POSEIDON-N1

開発者向け説明書

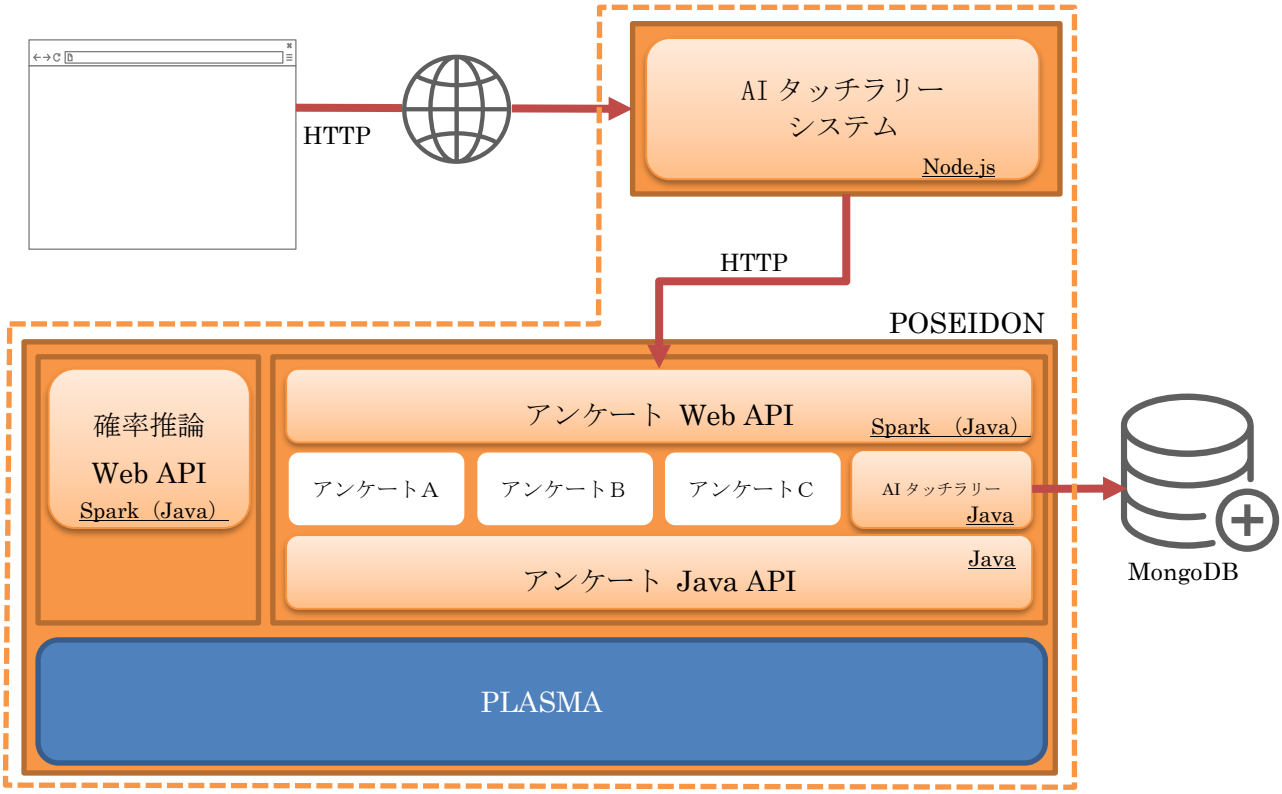
国立研究開発法人産業技術総合研究所

2020/8/14 改訂　2020/8/19 微修正

内容

1. システム概要

POSEIDON はアンケートデータ収集、アンケートに基づいたレコメンドを実施することを目的としたシステムです。レコメンドについては PLASMA と連携しベイジアンネットワークによる確率推論機能が利用できます。



構成としては「POSEIDON（汎用アンケート・レコメンド API）」をベースとし、そのアプリケーションの一つとして「AI タッチラリーシステム」が構築されています。AI タッチラリーシステムは GUI（管理画面やアンケート実施画面）の実装が主な役割です。ユーザー（ブラウザ）からのリクエストを受け、解釈し、POSEIDON から HTTP 通信により必要な情報を取得、UI を構築しレスポンスをブラウザに返します。

