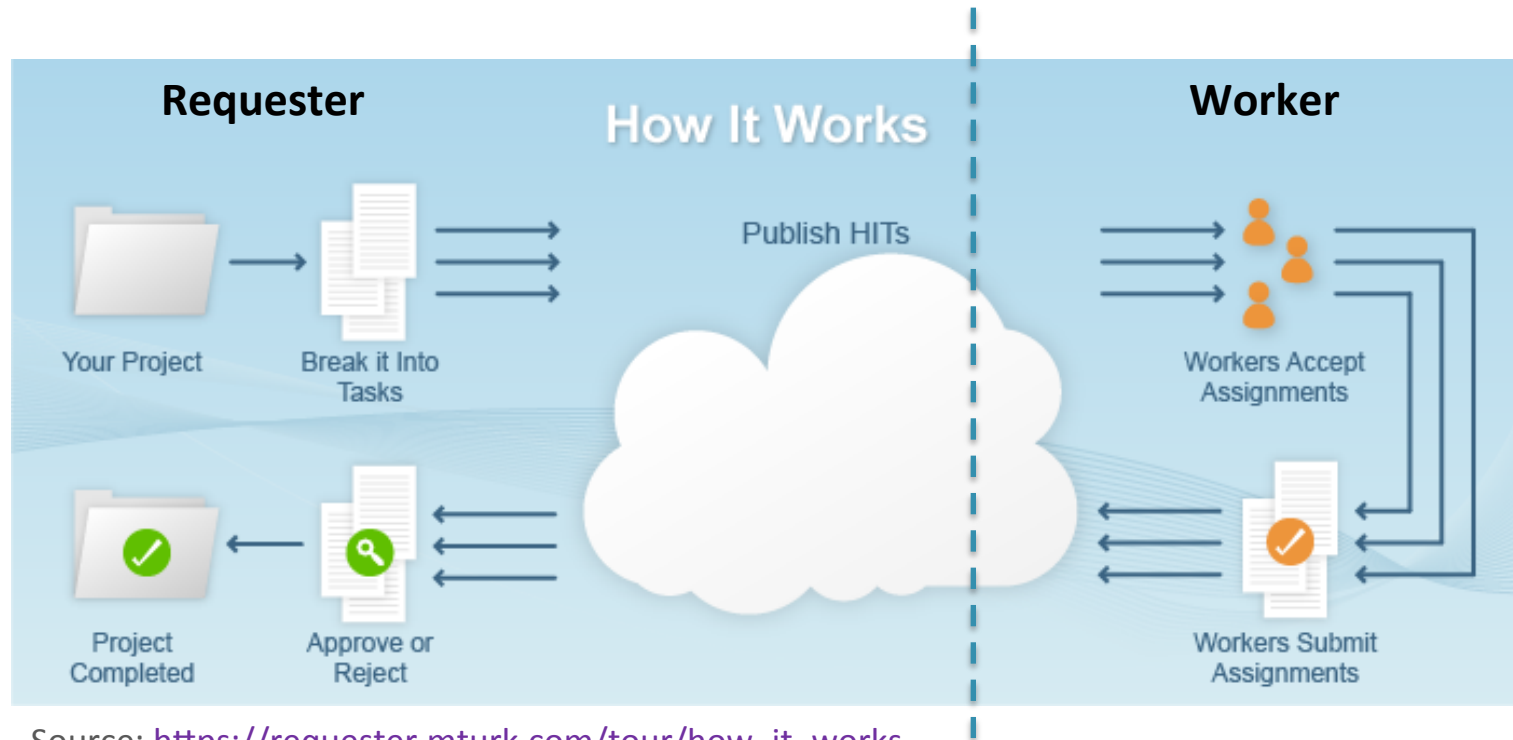# Amazon Mechanical Turk
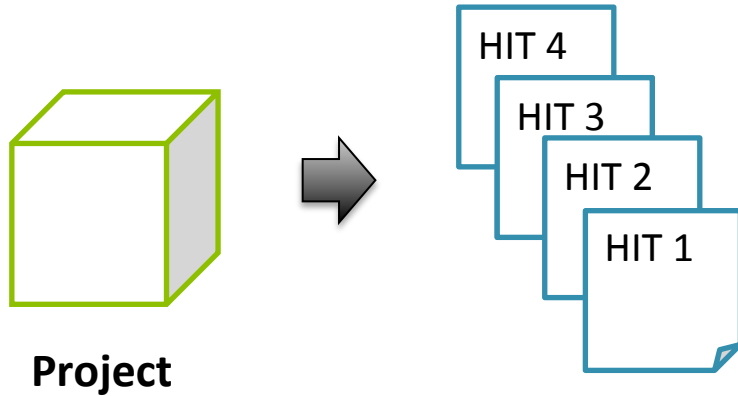# Hands-on session

Maribel Acosta

MTurk Basic Concepts (1)

Source: https://requester.mturk.com/tour/how_it_works

- **Requester:** creates and submits tasks to the platform.

- **Worker:** person who solves the tasks.

- **Human Intelligence Task (HIT):** work unit.

MTurk Basic Concepts (2)

