost:3030/sample02/update であることに注意してください）

```
PREFIX fos:       <http://bizar.aitc.jp/ns/fos/0.1/>
PREFIX :       <http://bizar.aitc.jp/ns/fos/0.1/local/label#>
INSERT{ ?hub :温度 21 . }
WHERE {
     ?s1 fos:tag :1F .
     ?s1 ?p1      ?s2 .
     ?s2 fos:tag :居間 .
     ?s2 :気温     ?hub .
}
```

値を更新するときには DELETE を使わないでください。現実装では、DELETE を実行したときの動作は不定です。

現実装では、つぎのデータ型を判別しており、これ以外はすべて言語指定なしの文字列として記録します。

現実装では桁あふれを検出していないため、64bit で表現できる範囲の数値のみ扱ってください。

- 符号か数字ではじまり、すべてが数字なら整数と判別する。
- 符号か数字ではじまり、1 個のピリオド以外すべて数字なら浮動小数点数と判別する。

すべてのプレースホルダの初期値は長さゼロの文字列です。

4.3. 拡張エンドポイントの使い方

拡張エンドポイントは、物理ノードの値の読み書きに使います。

- GET、POST、PUT
  指定された物理ノードの現在値を読み書きします。
  現実装では、これらのメソッドはすべて同じ動作をします。

- WebSocket
  指定された物理ノードの値を読み書きしつづけます。
  - 読出し：物理ノードへ値が書き込まれると、書き込まれた値をクライアントへ送信します。
  - 書き込み：クライアントが値を送信すると、物理ノードの値を更新します。

以下の記述は、GET、POST、PUT、WebSocket すべてに共通です。

現バージョンの Moyo は、HTTP のメソッドとして、GET、PUT、POST のみに対応しています。

4.3.1. URL の構造

拡張エンドポイントの URL の構造はつぎの通りです。

```
http://<拡張パス>/<データセット>/<動作＞/<指定形式>/<ノード指定>?[<オプション指定>&]<値指定>
例：
http://localhost:3030/openmasami/sample01/update/path/居間/気温?-link=構成&温度=19
```

- 拡張パス
  Fuseki のエンドポイントパスに文字列「openmasami」を付加したものです。
  「openmasami」はデータセット名と一致させないための無意味な（だが覚えられなくもない）文字列です。
- データセット
  データセット名を指定します。
- 動作
  読出し「read」または書き込み「update」を指定します。
- 指定形式
  ノード指定の形式を指定します。
  - id
    物理ノードの ID を 1 個指定します。
    例：`http://localhost:3030/openmasami/sample02/update/id/thermo-1?温度=18`
  - path
    タグパスで指定します。
    [<ノード指定>/...]<値名>の形式です。ノード指定は「:パス識別子」で指定するタグです。値名は物理ノードを指す Predicate のラベルです。現実装ではどちらも空間 OS の基底名前空間を使います。
    - ノード指定では、木構造を辿る Predicate を指定できます。
      たとえば「1F/構成-居間/気温」は、タグ「:1F」のノードから Predicate「:構成」を辿って「:居間」に至ることを示します。
      デフォルトではあらゆる Predicate にマッチしますが、オプション指定「-link」で指定しなかったと

きに辿る Predicate を指定できます。

- query
  SPARQL クエリをオプション指定「-query」で指定します。
  指定するクエリは SELECT です。SELECT で指定する変数は 1 個でなければなりません。

- オプション指定と値指定
  URL の？以下にこれらを指定します。「名前=値」ペアを「&」でつないだものになります。
  名前が半角マイナスで始まる値指定を「オプション指定」と呼び、Moyo の動作へのパラメータとして使います。
  それ以外の値指定は、物理ノードの値の名前を指定します。
  実装済みのオプション指定を以下に列挙します。
  - -link
    前述の通り、タグパスが辿る Predicate のデフォルト値を指定します。
  - -history=<数値>
    ＜数値＞を指定すると、指定した物理ノードの値の履歴のうち最新の＜数値＞件を返します。＜数値＞にゼロを指定するとすべての履歴を返します。WebSocket 接続のときには、すべての値を送信したあと長さゼロの文字列を返します。
  - -latest=[0|1]
    WebSocket 接続の時、1 を指定すると現在値を送信します。0 を指定すると現在値の送信をしません。
    「-history」と「-latest」を両方指定したときには、「履歴→長さゼロ文字列→現在値」の順で送信します。
  - -query=<SPARQL クエリ>
    現在は指定形式が「query」のときにのみクエリの指定に使います。

4.3.2.URL の例

```
http://localhost:3030/openmasami/sample02/read/path/居間/気温
http://localhost:3030/openmasami/sample02/update/path/居間/気温?温度=25
http://localhost:3030/openmasami/sample02/update/id/thermo-5?温度=21
http://localhost:3030/openmasami/sample02/update/id/cleaner-1?x=123&y=345&モード=停止
http://localhost:3030/openmasami/sample02/update/path/1F/居間/顔?検出=太郎&検出=花子
http://localhost:3030/openmasami/sample02/read/path/気温?-history=10
```

URL のパラメータで配列値をわたしたときには、要素の順番を保存しないことがあります。

4.4. 主な制約

- 物理ノードのプレースホルダ（leaf,array）を主語としたデータ構造を登録しないでください。
- プレースホルダへ物理ノード以外のノードからリンクしないでください。
- プレースホルダの値を指定したパターンマッチはできません。
  たとえば、つぎの SELECT 文で温度が 4 の物理ノードを検索できません。このようなケースでは FILTER を使ってください。

```
PREFIX :       <http://bizar.aitc.jp/ns/fos/0.1/local/label#>
SELECT ?s1
WHERE {
     ?s1 :温度 20 .
}
```

値「20」は物理ノードの現在値とマッチしません。以下のクエリで 20 度のノードを検索できます。

```
PREFIX :       <http://bizar.aitc.jp/ns/fos/0.1/local/label#>
SELECT ?s1
WHERE {
    {
      ?s1 :温度 ?temp .
    }
    Filter(?temp = 20)
}
```