2. 汎用アンケートエンジン（Java API）

2.1. 拡張の仕組

アンケートは SlideFactory を実装することで拡張できるようになっています。SlideFactory とはその名前の通り Slide を生成するクラスです。Slide とはスライドショーに該当するクラスで複数のページを持ち、その表示順を管理します。

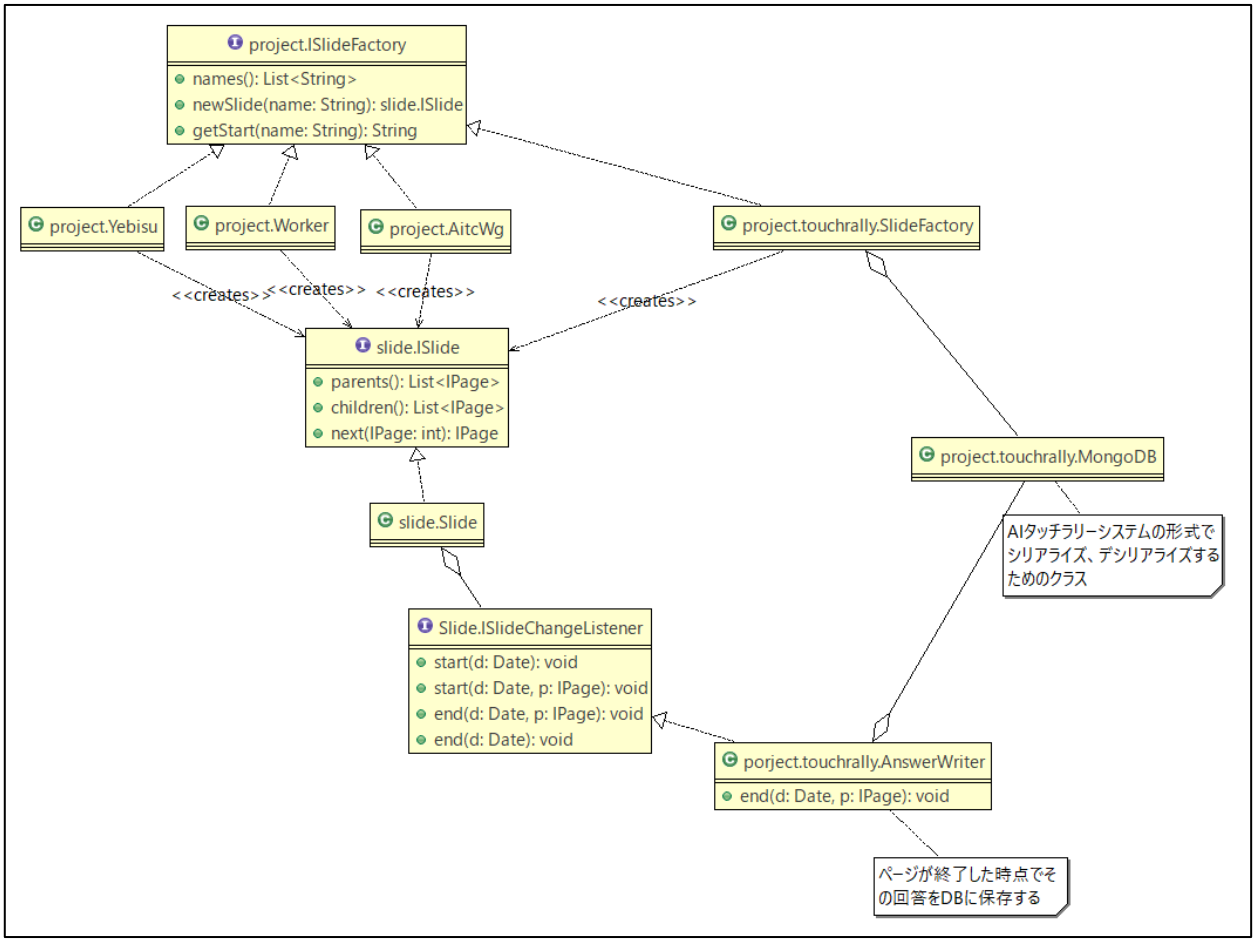
SlideFactory を固定のスライドを生成するようにハードコーディングすることもできますし、設定ファイルの内容に従って生成するような動的な実装もできます。後者については、現在、AI タッチラリーシステム用に MongoDB に登録された定義にしたがって Slide を生成するクラスを実装しています。

