Proposal: Actions in Schema.org (2013-05-11)

Editors: Yaar Schnitman (Google), Steve Macbeth (Microsoft)

Status: This document builds upon and supercedes earlier proposals from the schema.org team for describing Actions and Activities. See http://www.w3.org/wiki/WebSchemas/ActivityActions for earlier work. Discussion on the design is welcomed via the W3C WebSchemas public-vocabs@w3.org list. See W3C Wiki for details and archives.

Overview

This document proposes the introduction of "verbs" to the schema.org vocabulary, in the form of a new **Class** of **Things**, called **Actions**.

Problems Actions are supposed to solve:

1. **Identifying the semantic purpose of web forms:** User agents would like to understand the semantic meaning of a form in an HTML page. For example, know whether its a form that performs a search across the site, register a new account, or make an order.

2. **Actionable data:** Schema.org is used in various communication channels, such as web, email, notifications and social posts. In such contexts it is useful if the semantic items also describes interactive aspects of them selves. For example:
    a. An event could describe how to RSVP to the event.
    b. A movie could describe how to buy tickets for the movie.
    c. A website could describe how to register for the site.

3. **Activity confirmations and activity logs:** Once an action has been performed, there needs to be a way to record and describe it in schema.org. For example:
    a. A confirmation email would need to describe a completion of an action.
    b. An activity stream would need to describe the execution of an action.

4. **Delegation of Action Execution to other applications:** Typically publishers of data would also be in charge of executing actions on the data. However, we want to allow publishers to define actions, but let other parties execute actions. For example:
    a. A website that presents restaurant recommendations suggests a 3rd party tool

for booking restaurant tables.

The motivations for solving the problems above are:

- Improve search of both public web and personal private web history (browser history)
- Improve accessibility
- Improve interoperability between applications
- Help user agents to perform tasks on behalf of users

Executing Actions

Once actions are standardized in Schema.org, user-agents can standardize the way actions are presented to users and are executed.

User-agents can follow the following algorithm when trying to execute actions:

1. **Web Page:** If an action has a URL, user-agent can execute the action by opening the URL in a web browser and presenting it to the user. Alternatively, user-agent may open a dedicated installed app if it knows that such URLs are mapped to that app (via platform-specific deeplinking mechanisms).
2. **Web Form:** If the action is declared for a <form> element, user-agent can execute the action by presenting the webpage containing the form to the user, or possibly, executing the form on behalf of the user (assuming the user-agent has all the required information).
3. **Installed Application:** If the action has a handler of type ApplicationHandler, the user-agent should open the Application, and pass the action description to the application (exact format is platform-specific). See "Declaring an Installed App Handler" below for details.
4. **Inline Handler:** If the action has a handler of type InlineHandler, the user-agent may render a specialized input-capturing UI for that specific type of action, capture the user's input and send the data to the action publisher. The user-agent however needs some extra information in order to perform this. See "Declaring an Inline Handler" below for details.

Multiple Levels of Execution Specificity

This proposal described above allows for varying degrees of specificity of which actions apply to an item, and how to execute items.

1. An item can be declared with no actions at all. For example, a web page may contain a description of a Movie. Applications may register as handlers for such items. This is described here: Associating an Application with an Item Type or an Action Type.
2. An item can be declared with tentative actions but no explicit way to execute these actions. For example, a movie web page may declare that is available for sale, but not specify how it can be bought, leaving that up to user-agents to figure it out. See Declaring an Action for an Item.

3. An item can be declared with a tentative action as well as instructions of how to execute it. The instructions can be:
    a. A url of a web page where the action can be performed.
    b. A web form that can be used to perform the action.
    c. An installed application that can perform the action.
    d. An inline action handler that user agents can execute directly.

Definitions

Thing > Action

Base type for all schema.org Actoins. Whereas most of schema.org types represent "Nouns", an action represents a "Verb", making it possible to describe potential and concrete actions on schema.org types.

Property	Type	Description
status	ActionStatus	The status of the action.
handler	ActionHandler	Handler that can potentially execute the action.
performedBy	Thing	The performer of the action.
startTime	Datetime	When was the action performed (start time). An action with no startTime and endTime is typically in TENTATIVE or PENDING state.
endTime	Datetime	When was the action performed (end time). An action with no startTime and endTime is typically in TENTATIVE or PENDING state.
location	Location	Where the action is performed.

Thing > Enumeration > ActionStatus

Value	Description
TENTATIVE	The action is proposed.
PENDING	Action is in the process of being executed.
COMPLETED	Action has been executed.

