

SEARCH ENGINE USING WEB ANNOTATIONS

A PROJECT REPORT

Submitted By:

ANUBHA SRIVASTAVA- 161B038

KARTIKAE MISHRA-161B107

RITIK VISHWAKARMA-161B179

Under the Guidance of: Mr. Kunj Bihari Meena



November - 2019

Submitted in partial fulfillment of the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Department of Computer Science & Engineering

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY,

A-B ROAD, RAGHOGARH, DT. GUNA - 473226, M.P., INDIA

Declaration by the Student

I hereby declare that the work reported in the B. Tech. project entitled as “...**SEARCH ENGINE USING WEB ANNOTATIONS**..., in partial fulfillment for the award of degree of BACHELOR OF TECHNOLOGY submitted at Jaypee University of Engineering and Technology, Guna, as per best of my knowledge and belief there is no infringement of intellectual property right and copyright. In case of any violation I will solely be responsible.

Anubha Srivastava(161B038)

Kartikae Mishra(161B107)

Ritik Vishwakarma(161B179)

Department of Computer Science and Engineering

Jaypee University of Engineering and Technology

Guna, M.P., India

Date:

CERTIFICATE



JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY,
A-B ROAD, RAGHOGARH, DT. GUNA - 473226, M.P., INDIA

This is to certify that the work titled “**SEARCH ENGINE USING WEB ANNOTATION**” submitted by “ **Anubha Srivastava, Kartikae Mishra, Ritik Vishwakarma**” in partial fulfillment for the award of degree of B.Tech. of Jaypee University of Engineering & Technology, Guna has been carried out under my supervision. As per best of my knowledge and belief there is no infringement of intellectual property right and copyright. Also, this work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma. In case of any violation concern student will solely be responsible.

Signature of Supervisor

Mr.Kunj Bihari Meena

Designation

Date

ACKNOWLEDGEMENT

It is our privilege to express sincerest regards to our project supervisor Mr. Kunj Bihari Meena for there valuable input able guidance, encouragement, whole hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our head of department, Prof. Shishir Kumar for encouraging and allowing us to present the project on the topic Search Engine Using Web Annotations at our department premises for the partial fullfilment of the requirement leading to the award of B-Tech Degree .

We take this opportunity to thank all the faculty members who have directly or indirectly helped our project. We pay our respect and love to our parents and all other family members and friends for there love and encouragement throughout our career last but not the least we express our thanks to our friends for there cooperating and support.

Signature of Student

Anubha Srivastava(161B038)

Kartikae Mishra(161B107)

Ritik Vishwakarma(161B179)

Date : 4/12/2019

EXECUTIVE SUMMARY

The main objective of this project was to develop and design a Search Engine using web Annotations platform which focus on Searching Web pages. It primarily operates on web application development area, particularly in Massive Open Online Courses (MOOCs) are free online courses available for anyone to enroll. This covers the process of developing one such system. Search Engine Using Web Annotation was developed for U.G/P.G pursuing students & people who want to learn. The system acts as a centralized control panel and enables the user to manage learning & coding together. This will help their skill to meet requirements for professional projects & increase their knowledge & value in market. A couple different ways. Some, like Yahoo, are gargantuan directories where every listing has been meticulously placed within its multi-level directory structure. Others, like AltaVista, are giant indexing machines. They send programs (often called "spiders" or "robots") out through the Internet to add web sites to their indexes. The indexing type of Search Engines generally work one of two ways: they index all the words (or maybe only the first 250 words) on one or more pages of a web site. they grab behind-the-scenes information stored in "meta tags" that are near the top of each web page (if the webmaster put them there). All the course content is being collected through different sources like Udemy, Coursera, Lynda from their own consent that is with permission and data will be fetched directly through API's. Companies have been increasingly looking for employees who have industry leading skill set. This system provides sophisticated learning & handling of all types of coding with a wide range of courses. The growth of Online Educational Platform during the last few decades was on boom, the demand for online education solutions to teach different type of courses increases in different sector. Skills are becoming a key element for many roles in the industry.