2.1.1. SlideFactory

現在のところ SlideFactory の実装は次の四つです。

`Yebisu、Worker、AitcWg、touchrally.SlideFactory`

Yebisu、Worker、AitcWg は SlideFactory をハードコーディングしており、固定の Slide を一つ生成するだけのクラスです。一方、touchrally.SlideFactory はデータベースに登録された定義に従って、複数の Slide を生成できるように実装されています。



ISlideFactory

```
public interface ISlideFactory {
    /** スライド名を取得する */
    List<String> names();

    /**
     * スライドを生成する
     * @param name スライド名
     * @return 生成されたスライド
     */
    ISlide newSlide(String name);

    /**
     * スライドの開始ページの ID を取得す
     * @param name スライド名
     * @return スライドの開始ページの ID を取得する
     */
    String getStart(String name);
}
```

SlideFactory を新しく実装した場合は、下記のように設定ファイルに追記することで Web API として公開します。

設定ファイル：enquete.yaml

```
touchrally:
  slideFactory:
    className: jp.go.aist.trident.project.TouchrallyProject
    args:
      dbHost: localhost
      dbPort: 27017
      dbName: touch-rally
worker:
  slideFactory:
    className: jp.go.aist.trident.project.WorkerProject
    args:
      bifFilePath: worker.bif
```

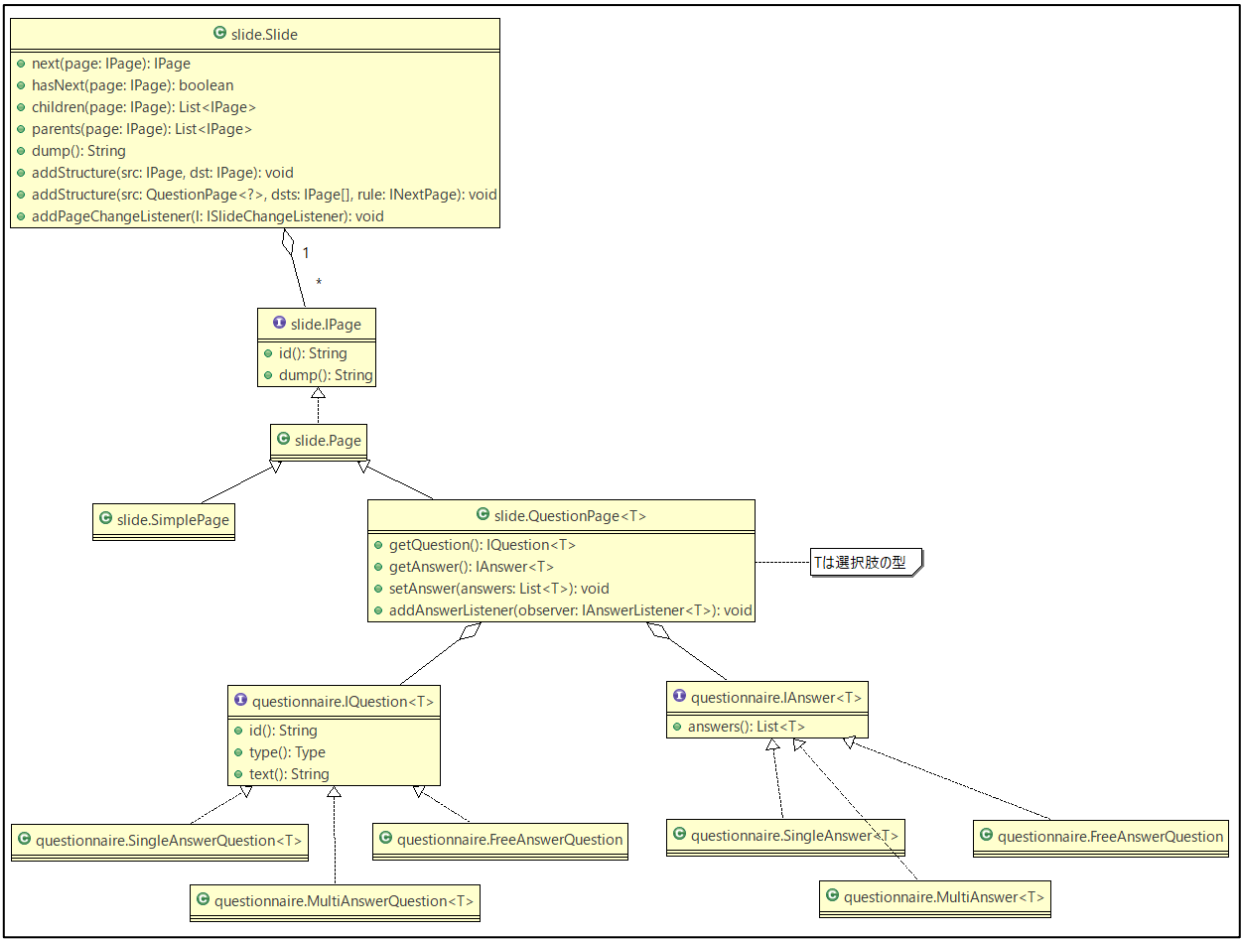
このように設定ファイルを書くことで、次のような URL にアクセスすることで、各クラスで定義されたアンケートを実施することができるようになります。

```
http://…/api/eq/user?factory=touchrally&slide=entrance
```