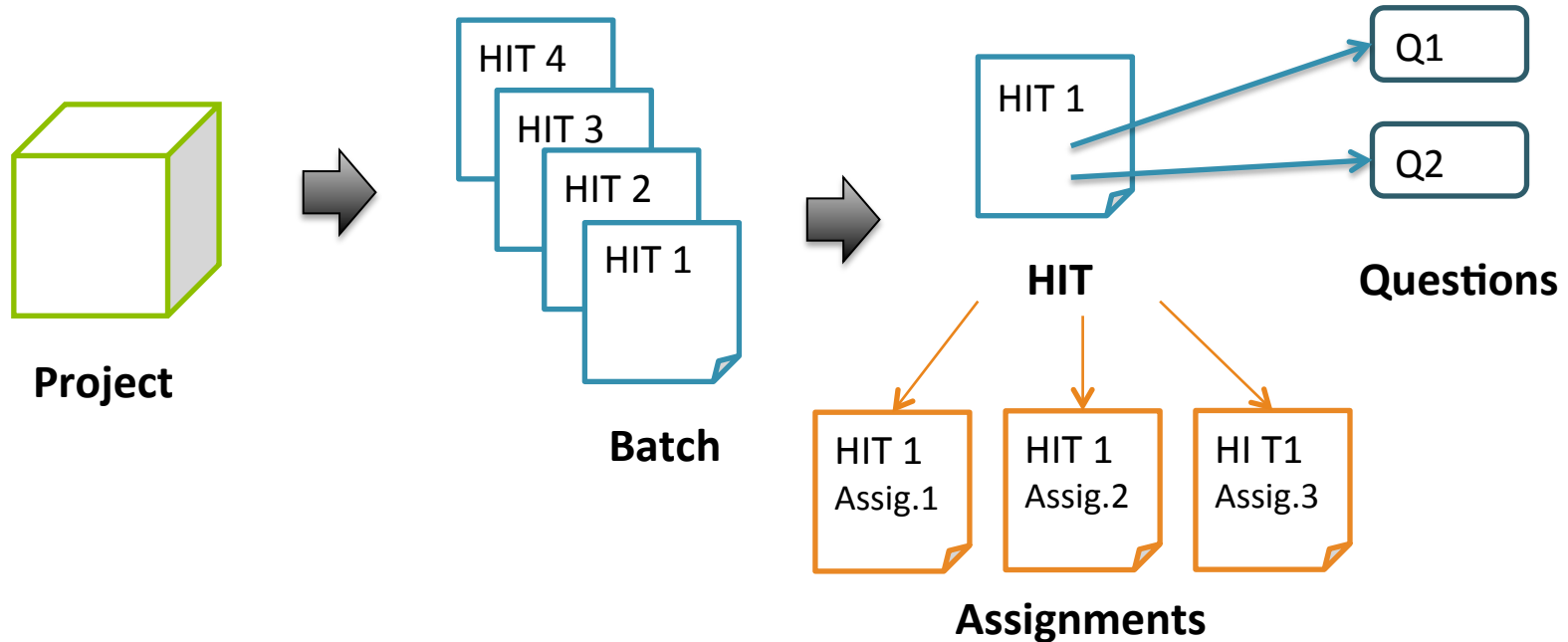**Project**

HIT 4
HIT 3
HIT 2
HIT 1

- **Project**: a graphical and functional template to create HITs.
  - The elements that stay the <u>same</u> in every HIT are denominated **template**.
  - The data that will <u>vary</u> from HIT to HIT are specified via **variables**.

- NOTE: If no variables are specified in the project, we will create a single HIT.

- **Variables:** allow creating several HITs in the project.

MTurk Basic Concepts (3)
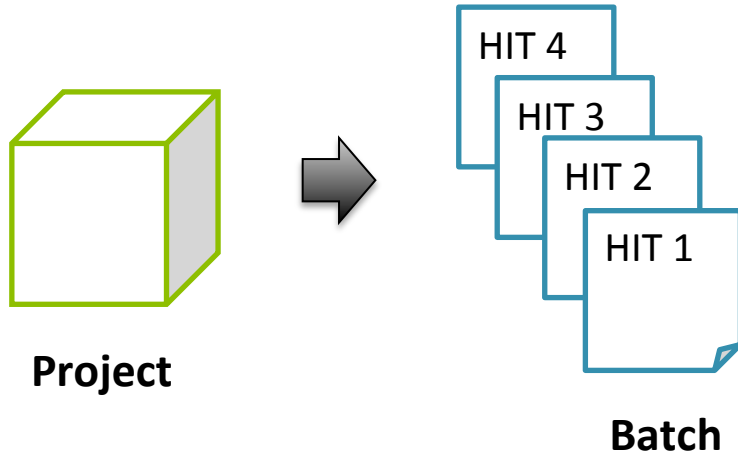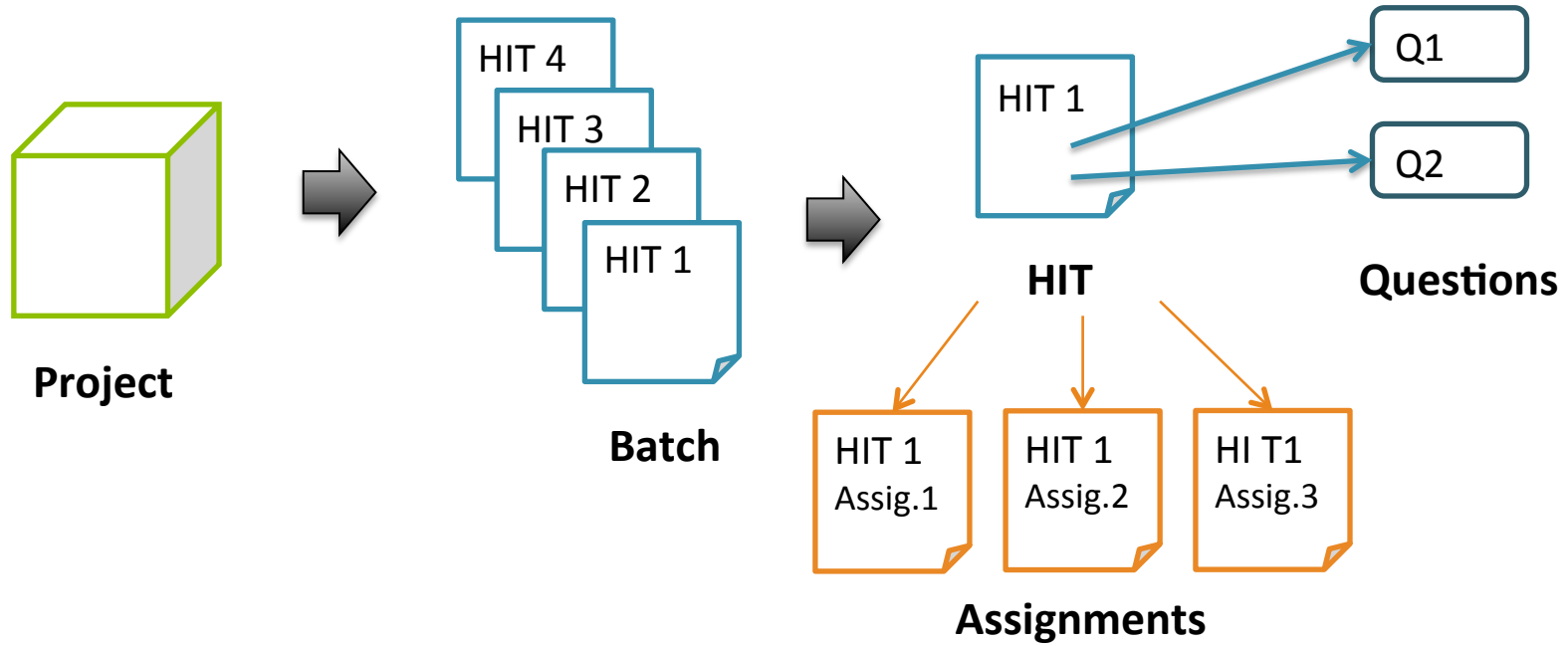

Project

Batch

- **Batch:** Group of HITs created by instantiating the variable(s) of a project.

- The values of the variables are specified in (CSV, TSV) files:
  - Each **column** corresponds to a variable
  - Each **row** is an instance -> HIT
  - Each file corresponds to a batch.
- We can create **several batches** for the same project.

MTurk Basic Concepts (4)

Project → **Batch** (HIT 4, HIT 3, HIT 2, HIT 1) → **HIT** (HIT 1) → **Questions** (Q1, Q2)

Assignments: HIT 1 Assig.1, HIT 1 Assig.2, HI T1 Assig.3

- **HIT:** Work unit. The same HIT can be solved by 1 or more workers (assignments).

- **Assignment:** How many workers should solve one exact same HIT.

