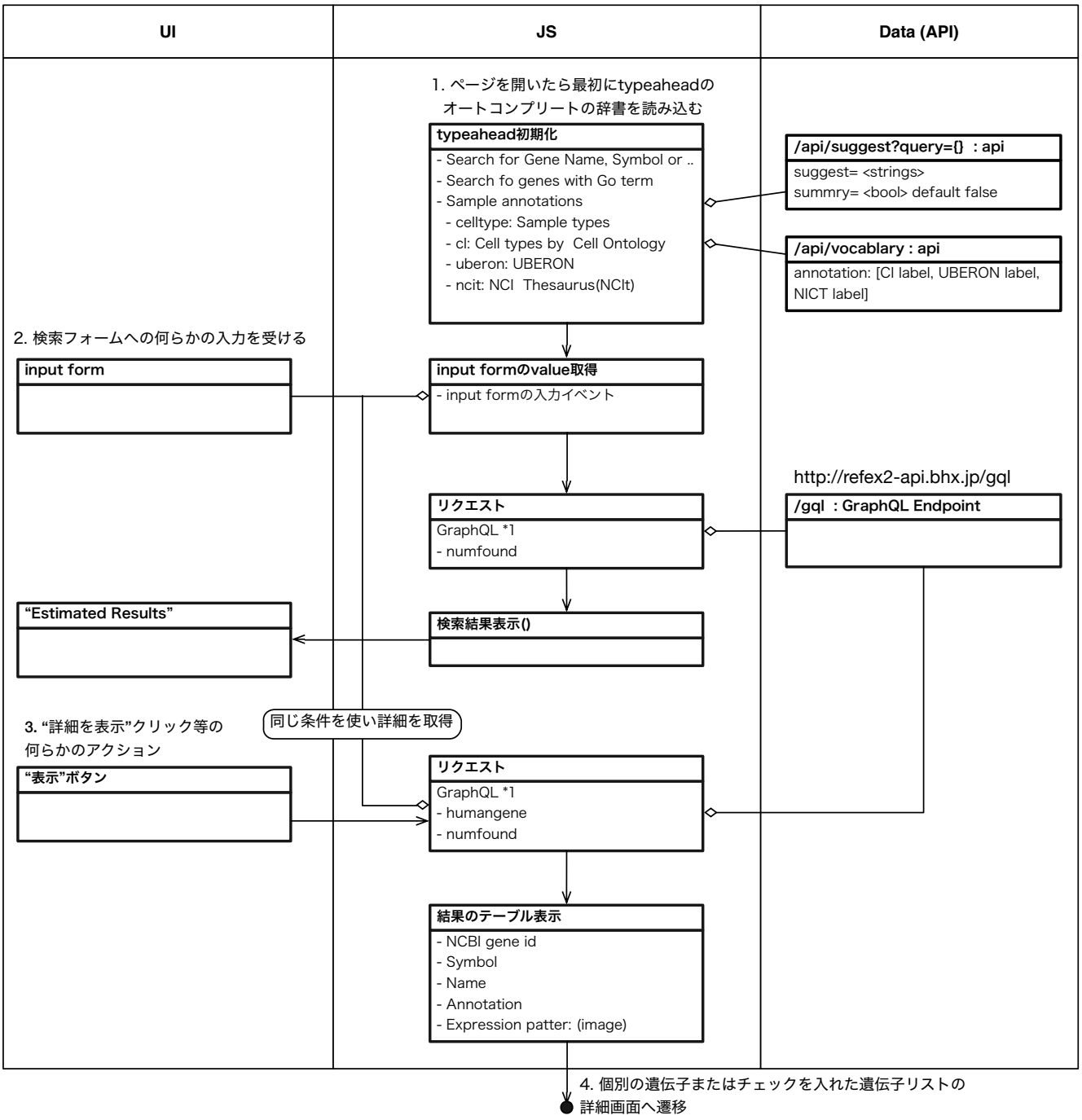
2020-10-30

refex2 検索ページ構築用資料

■ データ取得と結果表示プロセス



2020-10-30

■ GraphQLサンンプル

現在検索に使えるGraphQLのフィルター変数

text：Search for genesの値を渡す
geneid: カンマ区切り文字列。Gene id listを渡すことができる
go: go idを渡す。例. "GO:0010977"
summary: “True”を返すとsummary検索対象として含める
organism: 生物種. デフォルトで”human”
project: デフォルトで”fantom5”
#以下サンプル特的発現のアノテーション
uberon: サンプル特異性検索. UBERON,
cl: サンプル特異性検索.Cell Ontology
ncit: サンプル特異性検索.Biomedical concept
celltype: サンプル特異性検索.Sample types

リクエストの例

例０. 初期状態の遺伝子数

```
{
  numfound
}
```

POSTだと…
curl -X POST -H "Content-Type:application/json" -d'{"query":"{numfound}"}' http://refex2-api.bhx.jp/gql

---

例１. テキスト検索フォームで”BRCA”と検索

```
{
  humangene(text: "BRCA") {
    ncbiGeneId
    symbol
    name
    alias
  }
  numfound
}
```

POSTだと…
curl -X POST -H "Content-Type:application/json" -d'{"query":"{humangene(text:\"BRCA\")
{ncbiGeneId symbol name alias} numfound}"}' http://refex2-api.bhx.jp/gql

---

例２. 上記同様”BRCA”の検索で検索結果の件数のみ取得（numfoundにフィルター引数を渡す）

```
{
  numfound(text: "BRCA")
}
```

POSTだと…
curl -X POST -H "Content-Type:application/json" -d'{"query":"{numfound(text:\"BRCA\")}"}'
http://refex2-api.bhx.jp/gql

例３. サマリフィールドも含めてテキスト検索する

```
{
  humangene(text: "nervous", summary: "True") {