2.1.2. Slide

Slide とはここではスライドショーのように、複数のページを所定の順番で表示するためのクラスで、ページを頂点とする有向グラフを構成します。複数の子を持つ頂点には遷移ルールを設定することができ、今までの回答結果を条件式で参照することができます。

ページには、静的なページに対応する SimplePage クラス、アンケートに対応する QuestionPage クラスが実装されています。



2.1.3. SlideFactory 及び Slide の実装

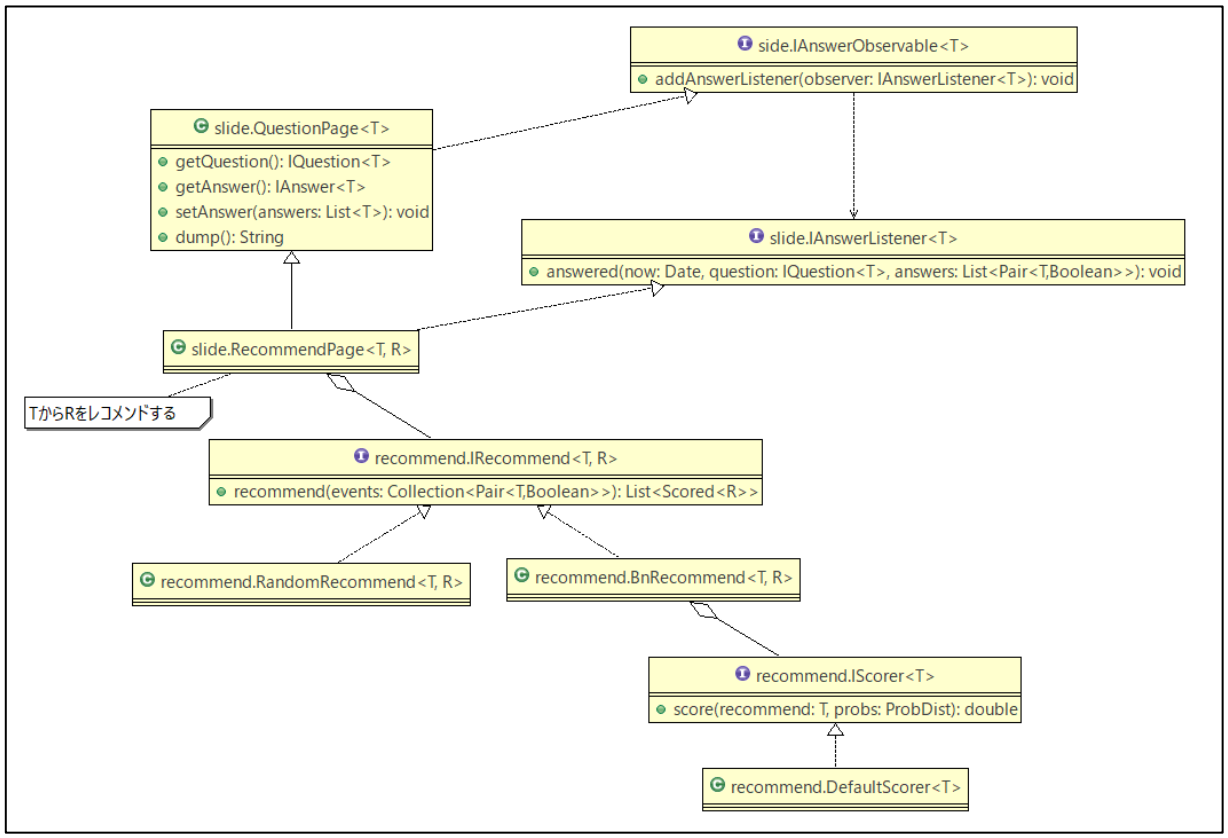
性別と年齢の二問のアンケートを実施し、最後にサンクスページを表示するスライドを生成するファクトリークラスの実装例を以下に示します。ここではスライド及びページの内容をハードコーディングしています。

```
/**
 * "aSlide" という名前の一つのスライドを生成するためのクラス
 */
public class Sample implements ISlideFactory {

    /**
     * 生成可能なスライド名のリスト
     */
    @Override
    public List<String> names() {
        return Arrays.asList("aSlide");
    }

    /**
     * 指定名のスライドを生成
     */
    @Override
    public ISlide newSlide(String name) {
        assert(name.equals("aSlide"));
        QuestionPage<String> p1 = new QuestionPage<>(
                new SingleAnswerQuestion<String>(
                        "P1",                           // ID
```

```
                    "あなたの性別を教えてください。", // 設問
                    new Choice<>(new String[] {"male", "female"})); // 選択肢

        QuestionPage<String> p2 = new QuestionPage<>(
                new SingleAnswerQuestion<String>(
                        "P2",                                   // ID
                        "あなたの年齢を教えてください。", // 設問
                        new Choice<>(new String[] {"20 代以下", "30 代", "40 代", "50 代", "60 代以上"}))); // 選択肢

        SimplePage p3 = new SimplePage("P3", "アンケートは終わりです。");

        Slide slide = new Slide();
        slide.addStructure(p1, p2); // グラフ構造 {親 (p1) → 子 (p2) } を登録
        slide.addStructure(p2, p3); // グラフ構造 {親 (p2) → 子 (q3) } を登録
        return slide;
    }

    /**
     * 指定スライドの開始ページ
     */
    @Override
    public String getStart(String name) {
        return "P1";
    }
```

2.2. レコメンドの仕組

レコメンドは RecommendPage と QuestionPage が Observer-Observable の関係を持ち、QuestionPage がその回答を RecommendPage に通知する仕組みを利用して実装しています。RecommendPage は IRecommend を持ち、QuestionPage からの通知（メッセージ）を IRecommend に引き渡します。