- **Questions:** A single HIT may contain one or several questions.
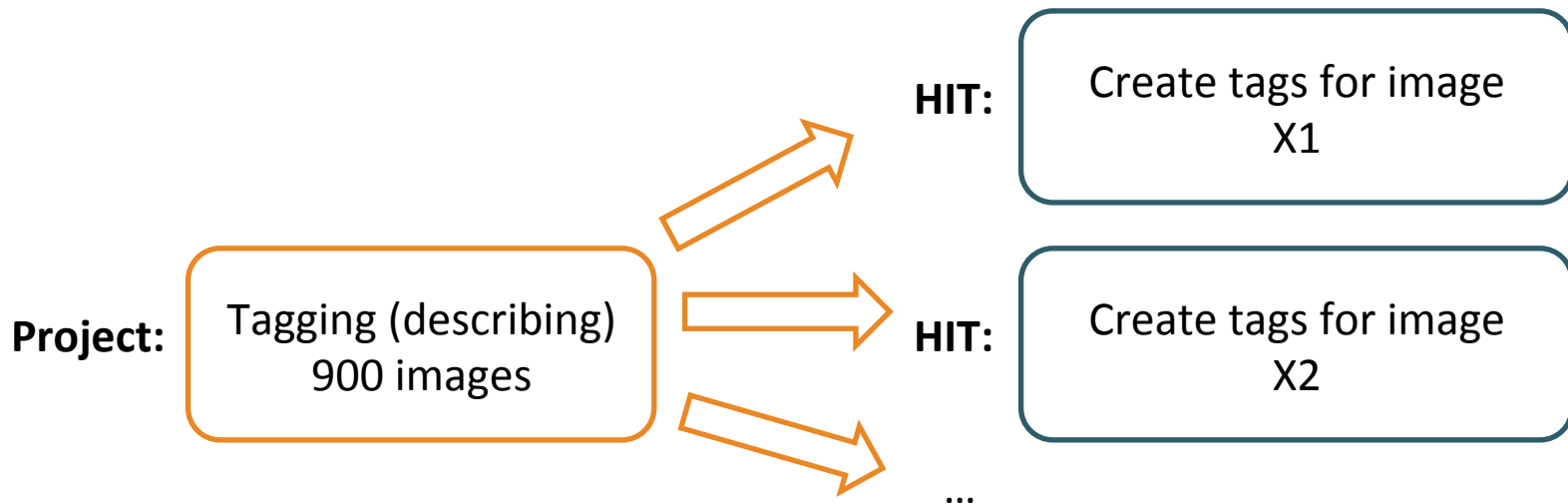
MTurk Basic Concepts (5)

Project → Batch

HIT 4
HIT 3
HIT 2
HIT 1

HIT 1 → Q1, Q2

Project

Batch

HIT

Questions

HIT 1
Assig.1

HIT 1
Assig.2

HI T1
Assig.3

Assignments

Total cost of the project = No. of HITs x No. of Assignments x (Reward per HIT + Fee)

MTurk Basic Concepts (6)

Example of Human Intelligence Tasks (HITs)

- Projects can be broken into smaller tasks called HITs

- A HIT represents a single work unit

Project: Tagging (describing) 900 images

HIT: Create tags for image X1

HIT: Create tags for image X2

…

No. of HITS = 900

MTurk Basic Concepts (7)

Example of Human Intelligence Tasks (HITs)

- Projects can be broken into smaller tasks called HITs

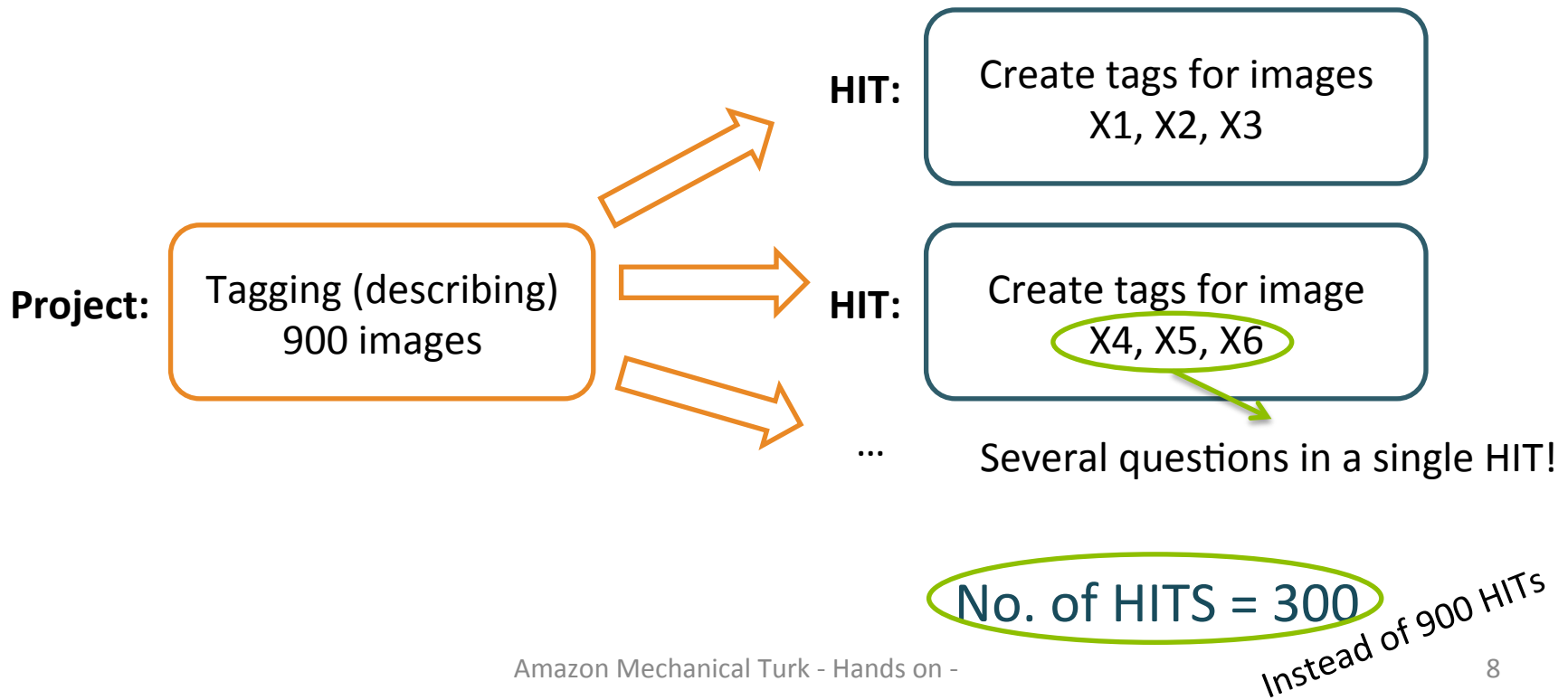- A HIT represents a single work unit

HIT: Create tags for images X1, X2, X3

Project: Tagging (describing) 900 images

HIT: Create tags for image X4, X5, X6

...