    ncbiGeneId
    symbol
    name
    alias
  }
  numfound
}
```

POSTだと…
curl -X POST -H "Content-Type:application/json" -d'{"query":"{humangene(text:
\"nervous\", summary:\"True\"){ncbiGeneId symbol} numfound}"}' http://refex2-
api.bhx.jp/gql

例４. テキスト検索の結果をGO Termでフィルターする

```
{
  humangene(text: "nervous", summary: "True", go: "GO:0010977") {
    ncbiGeneId
    symbol
    name
    alias
  }
  numfound
}
```

POSTだと…

```
curl -X POST -H "Content-Type:application/json" -d'{"query":"{humangene(text:
\"nervous\", summary:\"True\", go:\"GO:0010977\"){ncbiGeneId symbol} numfound}"}'
http://refex2-api.bhx.jp/gql
```

例５. テキスト検索の結果をサンプルアノテーションのCell Ontology（cl）でフィルターする

```
{
  humangene(text: "nerve", summary: "true", cl: "neuroblast") {
    ncbiGeneId
    symbol
    name
    alias
  }
  numfound
}
```

POSTだと…

```
curl -X POST -H "Content-Type:application/json" -d'{"query":"{humangene(text:\"nerve\",
summary:\"True\", cl:\"neuroblast\"){ncbiGeneId symbol} numfound}"}' http://refex2-
api.bhx.jp/gql
```

■ typeaheadの利用について

テキスト補完のtypeahead.jsはbloodhoundモジュールを通してリモートのデータベースを利用している。現状refex2では遺伝子の情報、GOの記述、サンプルアノテーションの記述の大きく三種（サンプルアノテーションはさらに四種）にtypeahead-bloodhoundを使っている。typeaheadを継続して使う必要が無いが、テキスト補完データベース利用の例としてbloodhoundの実装をあげておく。