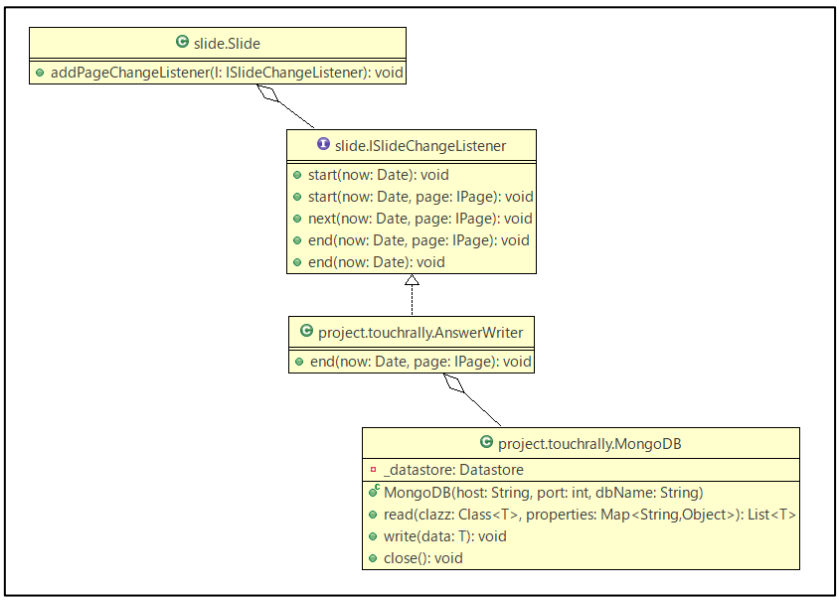


IRecommend は総称型として T と R に依存しており、T 型のインスタンスを受け取り R 型のインスタンスをレコメンドします。現在のところ入力を無視してランダムにレコメンドを行うクラス、ベイジアンネットワークの確率推論結果からスコアを計算するクラスを実装しています。

スコア計算用のインターフェース IScorer では、総称型 T のインスタンスと事後確率分布を引数として受け取り、T のインスタンスのスコアを計算します。例えば T が価格という属性を持てば、T の購買確率を掛け合わせて期待値をそのスコアとするようなことを想定しています。

2.3. 回答保存の仕組

Slide には SlideChangeListener が登録できるようになっており、回答を保存するクラスはこのインターフェースを実装します。現在は AI タッチラリーシステム用に project.touchrally.AnswerWriter クラスが実装されています。ページの表示が終了した時点で呼び出される end メソッドでデータベースへの登録処理を実装しています。



SlideChageListener は設定ファイルに追記することで Slide に登録されるようになります。任意個のリスナーを登録できるようになっています。

設定ファイル：enquete.yaml

```
touchrally:
  slide:
    className: jp.go.aist.trident.project.TouchrallyProject
    args:
      dbHost: localhost
      dbPort: 27017
      dbName: touch-rally
  slideChangeListener:
    -
      className: jp.go.aist.trident.project.touchrally.AnswerWriter
      args:
        dbHost: localhost
```

```
        dbPort: 27017
        dbName: touch-rally
    -
      className: jp.go.aist.trident.project.touchrally.TestWriter
      args: {}
worker:
  slide:
    className: jp.go.aist.trident.project.WorkerProject
    ＜以下省略＞
```

3. 汎用アンケート Web API

汎用アンケートエンジンを HTTP 通信で利用するための Web API です。

HTTP メソッド	API	機能概要
POST	/api/eq/user/:slide_id	回答者 ID（respondent ID）を発行し、スライドを開始する
GET	/api/eq/flow/:respondent_id	スライドのページ遷移を取得する
GET	/api/eq/next/:respondent_id	次のページを取得する
GET	/api/eq/back/:respondent_id	前のページを取得する
GET	/api/eq/data/:respondent_id/:page_id	ページ（:page_id）の内容を取得する
PUT	/api/eq/:respondent_id/:page_id	ページ（:page_id）のアンケートに回答する
DELETE	/api/eq/user/:respondent_id	回答者（:respondent_id）のスライドを終了し破棄する

3.1. スライドの開始

リクエスト

```
POST /api/eq/user
```

（クエリパラメータ）

name	type	説明
factory	String	アンケート定義ファイルのファクトリ名
slide	String	スライド名

例）

```
http://…/api/eq/user?factory=sample&slide=aSlide
```

レスポンス

Key	型	サイズ	必須	値の説明
respondent_id	String	-	○	回答者のユニークな ID

例）

```
{
    "respondent_id": "b1bd0a7d-c6b8-4195-8244-95bc5ef9b2f0"
}
```

3.2. ページ遷移の取得

開始時に指定したスライドのページ遷移を取得する。

リクエスト

```
GET /api/eq/flow/:respondent_id
```

例）

```
http://…/api/eq/flow/b1bd0a7d-c6b8-4195-8244-95bc5ef9b2f0
```

レスポンス

Key	型	サイズ	必須	値の説明
id	String	-	○	スライド id
start	String	-	○	スライドの開始設問 page_id
flow	Object Array	-	○	以下のオブジェクトの配列
flow.**src**	String	-	○	遷移元の page_id
flow.**dst**	String	-	○	遷移先の page_id
flow.**rule**	String	-		