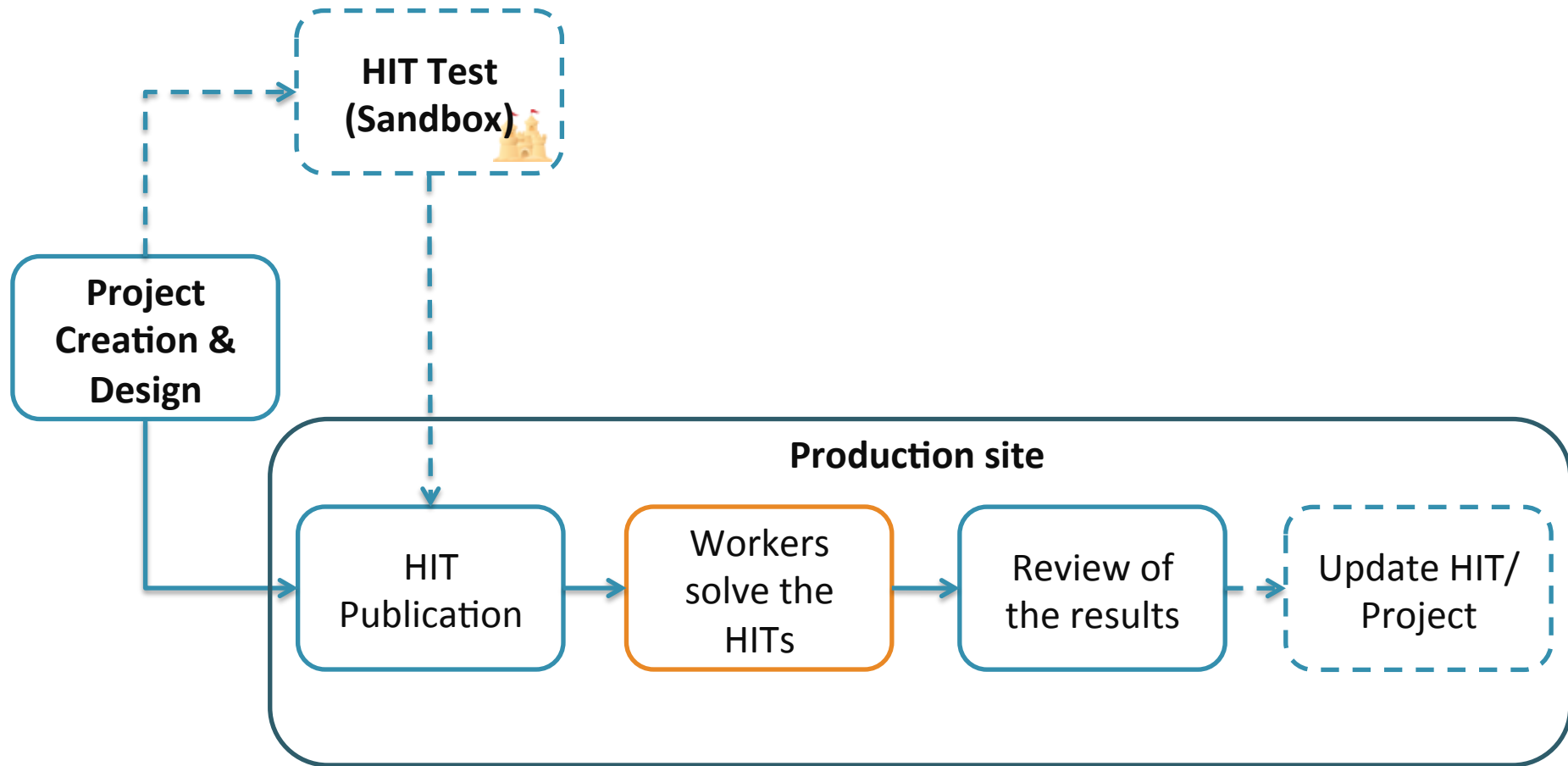Several questions in a single HIT!

No. of HITS = 300

Instead of 900 HITs

MTurk Basic Concepts (8)

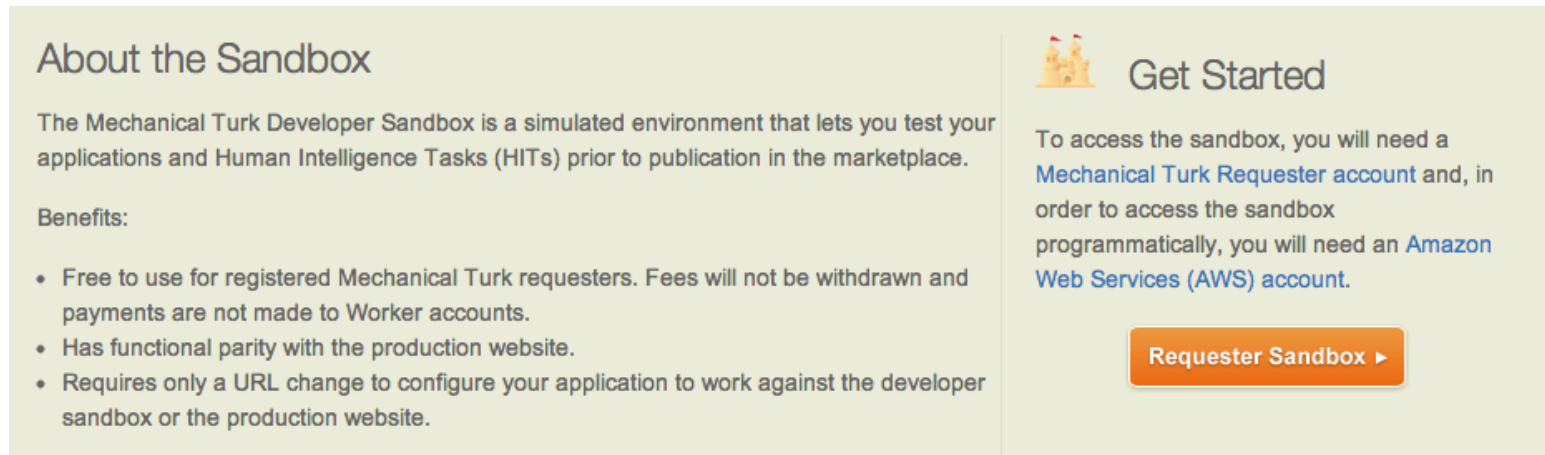When creating a project or individual HITs, the **HIT properties** must be specified:

- General information: includes the title and description of the HIT, as well as keywords which are used by worker for searching HITs.

- HIT duration time: time allotted to solve the HIT (before it is given to another worker).

- HIT life time: how long will the HIT be available on the platform.

- # Assignments: number of different persons that will perform the exact same HIT.

- Reward: payment for correctly solving each assignment.

MTurk Workflow for Requesters

HIT Test (Sandbox) 🏖️

Project Creation & Design

Production site

HIT Publication → Workers solve the HITs → Review of the results → Update HIT/ Project

MTurk Sandbox

The Sandbox is a simulated MTurk environment to test HITs.