CANCELLED	Action was executed but was cancelled.

Additions to Thing

Property	Type	Description
action	Action	Action that apply to the thing.

Action Sub-Types

SearchAction, LogInAction, … These are defined in Appendix A.Thing > ActionHandler
No properties

Thing > ActionHandler > ApplicationHandler

Used to declare how which installed application can execute the action.

Property	Type	Description
application	SoftwareApplicatio n	Information about the application that can perform the action.
url	URL	URL to invoke the application (deep link)

Additions to the existing Thing > SoftwareApplication

Property	Type	Description
handles	HandlerTarget	Action that apply to the thing.

Think > HandlerTarget

Uses in SoftwareApplication to declare type & action combinations that application can handle.

Property	Type	Description
targetType	Type	Schema.org types that the handler is applicable to.
actionType	Type	Schema.org action sub-types that hte handler is applicable to.

Thing > ActionHandler > InlineHandler

Used to declare how to perform inline action execution.

Property	Type	Description
method	HttpMethod	Enum Values: GET or POST.
url	URL	Where to send completed actions
encoding	Encoding	Enum Values: urlencoding or json-ld

Thing > ActionHandler > WebHandler
Declares a handler that is a web page where the action can be completed.

Property	Type	Description
url	URL	The web page where the action can be completed.

Thing > ActionHandler > WebFormHandler
Declares a handler that is a web form where the action can be completed.

Property	Type	Description
url	URL	The web form where the action can be completed.
formId	String	The id of the form in the web page.

Use Cases

Declaring the purpose of a webpage
Many webpages are not about a specific item (Movie, book, or article), but are functional pages, such as landing pages, search pages, registration forms, order forms, etc. For example, the following markup (in **blue**) can be added to any purchase order web page to describe the functional purpose of the page:

```
...
<div itemscope itemtype="http://schema.org/BuyAction">
   <meta itemprop="product" content="Diapers"/>
</div>
```

```
...
```

Notes:

1. The url of the webpage where the action is declared is the action's url property.
2. The "product" property of the BuyAction indicates indicates what is the item being bought. Each action type has its own properties that provide more semantic information about the action.
3. When stated in this form, the action's default state is TENTATIVE. The COMPLETED state may be used for web pages that are confirmations of performed actions.
4. This form of declaration is equivalent of declaring a http://schema.org/WebPage item with the action property as above.

Declaring the purpose of forms in a web page

The following markup (in **blue**) can be added to a <form> element to describe semantic meaning of the form:

```
...
<form action="http://mysite/search" method="get" itemscope
      itemtype="http://schema.org/SearchAction">
   <input itemprop="query" type="text" name="search"/>
   <input type="button" title="Search"/>
</form>
...
```

Notes:

1. The <form> is marked as an item of type SearchAction. SearchAction is defined in Appendix A.
2. The form's text input field is marked as the SearchAction's queryString property.
3. When stated in this form, the action's default state is TENTATIVE.

Declaring the semantic of a link in a web page

The following markup (in **blue**) can be added to a <a> element to describe the semantic meaning of the hyperlink:

```
...
<a href="http://mysite.com/login" itemscope
   itemtype="http://schema.org/LogInAction">
   <meta itemprop="service" content="mysite.com"/>
Log In</a>
...
```

Notes:

5. The url "http://mysite.com/login" is where the LogInAction can be performed.
6. The "service" property of the LogInAction indicates the service name that the user going to be logged-in into.
7. When stated in this form, the action's default state is TENTATIVE.
8. This form of declaration can be seen a shorthand for using an action handler of type WebHandler. See below.

Other supported ways to declare the same thing as above which are more verbose but may have :

```
<script type="application/json-ld">
{
  '@context': 'schema.org',
  '@type': 'http://schema.org/LogInAction',
  'service': 'mysite.com',
  'handler': {
     '@type': 'http://schema.org/WebHandler',
     'url': 'http://mysite.com/login',
  }
}
</script>
```

which is also equivalent to:

```
<script type="application/json-ld">
{
  '@context': 'schema.org',
  '@type': 'http://schema.org/LogInAction',
  'service': 'mysite.com',
  'url': 'http://mysite.com/login',
}
</script>
```

Declaring an Action for an Item

The following markup (in **blue**) can be added to existing microdata elements to specify actions:

```
...
<div itemscope itemtype="http://schema.org/SportEvent">
   <span itemprop="name">Miami Heat at Philadelipa 76ers - Game 3</span>
   <meta itemprop="startDate" content="2016-04-21T20:00">
   <a itemprop="action" itemscope
      itemtype="http://schema.org/BuyAction"
      href="http://mysite/buyTickets?event=123">
   Buy Ticket</a>
</div>
...
```

Notes:

1. A "BuyAction" is specified as an "action" property of a "SportEvent".
2. The url "`http://mysite/buyTickets?event=123`" is where the action can be performed.
9. When stated in this form, the action's default state is TENTATIVE.
10. The "name" property of the action is infered from the text of the hyperlink "BuyTicket".