テキスト検索に利用する遺伝子情報に関するテキストの補完

```
  // bloodhoundの定義
   var bl_gene_info = new Bloodhound({
          datumTokenizer: Bloodhound.tokenizers.obj.whitespace('Symbol', 'value', 'id', 'alias'),
          queryTokenizer: Bloodhound.tokenizers.whitespace,
          remote:{
              wildcard:'%QUERY',
              url:refex_api + 'suggest?query=',
              replace: function (url, uriEncodedQuery) {
                  var res = url + uriEncodedQuery + query_option();
                  return res
              },
              transform: function (response) {
                  gene_info = response.results;
                  return $.map(gene_info, function (bl) {
                      return bl
                  });
              }
          }
      });

   // typeaheadからデータソースにbloodhoudを指定
   var th = $('#refex .typeahead').typeahead(null, {
          name: 'gene_info',
          limit: 20000,

          templates: {
              empty: 'input gene name, id, or official symobol',
              suggestion: function (data) {
                  var symbol = data.symbol,
                      alias = data.alias,
                      id = data.entrezgene,
                      name = data.name;
                  return '<div><span class="symbol">' + symbol + '</span> (' + name + ' ' + alias + ',
NCBI_GeneID: ' + id + ' )</div>'
              }
          },
          source: bl_gene_info,
          display: 'symbol'
      }
   );
```

GO Termの補完

```
// bloodhoundの定義
 var bl_go = new Bloodhound({
        datumTokenizer: Bloodhound.tokenizers.obj.whitespace('id', 'term'),
        queryTokenizer: Bloodhound.tokenizers.whitespace,
        remote:{
            wildcard:'%QUERY',
            url: refex_api + 'suggest?go=True&query=%QUERY',
            transform: function (response) {
                var go_info = response.results;
                return $.map(go_info, function (bl) {
                    return bl
                });
            }
        }
    });

 var thgo = $('#go .typeahead').typeahead(null, {
        name: 'go_term',
        limit: 20000,
        templates: {
            empty: 'input go term',
            suggestion: function (data) {
                var goid = data.id,
                    term = data.term,
                    gocategory = data.gocategory;
                gocategory = gocategory ? gocategory: "";
                return '<div>' +  term  +  ', ' + gocategory +  ' ' + goid + '</div>'
            }
        },
        source: bl_go,
        display: 'term'
    }
 );
```

GO Termによるフィルターは最終的にgo idを抜き出す必要があり、go idを非同期的に文字列から取得するトリッキーな実装が必要になるかも（typeaheadを使う場合）。goidにクラス指定とか上記の段階でしておくといいのカモ。

サンプルアノテーションの補完（四種ある）

```
// bloodhound設定
  // sample_typeのみ値が４つしかないのでデータベースは参照しないでリストをハードコードしている
    var bl_sample_type = new Bloodhound({
        datumTokenizer: Bloodhound.tokenizers.whitespace,
        queryTokenizer: Bloodhound.tokenizers.whitespace,
        local: ["cell lines", "stem cells", "primary cells","tissues"]
    });


    var bl_cell_ontology = new Bloodhound({
        datumTokenizer: Bloodhound.tokenizers.whitespace,
        queryTokenizer: Bloodhound.tokenizers.whitespace,
        prefetch: 'http://refex2-api.bhx.jp/api/vocablary?annotation=CL%20label'
    });
    var bl_tissue_type= new Bloodhound({
        datumTokenizer: Bloodhound.tokenizers.whitespace,
        queryTokenizer: Bloodhound.tokenizers.whitespace,
        prefetch: 'http://refex2-api.bhx.jp/api/vocablary?annotation=UBERON%20label'
    });
    var bl_nict_label= new Bloodhound({
        datumTokenizer: Bloodhound.tokenizers.whitespace,
        queryTokenizer: Bloodhound.tokenizers.whitespace,
        prefetch: 'http://refex2-api.bhx.jp/api/vocablary?annotation=NCIT%20label'
    });
// typeahead設定
    $('#sample_type .typeahead').typeahead(null, {
        name: 'sample_types',
        source:bl_sample_type
    });
    $('#cell_ontology .typeahead').typeahead(null, {
        limit:25,
        source: bl_cell_ontology
    });  // 他２つのフォームも同様
```