About the Sandbox

The Mechanical Turk Developer Sandbox is a simulated environment that lets you test your applications and Human Intelligence Tasks (HITs) prior to publication in the marketplace.

Benefits:

- Free to use for registered Mechanical Turk requesters. Fees will not be withdrawn and payments are not made to Worker accounts.
- Has functional parity with the production website.
- Requires only a URL change to configure your application to work against the developer sandbox or the production website.

Get Started

To access the sandbox, you will need a Mechanical Turk Requester account and, in order to access the sandbox programmatically, you will need an Amazon Web Services (AWS) account.

Requester Sandbox ►

- Log in as **requester**: preview and test the interface of your HITs.
  - https://requestersandbox.mturk.com

- Log in as **worker**: solve your own HITs to test their functionalities and result output.
  - https://workersandbox.mturk.com

- **Best practice**: Always test your HITs before publishing them.

Managing HITs in MTurk

There are three different mechanism to manage your HITs in MTurk:

Web Interface

Command Line Tools

API

MTURK WEB INTERFACE 🌐

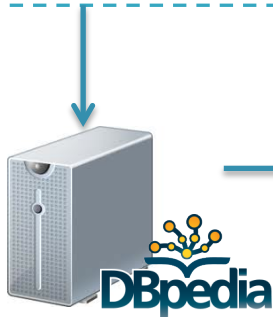# Hands On!

- **Project:** Crowdsourcing DBpedia triples to verify the links to external web pages

```
prefix dbpedia-ont:<http://dbpedia.org/ontology/>
prefix foaf:<http://xmlns.com/foaf/0.1/>
SELECT *
WHERE {
  ?s dbpedia-ont:wikiPageExternalLink ?o;        ──→  Triple to crowdsource
     foaf:name ?s_name;
     foaf:isPrimaryTopicOf ?s_wikipage .              Triples to build the UI
} LIMIT 200
```

MTurkDemo/data/sparql.csv

DBpedia

http://dbpedia.org/sparql

Hands On!

- Go to MTurk Sandbox as a **requester**:
  - https://requestersandbox.mturk.com/
- Click on **Sign In**
  - Email address
  - Password
- Now we are at "home"

1. Creating a Project

Different predefined templates: Select "other"

2. Setting up the HIT Properties (1)

HIT description

2. Setting up the HIT Properties (2)

Setting up your HIT

Reward per assignment

$ 0.04

Tip: Consider how long it will take a Worker to complete each task. A 30 second task that pays $0.05 is a $6.00 hourly wage.

Number of assignments per HIT

3

How many unique Workers do you want to work on each HIT?

Time allotted per assignment

10 | Minutes ⬍

Maximum time a Worker has to work on a single task. Be generous so that Workers are not rushed.

HIT expires in

15 | Days ⬍

Maximum time your HIT will be available to Workers on Mechanical Turk.

Results are automatically approved in

15 | Days ⬍

After this time, all unreviewed work is approved and Workers are paid.

HIT properties

Very **IMPORTANT**:
Set up quality mechanisms
Masters are selected by default

Advanced »

Save   Design Layout ▶

3. Selecting Qualifications

Worker requirements (filters)

Masters expect higher rewards
MTurk charges 20% for masters

4. Defining the Task

Edit Project

Use the HTML editor below to design the layout of your HIT. This layout is common for all of the HITs created with this project. You can define variables for data that will vary from HIT to HIT (Learn more).

① Enter Properties	② Design Layout	③ Preview and Finish

Project Name: DBpedia outlinks   This name is not displayed to Workers.

Frame Height   400   Height in pixels of the frame your HIT will be displayed in to Workers. Adjust the height appropriately to minimize scrolling for Workers.

Format ▾	Font ▾	U I B	A▾ Iₓ	≡ ≡ ≡ ≡	≟≡ ≔	↱ ↰	⟨⟩ Source

Instructions

Jump to questions

In this task, you will help us verify whether the links to external pages contained in Wikipedia articles are correct or not. The content of these external pages should be related to the content of the Wikipedia article. In this task, you will verify whether this is the case or not. In order to solve this task, we will provide the Wikipedia article and an external website that the article links to.

Your job: Compare whether the Wikipedia article and the external website are related. Try to **refresh the page** if the content is not displayed properly.

Example of incorrect data

In the following example, we are checking whether the external web pages are related to the Wikipedia article "John Two-Hawks".

body h2

WYSIWYG HTML editor

If we want to create HITs for several artists it is better to use **variables** ...

4. Defining the Task (with Variables)

Edit Project

Use the HTML editor below to design the layout of your HIT. This layout is common for all of the HITs created with this project. You can define variables for data that will vary from HIT to HIT (Learn more).

| ① Enter Properties | ② Design Layout | ③ Preview and Finish |

Project Name: DBpedia outlinks   This name is not displayed to Workers.

Frame Height   400   Height in pixels of the frame your HIT will be displayed in to Workers. Adjust the height appropriately to minimize scrolling for Workers.

Normal   ·   Font   ·   U I B A- I_x   ≡ ≡ ≡ ≡   ⅓≡ :≡   ↱ ↰   Source

Question

The content in "External page" corresponds to the topic covered in the Wikipedia article?

About:

${s_name}

Wikipedia article about: ${s_name}

Template: elements that stay the <u>same</u> in every HIT

Variables: data that will <u>vary</u> from HIT to HIT. Are denoted as follows: ${var_name}

5. Previewing the Template

Edit Project

This is how your HIT will look to Mechanical Turk Workers. Before you publish these HITs, any variables in the HIT will be replaced with the input data that you provide when you publish the HIT. You can download a sample of the input file for this project or learn more about acceptable file formats

Download sample

① **Enter Properties**   ② **Design Layout**   ③ **Preview and Finish**

This is what the workers will see

Project Name: DBpedia outlinks   This name is not displayed to Workers.

Comparing content between two web pages

Requester: mac21   Reward: $0.04 per HIT   HITs available: 0   Duration: 10 Minutes

Qualifications Required: HIT Approval Rate (%) greater than or equal to 95 , Number of HITs Approved greater than or equal to 1000

HIT Preview

Question

The content in "External page" corresponds to the topic covered in the Wikipedia article?

About:

$${s\_name}$$

The variables will be replaced by the input data

Wikipedia article about: ${s_name}

6. Creating Batches

7. Previewing the HITs

DBpedia outlinks

Comparing content between two web pages

Requester: mac21   **Reward:** $0.04 per HIT   **HITs available:** 200   **Duration:** 10 Minutes

Qualifications Required: HIT Approval Rate (%) greater than or equal to 95 ,

Number of HITs Approved greater than or equal to 1000

HIT Preview

Question

The content in "External page" corresponds to the topic covered in the Wikipedia article?

About:

!Action Pact!

Wikipedia article about: !Action Pact!

Create account   🔒 Log in

Article   Talk   Read   Edit   ▼   Search 🔍

!Action Pact!