LIST OF FIGURES

Fig 1.1: Annotation

Fig 1.2: Entity relationship diagram

Fig 1.3: Data flow diagram

Fig 2.1: Application Architecture

LIST OF TABLES

Table 3.1: Annotations

Table 3.2: External web resources

Table 3.3: Classes

Table 3.4: Segment of external resources

Table 3.5: Choice between bodies

Table 4.1: Specific resources

Table 4.2: Purpose of external web resources

Table 4.3: Selectors

Table 4.4: Text qoute

Table 4.5: SVG selectors

Table 4.6: Refinement of selection

Table 4.7: Request header state

Table 4.8: Refinement of state

Table 6.1: Correspondence among media type and selectors

Table 7.1: Proposed model

Table of Contents

1. Chapter 1 Introduction	1
1.1 Aims of the Model	2
1.2 Serialization of the Model	3
1.3 Conformance	3
1.3.1 Conformance Requirements Related to Selectors	3
1.4 Terminology	4
2. Chapter 2 Web Annotation Principles	6
3. Chapter 3 Web Annotation Framework	8
3.1 Annotations	8
3.2 Bodies and Targets	9
3.2.1 External Web Resources	10
3.2.2 Classes	12
3.2.3 Segments of External Resources	13
3.2.4 Embedded Textual Body	15
3.2.5 String Body	16
3.2.6 Cardinality of Bodies and Targets	18
3.2.7 Choice Between Bodies	19
3.3 Other Properties	20
3.3.1 Lifecycle Information	21
3.3.2 Agents	22
3.3.3 Intended Audience	23
3.3.4 Accessibility of Content	24
3.3.5 Motivation and Purpose	25
3.3.6 Rights Information	26

3.3.7 Other Identities	27
4. Chapter 4 Specific Resources	30
4.1 Purpose for External Web Resources	31
4.2 Selectors	32
4.2.1 Fragment Selector	33
4.2.2 CSS Selector	34
4.2.3 XPath Selector	35
4.2.4 Text Quote Selector	36
4.2.5 Text Position Selector	38
4.2.6 Data Position Selector	39
4.2.7 SVG Selector	40
4.2.8 Range Selector	42
4.2.9 Refinement of Selection	44
4.3 States	45
4.3.1 Time State	46
4.3.2 Request Header State	47
4.3.3 Refinement of State	49
4.4 Styles	50
4.5 Rendering Software	53
4.6 Scope of a Resource	54
5. Chapter 5 Collections	55
5.1 Annotation Collection	55
5.2 Annotation Page	56
6. Chapter 6 Correspondence Among Media Types and Selectors	59

7. Sets of Bodies and Targets	60
8. Conclusion	63
9. Appendix	64
8. References	65

Chapter-1

1. Introduction

Annotating, the act of creating associations between distinct pieces of information, is a pervasive activity online in many guises. Web citizens make comments about online resources using either tools built in to the hosting website, external web services, or the functionality of an annotation client. Comments about shared photos or videos, reviews of products, or even social network mentions of web resources could all be considered as annotations. In addition, there are a plethora of "sticky note" systems and stand-alone multimedia annotation systems. This specification describes a common approach to expressing these annotations, and more.

The Web Annotation Data Model provides an extensible, interoperable framework for expressing annotations such that they can easily be shared between platforms, with sufficient richness of expression to satisfy complex requirements while remaining simple enough to also allow for the most common use cases, such as attaching a piece of text to a single web resource.

An annotation is considered to be a set of connected resources, typically including a body and target, and conveys that the body is related to the target. The exact nature of this relationship changes according to the intention of the annotation, but the body is most frequently somehow "about" the target. This perspective results in a basic model with three parts, depicted below. The full model supports additional functionality, enabling content to be embedded within the annotation, selecting arbitrary segments of resources, choosing the appropriate representation of a resource and providing styling hints to help clients render the annotation appropriately. Annotations created by or intended for machines are also possible, ensuring that the Data Web is not ignored in favor of only considering the human-oriented Document Web.

