universität innsbruck

Department of Computer Science

STI INNSBRUCK

# Schema.org Actions in the example of Easybooking

Thibault Gerrier

14.03.2018

# Outline

1. Easybooking
2. Objective
3. Easybooking Mapping
4. Schema.org Actions
5. Development
6. Conclusion

# Easybooking

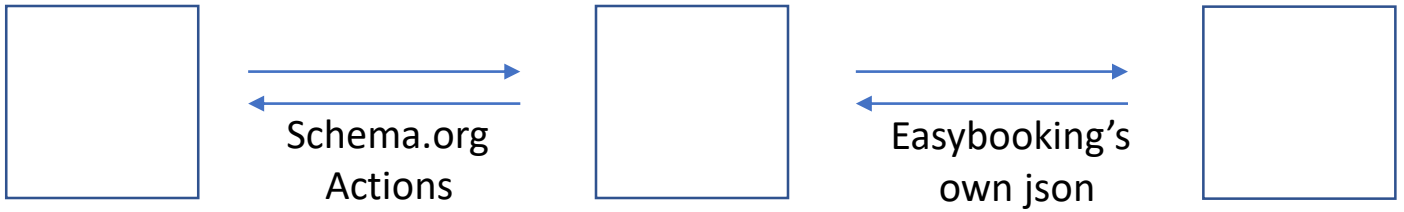
Booking engine used by Hotels

No public API

Workflow:

- Find suitable easybooking hotel (on your own, no special Actions)
- Choose:
  - Number of adults
  - Number of children (age)
  - Number of rooms
  - Arrival & departure dates
- Presented with rooms & prices
  - Choose room
  - Fill personal information

# Objective




Client
(Chatbot, Alexa, Google Home)

Server
(actions.semantify.it)

Easybooking API

Knows only
schema.org

Knows schema.org
and easybooking API

No idea what
schema.org is

# Easybooking Mapping

Required Input from the user:

- Number of adults -> schema.org/numAdults
- Number of children -> schema.org/numChildren
- Arrival date -> schema.org/checkinTime
- Departure date -> schema.org/checkoutTime

# Schema.org Actions (I)

Action Description:

```
{
 "@type": "SearchAction",
 "name": "Room availability for Hotel STI",
 "target": {
  "@type": "EntryPoint",
  "urlTemplate": "actions.semantify.it/api/easybooking/search/3896",
  "httpMethod": "POST",
  "encodingType": "application/ld+json",
  "contentType": "application/ld+json„
 },
 "query": {
  "@type": "LodgingReservation",
  "checkinTime-input": "required",
  "checkoutTime-input": "required",
  "numAdults-input": "required",
  "numChildren-input": "required"
 },
 "result": {
  "@type": ["Offer", "LodgingReservation"]
 }
}
```

Where to send the request with what HTTP Method

Client, who wants to use the action has to fill in all “-input” field that are required

What the client can expect as a result of calling the action

# Schema.org Actions (II)

Calling the Action Description:

POST https://actions.semantify.it/api/easybooking/search/3896

Client, filled in all input fields

Body:

```
{
 "@type": "SearchAction",
 "query": {
  "@type": "LodgingReservation",
  "checkinTime": "2018-03-16T00:00:00+00:00",
  "checkoutTime": "2018-03-19T00:00:00+00:00",
  "numAdults": "1",
  "numChildren": "0"
 }
}
```

Array of rooms.

Response:

```
{
 "@type": "SearchAction",
 "actionStatus": "CompletedActionStatus",
 "result": [Array],
}
```

# Schema.org Actions (III)

- Can go even further when response contains new potentialActions
- Eg. New Actions:

```
{
 "@type": "SearchAction",
 "actionStatus": "CompletedActionStatus",
 "result": [{
    "@type": "Offer",
    "potentialAction": {
       "@type": "BuyAction",
       "target": {…},
       "agent": {
           "@type": "Person",
           "givenName-input": "required",
        }
     }
 ]}
}
```

# Development

- actions.semantify.it server done in Node.js (JavaScript Runtime Environment)
- Node.js makes manipulation of JSON files (and JSON-LD) very easy, as JavaScript objects are JSON objects.
- Easy to use modules for making HTTP requests.

# Conclusion

- 2 parts:
  - Client:
    - Has no idea of the underlying APIs
    - Only knows schema.org
  - Server:
    - Maps clients request to requests how the API requires it
    - Knows schema.org and the API

Thank you for the attention
Questions?

www.uibk.ac.at/informatik
www.sti-innsbruck.at