Variables are replaced with the data from the input file

Showing HIT 1 of 200   **Next HIT**

8. Publishing the HITs

Batch Summary

Batch Name: DBpedia outlinks 5   **Description:** Compare the content between a Wikipedia article

Batch Properties

HITs	
Number of HITs in this batch:	200
Number of assignments per HIT:	x 3
Total number of assignments in this batch:	**600**

Cost		
Reward per Assignment:		$0.040
	x	600
Estimated Total Reward:		$24.000
Estimated Fees to Mechanical Turk:	+	$3.000
Estimated Total Cost:		**$27.000**
Your Available Balance:		$10,000.000
Your Projected Balance:		$9,973.000

Estimated Total Cost: $27.000   *(this is the amount that will be deducted from your Available Balance when you click "Publish HITs")*

Your Available Balance: $10,000.000   *(before clicking "Publish HITs")*
Your Projected Balance: $9,973.000   *(after clicking "Publish HITs")*

Summary of the project:
- # of HITs
- Rewards
- Total payment
- Account balance

9. Retrieving the Results

Review Results

Select the check boxes on the left to approve or reject results. You only pay for approved results. To evaluate results offline, select Download CSV.

For additional batch information, view batch details.

DBpedia outlinks 10

| Customize View | Filter Results | | | Upload CSV | Download CSV |

1 of 1 assignments (FILTER APPLIED: only show assignments that are in 'Submitted' status)

Approve	Reject

	HIT ID ▲	Worker ID	Lifetime Approval Rate	Input Data	Q1
☐	2N6Q1SNQQ7Q685ZCMH3NU12T1XJS8I	A3QBKUNGNMLHTF	98% (131/133)	BatchId:58757;	correct
☐	HIT ID ▲	Worker ID	Lifetime Approval Rate	Input Data	Q1

Approve	Reject

MTURK COMMAND LINE TOOLS

List of Commands

[HITs]

```
Make template
Load HITs
Delete HITs
Update HITs
Extend HITs
```

[Results]

```
Get result
Approve/reject work
Review results
Generate result summary
Grant bonus
Block/unblock worker
```

[Qualification types]

```
Create qualification type
Update qualification type
Get qualification results
Evaluate qualification request
Approve/reject qualification request
Revoke qualifications
Update qualification score
```

[Account]

```
Reset account
Get balance
```

Creating HITs with CLT

1. Configuring the connection to the platform

2. Setting up the HIT properties

3. Defining the task

4. Creating the batch

5. Publishing the HITs

6. Retrieving the results

1. Configuring the connection to the platform

- Configure the `./bin/mturk.properties` file

- Add the keys to access your MTurk account

```
# Information to access the MTurk account.
access_key=
secret_key=
```

- Specify the service to use (Sandbox or production site)

```
# If you want to use the Sandbox use the following service_url
service_url=https://mechanicalturk.sandbox.amazonaws.com/?Service=AWSMechanicalTurkRequester

# If you want to use the Sandbox use the following service_url
#service_url=https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
```

2. Setting up the HIT Properties

- Create a HIT properties file `myhits.properties`

```
#####################################
## Basic HIT Properties
#####################################

title: Information about artists.
description: Help us to select the most representative picture of an artist.
keywords:pictures, artists, comparison, images, famous people
reward: 0.04
assignments: 3
annotation:artist#pictures


#####################################
## HIT Timing Properties
#####################################

# this Assignment Duration value is calculated based on the number questions.
assignmentduration: 600

# this HIT Lifetime value is 60*60*24*15 = 15 days
hitlifetime:1296000

# this Auto Approval period is 60*60*24*15 = 15 days
autoapprovaldelay:1296000


#####################################
## Qualification Properties
#####################################

# Qualifications can be defined in the properties file instead of in code.
# You can add multiple qualifications for this HIT by simply increasing the # suffix.
# i.e. qualification.2: XXXXX
#       qualification.comparator.2:greaterthan

# this is a built-in qualification -- user must have an approval rate of 25% or greater
qualification.1:000000000000000000L0
qualification.comparator.1:greaterthan
qualification.value.1:25
qualification.private.1:false
```

3. Defining the Task (1)

- Create a template file `myhits.question`

Data structure →

Task description

```xml
<?xml version="1.0" encoding="UTF-8"?>
<QuestionForm xmlns="http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2005-10-01/QuestionForm.xsd">
  <Question>
    <QuestionIdentifier>1</QuestionIdentifier>
    <QuestionContent>
    <Text>Select the most representative picture for the following artist: ${artist}</Text>
    </QuestionContent>
    <AnswerSpecification>
      <SelectionAnswer>
        <MinSelectionCount>1</MinSelectionCount>
        <MaxSelectionCount>1</MaxSelectionCount>
        <StyleSuggestion>radiobutton</StyleSuggestion>
        <Selections>
          <Selection>
            <SelectionIdentifier>${image1}</SelectionIdentifier>
            <Binary>
              <MimeType><Type>image</Type><SubType>jpg</SubType></MimeType>
              <DataURL>${image1}</DataURL>
              <AltText>${image1}</AltText>
            </Binary>
          </Selection>
          <Selection>
          <SelectionIdentifier>${image2}</SelectionIdentifier>
            <Binary>
              <MimeType><Type>image</Type><SubType>jpg</SubType></MimeType>
              <DataURL>${image2}</DataURL>
              <AltText>${image2}</AltText>
            </Binary>
          </Selection>
          <SelectionIdentifier>noneoftheabove</SelectionIdentifier>
            <Text>None of the Above</Text>
          </Selection>
        </Selections>
      </SelectionAnswer>
    </AnswerSpecification>
  </Question>
</QuestionForm>
```

Type of selection →

Option 1

Type of data: image →

Option 2

3. Defining the Task (2)

Question data structures

[QuestionForm]

> Describes one or more questions for a HIT, or for a Qualification test.
>
> **Elements:** `QuestionIdentifier, DisplayName, IsRequired, QuestionContent, AnswerSpecification`

[ExternalQuestion]

> Displays a web page from your website in a frame.
>
> **Elements:** `ExternalURL, FrameHeight`

[HTMLQuestion]

> Describes the HIT with HTML code, without running an external website.
>
> **Elements:** `HTMLContent, FrameHeight`
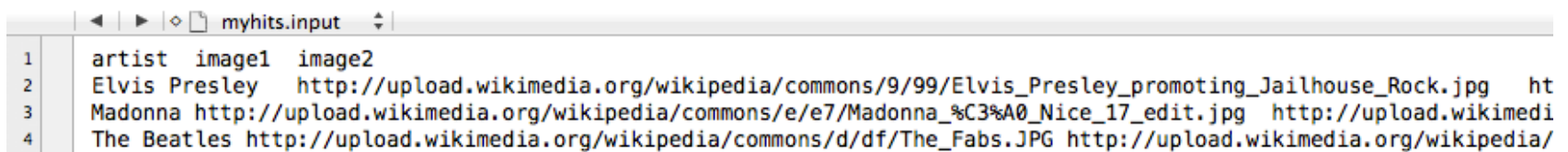
[HITLayout]

> Reusable template to provide HITs. It is created by creating a project on the website (MTutk Web Interface).