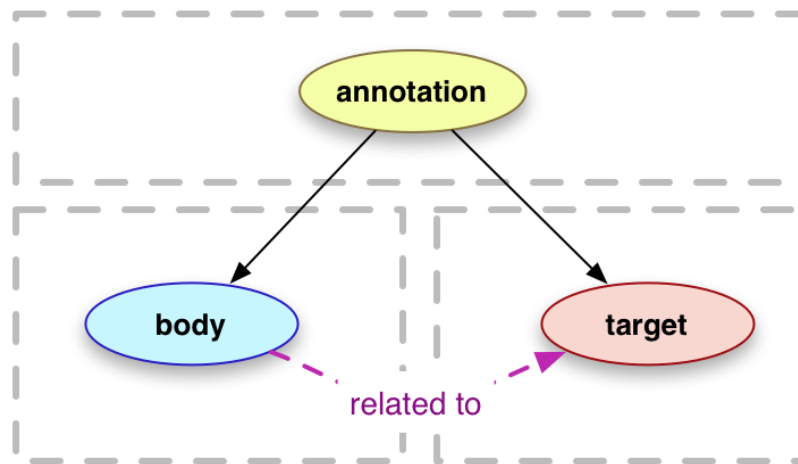


Fig-1.1 Annotations

The Web Annotation Data Model does not prescribe a transport protocol for creating, managing and retrieving annotations. Instead it describes a resource oriented structure and serialization of that structure that could be carried over many different protocols. The related [\[annotation-protocol\]](#) specification describes a recommended transport layer, which may be adopted separately.

1.1 Aims of the Model

The primary aim of the Web Annotation Data Model is to provide a standard description model and format to enable annotations to be shared between systems. This interoperability may be either for sharing with others, or the migration of private annotations between devices or platforms. The shared annotations must be able to be integrated into existing collections and reused without loss of significant information. The model should cover as many annotation use cases as possible, while keeping the simple annotations easy and expanding from that baseline to make complex uses possible.

The Web Annotation Data Model is a single, consistent model that can be used by all interested parties. All efforts have been made to keep the implementation costs for both producers and consumers to a minimum. A single method of fulfilling a use case is strongly preferred over multiple methods, unless there are existing standards that need to be accommodated or there is a significant cost associated with using only a single method. While the Data Model is built using Linked Data fundamentals, the design is intended to allow rich and performant non-graph-based implementations. As such, inferencing and other graph-based queries are explicitly not a priority for optimization in the design of the model.

1.2 Serialization of the Model

The examples throughout the document are serialized as [JSON-LD] using the Context given in Appendix A of the Annotation Vocabulary [annotation-vocab], which is the preferred serialization format. The media type of this format is defined in Section 3 of the Annotation Protocol [annotation-protocol] .

When the only information that is recorded in the annotation is the IRI of a resource, then that IRI is used as the value of the relationship, as in Example 1. When there is more information about the resource, the IRI is the value of the `id` property of the object which is the value of the relationship, as in Example 2.



1.3 Conformance

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

1.3.1 Conformance Requirements Related to Selectors

Not all [Selectors](#) are relevant for all media types; some combinations are meaningless or not formally defined. An implementation may therefore ignore certain types of Selectors in case the corresponding media types are not handled by that particular implementation.

The table in A. [Correspondence Among Media Types and Selectors](#) shows the correspondence among the main media types addressed in this specification and Selector types. The meaning of the table elements, and their effect on implementation conformance, is as follows.