Recording an Action

The following markup (using json-ld) can be used to record an action that has been performed. Such records are useful in confirmation web-pages, an activity streams, or in confirmation email.

```
<script type="application/json-ld">
{
  '@context': 'schema.org',
  '@type': 'http://schema.org/BuyAction',
  'startTime': '2013-04-02T12:31-0800',
  'endTime': '2013-04-02T12:31-0800',
  'performer': [ Person or Organization ],
  'status': 'COMPLETED',
  'bought': [ Product or Service],
}
</script>
```

Notes:

1. The BuyAction has the status 'COMPLETED'. If not specified, this status can be inferred from the existance of startTime & endTime.

Declaring an Installed App Handler

An action can refer the user to a specific installed application that may be used to execute the action. This is done by specifying an explicit hander of type ApplicationHandler on the action. The following example shows how a website that describes the movie "Skyfall" may add a Buy Tickets action that is fulfilled by the "Fandango for Android" application:

```
<script type="application/json-ld">
{
  '@context': 'schema.org',
  '@type': 'http://schema.org/Movie',
  'name': 'Skyfall',
  'url': 'http://skyfallmovie.com/',
  ... information about the movie ...
  'action': {
    '@type:': 'http://schema.org/ViewAction',
    'handler:' {
      '@type': 'http://schema.org/ApplicationHandler',
      'application': {
        '@type': 'http://schema.org/SoftwareApplication',
        'name': 'Netflix.com',
        'platform': 'Android',
        'platformVersion': '2.1',
        'appId': '123456abcdef',
      },
      'url': 'netflix://play?movie=skyfall',
    },
  }
}
</script>
```

Notes:

1. The User-Agent should delegate the execution of the action to an installed App specified in the action.handler.application property.
2. action.handler.application property can alternatively contain the name or URL of a webpage containing the application specification.
3. action.url is the deeplink (=command line argument) that is passed to the invoked application.
4. More than a single handler can be specified.
5. More than a single handler.application can be specified.
6. User agents may also infer the application based on the deeplink in action.handler.url.

Associating an Application with a Type or an Action Type

Application developers may want to tell the OS or App Store which schema.org Types and Action Types they can handle. For example, the Netflix Android Application can be used to perform the **ViewAction** and **RentAction** on **Movies**. We suggest the following markup to describe an application and the types+action types it can handle.

```
<script type="application/json-ld">
{
  '@context': 'schema.org',
  '@type': 'http://schema.org/SoftwareApplication',
  'name': 'Netflix for Android',
  'platform': 'Android',
  'platformVersion': '2.1',
  'appId': '123456abcdef',   // Platform-specific unique application id
  'handles': {
     '@type': 'http://schema.org/HandlerTarget',
     'targetType': 'http://schema.org/Movie',
     'actionType': {
        'http://schema.org/ViewAction',
        'http://schema.org/RentAction'
     }
  },
}
</script>
```

Notes:

1. Netflix may not support viewing or renting ANY movie, but only movies that are in in its catalogue. User Agents should therefore present to Netflix the movie that the user is interested in viewing or renting, or access Netflix's catalogue somehow, in order to make a decision whether to suggest the Netflix App to the user as a way to execute the View action.
2. This markup can be placed in the application's download page (so web crawlers can find it) and/or in its manifest.

Declaring an Inline Action Handler

As actions are standardizing by schema.org, user-agents may provide standardized handling for some actions. That however requires some extra help from the publisher of the action when data needs to be sent back from the publisher.