4. Creating a Batch

- Create an input file `myhits.input`:

  - Use **tab** as separator (very sensitive)

  - First row corresponds to the variables used in the task template. In this example: `${artist}`, `${image1}` and `${image2}`

```
        ◄ | ► | ◇  🗋  myhits.input    ◆ |
1    artist   image1   image2
2    Elvis Presley    http://upload.wikimedia.org/wikipedia/commons/9/99/Elvis_Presley_promoting_Jailhouse_Rock.jpg    ht
3    Madonna http://upload.wikimedia.org/wikipedia/commons/e/e7/Madonna_%C3%A0_Nice_17_edit.jpg  http://upload.wikimedi
4    The Beatles http://upload.wikimedia.org/wikipedia/commons/d/df/The_Fabs.JPG http://upload.wikimedia.org/wikipedia/
```

  - Each line contains the input data to generate the tasks

5. Publishing the HITs

- Run the `loadHits` command. In the example:

```
./loadHITs.sh -label ../samples/mturk_demo/myhits
               -input ../samples/mturk_demo/myhits.input
               -question ../samples/mturk_demo/myhits.question
               -properties ../samples/mturk_demo/myhits.properties
```

- The following output files are generated:

  – Success file: Contains identifiers to the HIT that were created. This information is used to retrieve the results of these HITs.

  – Failure file: Contains the rows of the input files that could not be published in the platform.

6. Retrieving and Reviewing the Results

- The results are retrieved with the `getResults` command. In the example:

```
./getResults.sh –successfile ../samples/mturk_demo/myhits.success
             -output ../samples/mturk_demo/myhits.results
```

Success file containing the identifiers of the HITs created with `loadHITs`

- The output file contains a column named "`reject`" which should be marked in case the assignment should be rejected.

- To approve/reject HITs execute the `reviewResults` command. In the example:

```
./reviewResults.sh –resultsfile ../samples/mturk_demo/myhits.results
```

MTURK API

Select a Programming Language

http://aws.amazon.com/code/SDKs/695

http://aws.amazon.com/code/SDKs/922

http://aws.amazon.com/code/SDKs/793

http://aws.amazon.com/code/SDKs/923

Java API:
1. Pre-requisites

- Download the MTurk SDK for Java

  - http://sourceforge.net/projects/mturksdk-java/files/latest/download

- Include all the `*.jar` files from the `/lib` and `/lib/build` in your classpath

- Set up the `mturk.properties` file

- Include the following classes:

  - `com.amazonaws.mturk.service.axis.RequesterService:` Establishes the connection to the platform

  - `com.amazonaws.mturk.util.ClientConfig:` Connection specifications (path to the mturk.properties files)

  - `com.amazonaws.mturk.service.exception.ServiceException:` Handles the exceptions when contacting the platform

  - `com.amazonaws.mturk.requester.HIT:` Class for HIT management.

Java API:
2. Setting up the connection

- Create a `RequestServiceObject`

- Specify the configurations with a `PropertiesClientConfig` object. This class has two different constructors with the following arguments:

    – No arguments: the configuration is set up using the methods `setAccessKeyId`, `setSecretAccessKey`, `setServiceURL`.

    – String specifying the path to the property file mturk.properties

```
service = new RequesterService(
            new PropertiesClientConfig("./mturk.properties"));
```

Java API:
3. Creating HITs (1)

There are different ways to create HITs using the Java API:

1. Reading certain information about the HITs from **CSV files** (properties, question, input) – same files used in the CLT example.

2. Specifying certain the information about the HITs using **Java objects**

    - This method is more suitable –specially for the input specification– when consuming data directly from other applications, for example, the results of a SPARQL query.

3. Combining 1. and 2.

Java API:
3. Creating HITs (2)

1. Reading from CSV Files

```java
// Locations of files containing HIT information.
String inputFile = "./src/site_category/site_category.input";
String propertiesFile =  "./src/site_category/site_category.properties";
String questionFile =  "./src/site_category/site_category.question";
...

// Reading the information from each file.
HITDataInput input = new HITDataCSVReader(inputFile);
HITProperties props = new HITProperties(propertiesFile);
HITQuestion question = new HITQuestion(questionFile);

// Specification of output files.
HITDataOutput success = new HITDataCSVWriter(inputFile + ".success");
HITDataOutput failure = new HITDataCSVWriter(inputFile + ".failure");

// The method service.createHITs publishes the HITs in the platform.
HIT[] hits = service.createHITs(input, props, question, success, failure);
```

Creates bulk of HITs

Java API:
3. Creating HITs (3)

2. Creating Java Objects

```
// Specification of a question (String). It could be any form of Mturk questions:
// QuestionForm, ExternalQuestion, HTMLQuestion or even a simple question
// created with RequesterService.getBasicFreeTextQuestion("question here")
String question = "<HTMLQuestion xmlns=...>...</HTMLQuestion>";

// Specification of the properties.
String title = "Information about artists.";
String description = "Help us to select the most representative picture of an artist.";
double reward = 0.04;
int assignments = 3;

// The method createHIT creates and publishes a single HIT using default values
// for the HIT properties not specified in the parameters.
// There are two createHIT methods with different arguments.
HIT hit = service.createHIT(title, description, reward, question, assignments);


        Create an individual HIT


// We need to keep track of success/failure while publishing the HIT.
// Print in success file: hit.getHITId() + "\t" + hit.getHITTypeId();
```

Java API:
3. Creating HITs (4)

3. Combining 1. and 2.

```java
// Specification of a question (String). It could be any form of Mturk questions:
// QuestionForm, ExternalQuestion, HTMLQuestion or even a simple question
// created with RequesterService.getBasicFreeTextQuestion("question here")
String question = "<HTMLQuestion xmlns=...>...</HTMLQuestion>";

// Reading the specification of the properties from a file.
String propertiesFile =  "./src/site_category/site_category.properties";
HITProperties props = new HITProperties(propertiesFile);

// The method createHIT creates and publishes a single HIT using default values
// for the HIT properties not specified in the parameters.
// There are two createHIT methods with different arguments.
HIT hit = service.createHIT(
        props.getTitle(),
        props.getDescription(),
        props.getRewardAmount(),
        question,
        props.getMaxAssignments());
...
```

Java API:
4. Retrieving the Results

```java
// Read from the success file the id of the HITs which were effectively published.
// Each row of the success file is as follows: HITid \t HITTypeID.
String successFile = "./src/site_category/site_category.success";
HITDataInput success = new HITDataCSVReader(successFile);

// Read each line of the success file to get the hitId.
for (int i = 1; i < success.getNumRows(); i++) {

    // Retrieve the information of submitted (completed) assignments for the HIT.
    Assignment[] a = service.getAllSubmittedAssignmentsForHIT(success.getRowValues(i)[0]);

    // Iterate over each assignment to get the answer.
    for (Assignment assignment : a) {
        // Print the results and the id of the assignment in the .results file.
        // The results of each assignment are stored in assignment.getAnswer();
        // The API offers methods to parse the XML answer.
    }
}
```