- A “” sign in the table means that the Selector type is relevant for that particular media types. A conforming implementation **must** implement that particular combination *if* it handles the corresponding media type.
- A “” sign in the table means that the Selector type is not relevant for that particular media types. Conforming implementations **should** ignore that particular combination.
- A “?” means that it is not possible to specify, in general, whether that particular combination is meaningful (e.g., fragment identifiers are defined for specific media types, i.e., specific text media types other than plain text may have fragments defined but they are not listed in the table). In other cases the usefulness of such combination is not clear (e.g., Data Position

selectors for binary images). Conforming implementations **may** implement that particular combination.

1.4 Terminology

IRI

An **IRI**, or Internationalized Resource Identifier, is an extension to the URI specification to allow characters from Unicode, whereas URIs must be made up of a subset of ASCII characters. There is a mapping algorithm for translating between IRIs and the equivalent encoded URI form. IRIs are defined by [rfc3987].

Resource

An item of interest that **may** be identified by an **IRI**.

Web Resource

A **Resource** that **must** be identified by an **IRI**, as described in the Web Architecture [webarch]. Web Resources **may** be dereferencable via their IRI.

External Web Resource

A **Web Resource** which is not part of the representation of the Annotation, such as a web page, image, or video. External Web Resources are dereferencable from their **IRI**.

Property

A feature of a **Resource**, that often has a particular data type. In the model sections, the term "Property" is used to refer to only those features which are *not* **Relationships** and instead have a literal value such as a string, integer or date. The valid values for a Property are thus any data type other than object, or an array containing members of that data type if more than one is allowed.

Relationship

In the model sections, the term "Relationship" is used to distinguish those features that refer to other **Resources**, either by reference to the **Resource's IRI** or by including a description of the **Resource** in the Annotation's representation. The valid values for a Relationship are:

ER Diagram

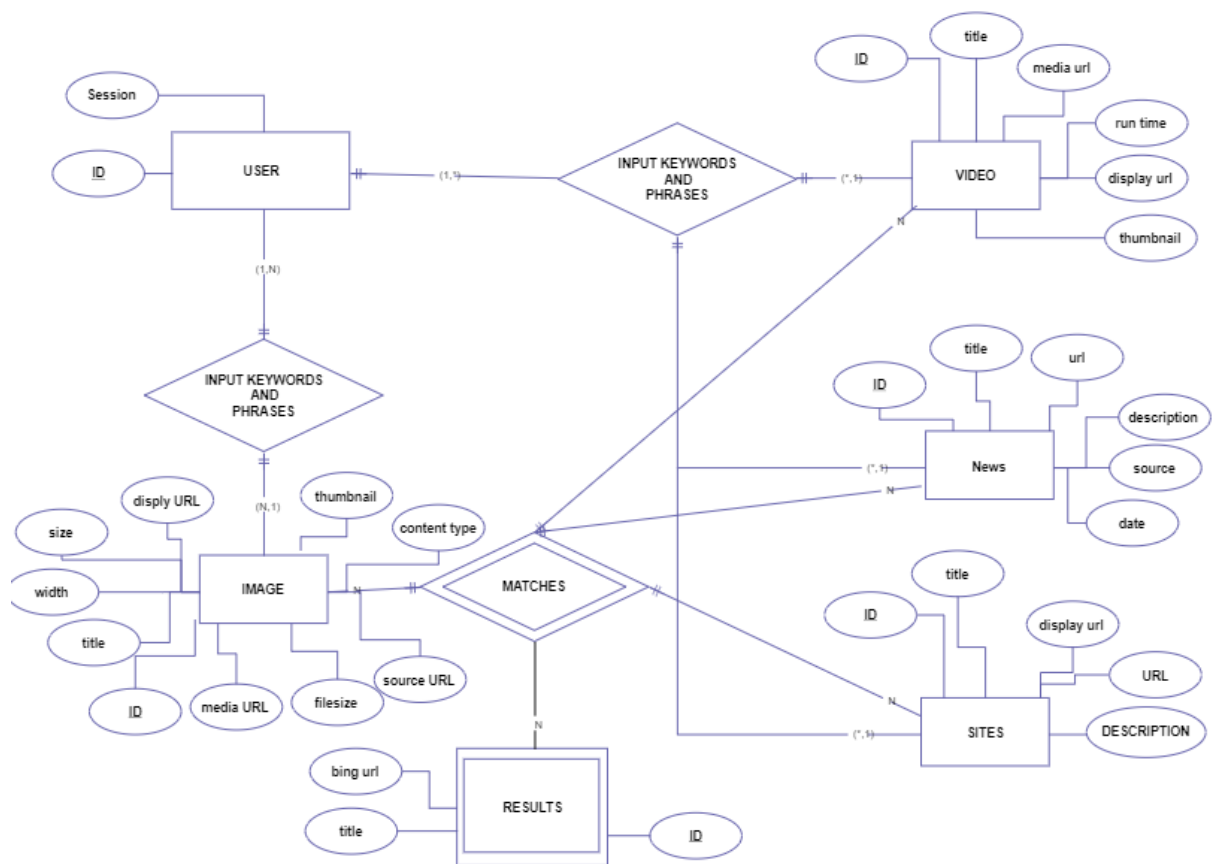


Fig 1.2 ER Diagram

Data Flow Diagram

Data Flow Diagram

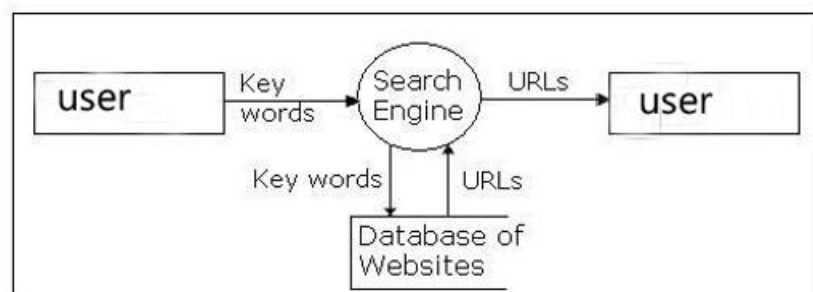
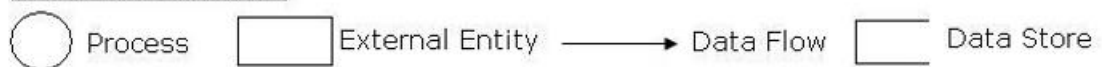


Fig-1.3 DFD

Chapter-2

Web Annotation Principles

The Web Annotation Data Model is defined using the following basic principles:

- An Annotation is a rooted, directed graph that represents a relationship between resources.
- There are two primary types of resource that participate in this relationship, Bodies and Targets.
- Annotations have 0 or more Bodies.
- Annotations have 1 or more Targets.
- The content of the Body resources is related to, and typically "about", the content of the Target resources.
- Annotations, Bodies and Targets may have their own properties and relationships, typically including creation and descriptive information.
- The intent behind the creation of an Annotation or the inclusion of a particular Body or Target is an important property and represented by a Motivation resource.

The following principles describe additional distinctions regarding the exact nature of Target and Body:

- The Target or Body resource may be more specific than the entity identified by the resource's [IRI](#) alone.
- In particular, The Target or Body resource may be a specific segment of the resource.
- The Target or Body resource may be styled in a specific way.
- The Target or Body resource may be a specific state of the resource.
- The Target or Body resource may be included in the Annotation to play a specific role.
- The Target or Body resource may be any combination of the above.
- The resource with these constraints is a separate resource from the Annotation, Body or Target, and is called a `SpecificResource`.
- The `SpecificResource` refers to the source resource and the constraints that make it more specific.
- The identity of the `SpecificResource` is separate from the descriptions of the constraints.

- The Body resource may be a choice between multiple resources.