例）

```
{
    "start":"P1",
    "id":"aSlide",
    "flow":[
        {
            "src":"P1",
            "dst":"P2"
        },
        {
            "src":"P2",
            "dst":"P3"
        }
    ]
}
```

3.3. 次のページの取得

次のページ（ID）を取得する。

リクエスト

```
GET /api/eq/next/:respondent_id
```

例）

```
http://localhost:4567/api/eq/next/ b1bd0a7d-c6b8-4195-8244-95bc5ef9b2f0
```

レスポンス

レスポンス

Key	型	サイズ	必須	値の説明
status	String	-	○	"has"(次がある), "finish"(次がない),"not decided"(確定していない)のいずれか
next	String	-	-	status が"has next"の場合、次に表示すべき設問の page_id

例）

```
{"next":"P2","status":"has"}
```

3.4. ページの内容の取得

指定ページの詳細を取得する。

リクエスト

```
GET /api/eq/data/:respondent_id/:page_id
```

例）

```
http://…/api/eq/data/b1bd0a7d-c6b8-4195-8244-95bc5ef9b2f0/P1
```

レスポンス

例）

```
{
    "id":"P1",
    "type":"question",
    "sub_type":"single",
    "html":"あなたの性別を教えてください。",
    "choice_list":[
        {
            "id":"1",
            "contents":"male"
        },
        {
            "id":"2",
            "contents":"female"
        }
    ]
}
```

3.5. アンケートに回答する

指定ページ（アンケート）に回答を設定する。

```
PUT /api/eq/:respondent_id/:page_id
```

リクエスト

Key	型	サイズ	必須	値の説明
answer	String Array	-	○	回答した choice_id

例）

```
{
  "answer": [
    "1"
  ]
}
```

3.6. スライドを終了する

スライドを終了する。呼び出し以降、回答者 ID は使用できない。

```
DELETE /api/eq/user/:respondent_id
```

4. 確率推論 Web API

システムで管理しているモデルに対し、その構造の取得や、確率推論を実行することできます。

現在はモデルの登録や収集したデータからのモデルの更新機能は実装されていません。

4.1. モデル一覧取得

登録されているモデルの一覧を取得する。

リクエスト

```
GET /api/bayesnet
```

例）

```
http://…/api/bayesnet
```

レスポンス

例）

```
["nedofesta","univ_shindan","worker"]
```

4.2. 指定モデルのバージョン一覧を取得

指定モデルのバージョン一覧を取得する。

リクエスト

```
GET /api/bayesnet/:modelName/model
```

例）

```
http://…/api/bayesnet/nedofesta/model
```

レスポンス

例）

```
["191213_110017","191212_211153","191212_204749","191212_204308"]
```

4.3. 指定モデルを取得（最新）

指定モデルの最新バージョンを取得する。

リクエスト

```
GET /api/bayesnet/:modelName/model/current
```

（クエリパラメータ）

name	type	説明
type	String	JSON または HTML を指定
format	Boolean	type が **JSON** の場合、**true** を指定するとインデント及び改行で整形する。