Java API:
5. Approving/Rejecting a Work

```java
// Read from the output file the assignements to accept/reject.
String resultsFile = "./src/site_category/site_category.results";
HITDataInput results = new HITDataCSVReader(resultsFile);

// Read each line of the results file.
for (int i = 1; i < results.getNumRows(); i++) {
    Map<String,String> row = results.getRowAsMap(i);
    String reject = (String)row.get("reject");

    // If the column reject is not marked -> accept assignment.
    // Otherwise -> reject assignment and submit a new (available) assignment.
    if (reject.equals("")) {
        service.approveAssignment(row.get("assignmentid"), "Thank you");
    } else {
        service.rejectAssignment(row.get("assignmentid"), "Sorry");
        HIT hit = service.getHIT(row.get("hitid"));
        service.extendHIT(row.get("hitid"), hit.getNumberOfAssignmentsPending()+1, (long)3600.00);
    }
}
```
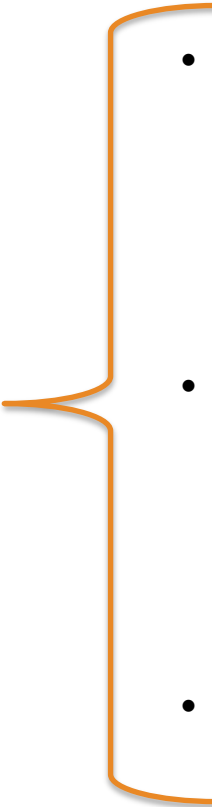
SUMMARY

Project/HIT Creation & Design (1)

- The requester is able to create **projects** or **individual HITs**

- Then, the **HIT properties** must be specified:

  - General information: includes the title and description of the HIT, as well as keywords which are used by worker for searching HITs.

  - HIT duration time: time allotted to solve the HIT (before it is given to another worker).

  - HIT life time: how long will the HIT be available on the platform.

  - # Assignments: number of different persons that will perform the same HIT.

  - Reward: payment for correctly solving each assignment.

Project/HIT Creation & Design (2)
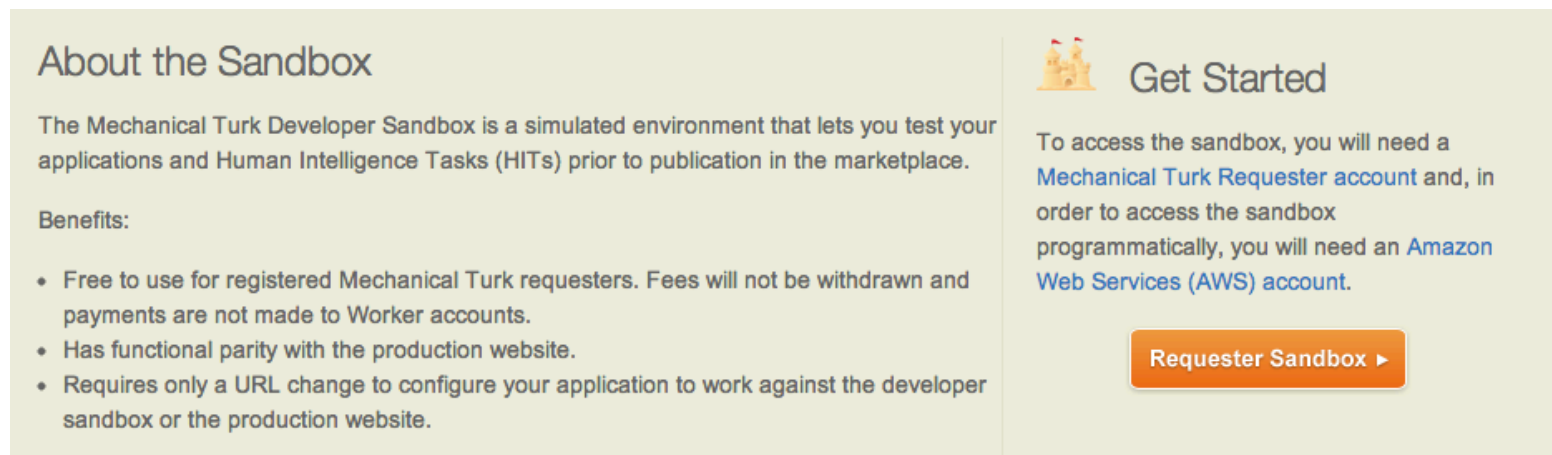
- Selection of **MTurk quality control** mechanisms:

Worker requirements

- **High quality workers**
  - Masters
  - Photo moderation masters
  - Categorization masters
    Masters expect higher rewards
    MTurk charges 20% for masters

- **System qualifications**
  - Location by country
  - HIT submission rate (%)
  - HIT approval/rejection rate (%)
  - (Absolute) Number of HITs approved

- **Qualification types**
  - Simply granted or attributed via customized tests

- These filters are automatically performed by the platform

HIT Test

- **Best practice**: Always test your HITs before publishing them
    1. Perform **technical tests** (both as requester and worker) in the MTurk Sandbox environment.

About the Sandbox

The Mechanical Turk Developer Sandbox is a simulated environment that lets you test your applications and Human Intelligence Tasks (HITs) prior to publication in the marketplace.

Benefits:

- Free to use for registered Mechanical Turk requesters. Fees will not be withdrawn and payments are not made to Worker accounts.
- Has functional parity with the production website.
- Requires only a URL change to configure your application to work against the developer sandbox or the production website.

Get Started

To access the sandbox, you will need a Mechanical Turk Requester account and, in order to access the sandbox programmatically, you will need an Amazon Web Services (AWS) account.

Requester Sandbox ▶

Source: https://requester.mturk.com/developer/sandbox

2. Publish a small subset of tasks in the production site to test **usability** and **responsiveness**.

Run live HITs

- **HIT publication:**
  - Make the HITs available to the workers

- **Review of the results:**
  - Monitor the submitted assignments constantly
  - Download the results
  - Accept/reject assignments
  - Block spammers (optional)

- **Update HIT/Project:**
  - Extend/expire HITs or modify other HIT properties
  - Add additional assignments

References

- AMT. Getting Started Guide. API Version 2012-03-25
  http://s3.amazonaws.com/awsdocs/MechTurk/latest/amt-gsg.pdf

- The Mechanical Turk Blog
  http://mechanicalturk.typepad.com/

- MTurk Java API
  http://people.csail.mit.edu/glittle/MTurkJavaAPI/

BACK-UP SLIDES

MTurk Platform Requirements

Create an AWS account

- Enables you to use Amazon Web Services
- May be linked to your personal Amazon account (optional)
- Credentials: access key and secret access key (20 and 40 chars)

Setup requester account

- Create the account at http://requester.mturk.amazon.com

Prepay for the HITs

- Before using MTurk, you must have money in your *balance*
- There are different mechanisms to perform the payments: credit card, debit card, or Amazon payment account.