Application Architecture

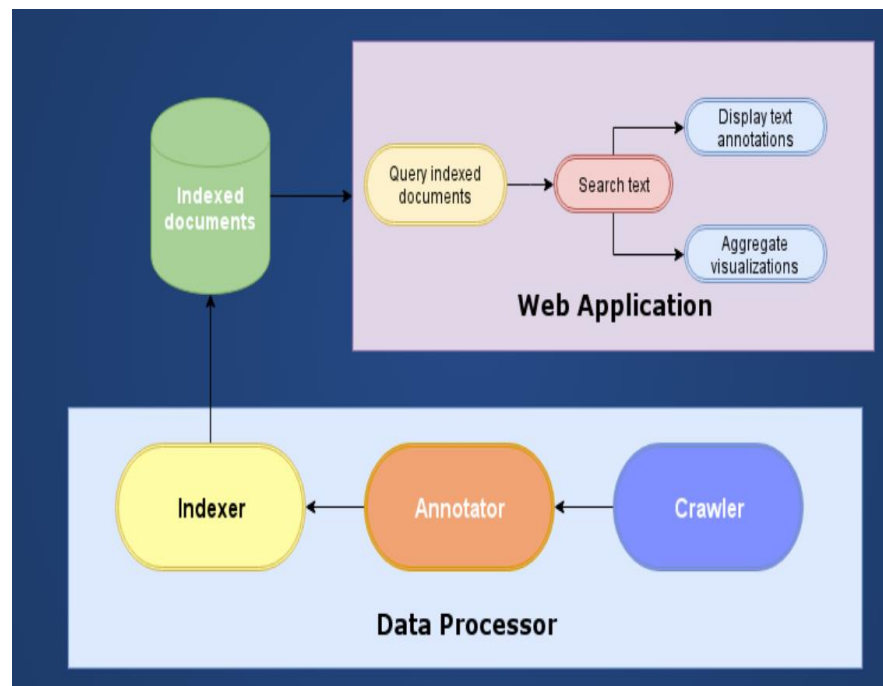


Fig-2.1 Application Architecture

Chapter-3

Web Annotation Framework

3.1 Annotations

An Annotation is a Web Resource. Typically, an Annotation has a single Body, which is a comment or other descriptive resource, and a single Target that the Body is somehow "about". The Annotation likely also has additional descriptive properties.

Term	Type	Description
@context	Property	The context that determines the meaning of the JSON as an Annotation. The Annotation must have 1 or more @context values and http://www.w3.org/ns/anno.jsonld must be one of them. If there is only one value, then it must be provided as a string.
Id	Property	The identity of the Annotation. An Annotation must have exactly 1 IRI that identifies it.
Type	Relationship	The type of the Annotation. An Annotation must have 1 or more types, and the Annotation class must be one of them.
Annotation	Class	The class for Web Annotations. The Annotation class must be associated with an Annotation using type .
Body	Relationship	The relationship between an Annotation and its Body. There should be 1 or more body relationships associated with an Annotation but there may be 0.
target	Relationship	The relationship between an Annotation and its Target. There must be 1 or more target relationships associated with an Annotation.

Table-3.1

Example Use Case: Alice has written a post that makes a comment about a particular web page. Her client creates an Annotation with the post as the body resource, and the web page as the target resource.

Example

EXAMPLE 1: Basic Annotation Model

```
{  
  
  "@context": "http://www.w3.org/ns/anno.jsonld",  
  
  "id": "http://example.org/anno1",  
  
  "type": "Annotation",  
  
  "body": "http://example.org/post1",  
  
  "target": "http://example.com/page1"  
}
```

3.2 Bodies and Targets

The Web is distributed, with different systems working together to provide access to content. Annotations can be used to link those resources together, being referenced as the Body and Target. The Target resource is always an External Web Resource, but the Body may also be embedded within the Annotation. External Web Resources may be separately dereferenced to retrieve a representation of their state, whereas the embedded Body does not need to be dereferenced as the representation is included within the Annotation's representation.