例）

```
http://…/api/bayesnet/nedofesta/model/current?type=json&format=true
```

レスポンス

例）

```
{
  "nodes" : {
    "b:001:図書館" : [ "0", "1" ],
    "b:004:ファーストフード店" : [ "0", "1" ],
    "e: あなたの年齢を教えてください。: 40 代" : [ "0", "1" ],
    "b:009:ラーメン屋" : [ "0", "1" ],
    "b:002:運動施設" : [ "0", "1" ],
    "b:003:居酒屋" : [ "0", "1" ],
    ＜省略＞
    "e: 5.新しいことが好きで、変わった考えを持つと思う: ややそう思う" : [ "0", "1" ],
    "e: 3.しっかりしていて、自分に厳しい人間だと思う: ややそう思う" : [ "0", "1" ]
  },
  "links" : [ {
    "src" : "b:001:図書館",
    "dst" : "e: あなたの職業を選択してください。: 事務職"
  }, {
    "src" : "b:001:図書館",
    "dst" : "e: 2.人に気を遣う、やさしい人間だと思う: どちらともいえない"
  },
  ＜省略＞
  { "src" : "e: 4.心配性で、おろおろしやすいと思う: どちらともいえない",
    "dst" : "e: 5.新しいことが好きで、変わった考えを持つと思う: ややそう思う"
  }, {
    "src" : "e: 5.新しいことが好きで、変わった考えを持つと思う: ややそう思う",
    "dst" : "e: 3.しっかりしていて、自分に厳しい人間だと思う: ややそう思う"
  } ]
}
```

4.4. 指定モデルの指定バージョンを取得

指定モデルの子弟のバージョンを取得する。

リクエスト

```
GET /api/bayesnet/:modelName/model/version/:version
```

例）

```
http://…/api/bayesnet/nedofesta/model/version/191209_193444
```

レスポンス

「4.3 指定モデルを取得（最新）」と同じ。

4.5. 確率推論（GET）

指定モデルの最新バージョンで確率推論を実行する。

リクエスト

```
GET /api/bayesnet/:modelName/infer
```

エビデンスはクエリパラメータで、「ノード名＝状態名」の形式で引き渡します。

例）

```
http://…/api/bayesnet/nedofesta/infer?b:001:図書館=1&b:008:ショッピング施設=1
```

レスポンス

例）

```
{
    "b:008:ショッピング施設":{
        "0":0,
        "1":1
    },
    "e: あなたの職業を選択してください。: 教職":{
        "0":0.9375,
        "1":0.0625
    },
    ＜省略＞
    "e: 3.しっかりしていて、自分に厳しい人間だと思う: 全くそう思わない":{
        "0":0.8495329711983738,
        "1":0.15046702880162616
    },
    "e: 1.活発で、人とよく関わると思う: 強くそう思う":{
        "0":0.8686552944264049,
        "1":0.13134470557359507
    }
}
```

4.6. 確率推論（POST）

リクエスト

指定モデルの最新バージョンで確率推論を実行する。

```
POST /api/bayesnet/:modelName/infer
```

エビデンスはリクエストボディで引き渡す。

例）

```
http://…/api/bayesnet/nedofesta/infer
```

（リクエストボディ）

```
{
  "evidence": {
    "e: あなたの職業を選択してください。: 教職": "1",
    "e: 1.活発で、人とよく関わると思う: 強くそう思う":"1"
  },
  "target": [
    "b:002:運動施設", "b:001:図書館"
  ]
}
```

※ キー target が無い場合は全変数の事後確率分布をレスポンスとして返す。

レスポンス

「4.5 確率推論（GET）」と同じ。

例）

```
{
    "b:001:図書館":{
        "0":0.5625,
        "1":0.4375
    },
    "b:002:運動施設":{
        "0":0.8125,
        "1":0.1875
    }
}
```

5. AI タッチラリーシステム

AI タッチラリーシステムは展示会などのイベント会場において、各展示ブースにアンケート用の端末を置き、来訪者がブースの訪問直後に端末に NFC カードをタッチ、簡単なアンケートに回答することで、ユーザーの回遊行動や展示ブースの評価などのデータを収集するシステムです。収集したデータを AI により解析することで仮説検証を繰り返し、より良い状態へのスパイラルアップを目的としています。

当システムは、POSEIDON（汎用アンケート API）を基盤として実装されています。また提供する機能は次の通りです。

管理者向け機能

- プロジェクト、スライド、ページ（アンケート）などをブラウザから GUI 操作で構築
- ブースの作成、ブースへのスライド（アンケート）の紐づけ、フロア画像上への配置
- レコメンド設定

タッチラリー機能

イベント会場でイベントに参加する人向けの機能で、管理者が作成した内容に従ってアンケート用のページを構成し、参加者に提示、回答を収集します。ブースの種類には受付、展示ブース、振返りブースなどがあります。

詳細は別紙の AI タッチラリーシステムの操作説明書をご参照ください。


タッチラリー実施の様子