A good example is Event RSVP, which is a pretty common action. Event is already defined in schema.org and we are introducing EventRsvpAction, with "answer" (=Yes/No/Maybe), "comment" and "bringingOthers" properties. A publisher may embed the following in a post or an email to suggest to user agents how to complete the action:

```
<script type="application/json-ld">
{
  '@context': 'schema.org',
  '@type': 'http://schema.org/Event',
  'name': 'John's Dinner Party',
  'url': 'http://events.com/event?id=123',
  'startDate': '2013-04-03T19:00-0800',
  ... more information about the event ...
  'action': {
    '@type:': 'http://schema.org/EventRsvpAction',
    'handler:' {
      '@type': 'http://schema.org/InlineHandler',
      'method': 'post',
      'encoding': 'uri-encoded',
      'url': 'http://events.com/rsvp?event=123',
      'requiredProperties': ['answer'],
      'optionalProperties': ['comment', 'bringingOthers'],
    },
  }
}
</script>
```

Notes:

1. action.handler.requiredProperties indicates which properties need to be completed by the user agent (done by asking the user for the information). Note that these properties are defined on http://schema.org/EventRsvpAction, so the user agent knows the semantics of each property.
2. action.handler.optionalProperties indicates which properties can be completed by the user agent, but are not mandatory.
3. action.handler.method, encoding and url indicate where and how to send the completed action information.

Relationship with Intent Systems

Typical intent systems, like Android Intents or Web Intent consist of 2 main components:

1. Intents: Data objects that express a tentative operation that needs to be performed.
2. Intent Handlers: Components that can execute intents.

This proposal is designed to match these concepts. Actions map to Intents. Action Handlers map to Intent Handlers. User-Agent may therefore bridge platform specific intent systems with the schema.org actions.

For example, a ViewAction for an item of type Image, can be automatically mapped by the user agent to a platform intent for viewing images. A platform application that registers itself as intent handled for viewing images can be used by the user-agent as an ActionHandler for image ViewAction.

Custom Actions

This proposal includes adding a vocabulary of pre-defined actions to schema.org. Actions, like any schema.org type, have semantic meanings, and therefore it makes sense to standardize.

However, its expected that various parties will want to extend the actions vocabulary. Schema.org has an extensions mechanism (see http://schema.org/docs/extension.html) which can be used to introduce custom actions.

For example, say that Netflix want to define a special verb for adding movies to their queue. They could name the action:

```
http://schema.org/Action/NetflixAddToQueueAction
```

Since the action is not part of the schema.org vocabulary, user-agents might not make sense of it. However, they will know it is an action. Netflix might help user-agents make sense of it, by populating common properties such as "name", "url" and even basic handlers, which will make the action executable. For more enhanced user experience, Netflix and its collaborators may create installed applications associated with this action type.

Furthermore, instead of extending Action, Netflix may extend a more specific action sub-type, such as http://schema.org/RentAction:

```
http://schema.org/RentAction/NetflixAddToQueueAction
```

This will give user-agents even more understanding what the action is, which might help them index, render and execute it better.

Appendix - Preliminary Action Sub-Type Definitions

Following is a list of Action sub-types. Each action corresponds with a "verb", and has properties corresponding with the qualifiers of the action.

This is an initial list that covers all examples in this document. Over time, Schema.org will extend this list with more actions corresponding with more verbs.

In addition, schema.org may also standardize more specialized actions: For example, in addition to Thing > Action > BuyAction, schema.org might introduce more specific actions, such as:

- Thing > Action > BuyAction > BuyTicketAction
- Thing > Action > BuyAction > BuyTicketAction > BuyMovieTicketAction.

Note that such actions can be declared using solely with BuyAction. The intention of introducing more specialized actions in the vocabulary is to make markup simpler. For example, instead of:

```
{
  "@type": "BuyAction",
  "product": {
    "@type": "Ticket",   // Note that Ticket is not in schema.org vocabulary.
    "ticketFor": {
      "@type": "Movie",
      … details about the movie ...
    }
   "seat": "14F",
    ... other ticket details ...
  }
}
```

publishers would declare:

```
{
  "@type": "BuyMovieTicketAction",
  "movie": { … movie details …" }
   "seat": "14F",
  … ticket details ...
}
```

Thing > Action > SearchAction

Property	**Type**	**Description**
query	Text	The search query string

Thing > Action > EventRsvpAction

Property	Type	Description
event	Event	The event being RSVP-ed
guest	Person	The person being invited
answer	EventRsvpAnswer	Values: Yes/No/Maybe
comment	Comment	An RSVP comment
bringingOthers	Number	How many other guests the guest is bringing with her
bringingKids	Number	How many kids the guest is bringing with her

Thing > Action > BuyAction

Property	Type	Description
product	Product or Service	The product that is being bought
order	Order	The order details

Thing > Action > LogInAction

Property	Type	Description
service	Text or URL	The service being logged out from

Thing > Action > ViewAction

Property	Type	Description
item	Thing	The item that is being viewed

Thing > Action > RentAction

Property	**Type**	**Description**
item	Thing	The item that is being rented