3.2.1 External Web Resources

Term	Type	Description
id	Property	The IRI that identifies the Body or Target resource. Bodies or Targets which are External Web Resources must have exactly 1 id with the value of the resource's IRI .
format	Property	The format of the Web Resource's content. The Body or Target should have exactly 1 format associated with it, but may have 0 or more. The value of the property should be the media-type of the format, following the [rfc6838] specification.
language	Property	The language of the Web Resource's content. The Body or Target should have exactly 1 language associated with it, but may have 0 or more, for example if the language cannot be identified or the resource contains a mix of languages. The value of the property should be a language code following the [bcp47] specification.
processingLanguage	Property	The language to use for text processing algorithms such as line breaking, hyphenation, which font to use, and similar. Each Body and Target may have exactly 1 processingLanguage . The value of the property should be a language code following the [bcp47] specification. If this property is not present and the language property is present with a single value, then the client should use that language for processing requirements.
ltr	Instance	The direction that indicates the value of the resource is explicitly directionally isolated left-to-right text.

Table-3.2

Web Resources are identified with a [IRI](#) and have various properties, often including a format or language for the resource's content. This information may be recorded as part of the Annotation, even if the representation of the resource must be retrieved from the Web.

Example Use Case: Beatrice records a long analysis of a patent, and publishes the audio on her website as an mp3. She then creates an Annotation with the mp3 as the body, and the PDF of the patent as the target.

Example

EXAMPLE 2: External Web Resources

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno2",
  "type": "Annotation",
  "body": {
    "id": "http://example.org/analysis1.mp3",
    "format": "audio/mpeg",
    "language": "fr"
  },
  "target": {
    "id": "http://example.gov/patent1.pdf",
    "format": "application/pdf",
    "language": ["en", "ar"],
    "textDirection": "ltr",
    "processingLanguage": "en"
  }
}
```

3.2.2 Classes

It is useful for clients to know the general type of a Web Resource in advance. If the client cannot render videos, then knowing that the Body is a video will allow it to avoid needlessly downloading a potentially large content stream. For resources that do not have obvious media types, such as many data formats, it is also useful for a client to know that a resource with the format `text/csv` should not simply be rendered as plain text, despite the first part of the media type, whereas `application/pdf` may be able to be rendered by the user agent despite the main type being 'application'.

Example Use Case: Corina shoots a video of her comment about a website on her phone and uploads it. She associates the video with the website via an Annotation, and her client adds types as a hint to consuming systems.

Model

Term	Type	Description
Type	Relationship	The type of the Body or Target resource. The Body or Target may have 1 or more types , and if so, the value should be drawn from the list of classes below, but may come from other vocabularies.
Dataset	Class	The class for a resource which encodes data in a defined structure.
Image	Class	The class for image resources, primarily intended to be seen.
Video	Class	The class for video resources, with or without audio.
Sound	Class	The class for a resource primarily intended to be heard.
Text	Class	The class for a resource primarily intended to be read.

Table-3.3

Example

EXAMPLE 3: Typing of Body and Target

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno3",
  "type": "Annotation",
  "body": {
    "id": "http://example.org/video1",
    "type": "Video"
  },
  "target": {
    "id": "http://example.org/website1",
    "type": "Text"
  }
}
```

3.2.3 Segments of External Resources

Many Annotations involve part of an External Web Resource, rather than its entirety. In the Web [webarch], segments of resources are identified using IRIs with a fragment component that at the same time both describes how to extract the segment of interest from the resource, and identifies the extracted content. For simple Annotations, it is valuable to be able to use these IRIs with a fragment component as the identifier for either Body or Target.

Example Use Case: Dawn wants to describe a particular region of an image. She highlights that area in her client and types in the description. Her client then constructs an IRI with an appropriate fragment component as the target.

It is important to be aware of the consequences of using an IRI with a fragment component, and the restrictions that using them places on implementations.

- Fragments are defined with respect to individual media types. For example, HTML has a specific set of semantics regarding the meaning of the fragment part of the IRI.

- Not every media type has a fragment specification. For example, Office documents might have a media-type and be published on the web, but not have semantics associated with the fragment part of the IRI.
- Even if a media type does have a fragment definition, it is often not possible to describe the segment of interest sufficiently precisely. For example, fragments for HTML cannot be used to describe an arbitrary range of text.
- It is not possible to determine with certainty what is being identified without knowing the media type, as the same fragment string might be possible in different specifications. For example, the same fragment string could identify either a rectangular area in an image, or a strangely named section of an HTML document.
- IRIs with a fragment component are not compatible with other methods of describing the segment more specifically. For example, it is not possible to describe how to retrieve the correct representation, add style information, or associate a role with the resource, using such IRIs. The method to accomplish these requirements is described in the [Fragment Selector](#) portion of the [Specific Resources](#) section.
- As IRIs are considered to be opaque strings, annotation systems may not discover annotations with fragment components when searching by means of the IRI without the fragment. For example, an Annotation with the Target <http://example.com/image.jpg#xywh=1,1,1,1> would not be discovered in a simple search for <http://example.com/image.jpg>, even though it is part of it.

For more information regarding the use of IRIs with fragment components, please see the Best Practices for Fragment Identifiers and Media Type Definitions [[fragid-best-practices](#)].

Example

EXAMPLE 4: IRIs with Fragment Components

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno4",
  "type": "Annotation",
  "body": "http://example.org/description1",
  "target": {
    "id": "http://example.com/image1#xywh=100,100,300,300",
```

```

    "type": "Image",

    "format": "image/jpeg"

  }
}

```

Term	Type	Description
Id	Property	The IRI that identifies the Textual Body. The Body may have exactly 1 IRI that identifies it.
Type	Relationship	The type of the Textual Body resource. The Body should have the TextualBody class, and may have other classes.
TextualBody	Class	A class assigned to the Body for embedding textual resources within the Annotation. The Body should have the TextualBody class.

Table-3.4

3.2.4 Embedded Textual Body

In many situations, the Body of the Annotation will be in a text format, and created at the same time as the Annotation without a separate IRI. In these cases, the Body's text can be included as part of the Annotation to avoid having to interact with multiple systems. The Body may also have the features of External Web Resources, including especially the language of the text and the format that it is conveyed in.

Example Use Case: Emily writes a comment about how much she likes an image on a photo sharing website. Her client creates an Annotation with the comment embedded within it, and adds that it is in French and formatted using HTML.

The fundamental features of a Textual Body are:

Systems **should** assume that Textual Bodies have the **Text** class, described in **Classes** above, even if it is not explicitly included in the **type** property.

The properties of External Web Resources, such as **language** and **format** also apply to embedded Textual Body resources.

Example

EXAMPLE 5: Textual Body

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno5",
  "type": "Annotation",
  "body": {
    "type" : "TextualBody",
    "value" : "<p>j'adore !</p>",
    "format" : "text/html",
    "language" : "fr"
  },
  "target": "http://example.org/photo1"}
```

3.2.5 String Body

The simplest type of Body is a plain text string, without additional information or properties. This type of Body is useful for the simplest of Annotations only, and is **not recommended** for uses where the Body may need to be referred to from outside of the Annotation.

Example Use Case Francesca wants to create a quick Annotation from a simple, command line client. She creates the JSON serialization in a text file and sends it to her Annotation server to maintain.

There are several restrictions on when this form may be used and how it is to be interpreted. The string Body:

- **must** be a single **xsd:string** and the data type **must not** be expressed in the serialization.
- **must not** have a language associated with it.
- **must** be interpreted as if it were the value of the **value** property of a Textual Body.
- **must** be interpreted as if the Textual Body resource had a **format** property with the value **text/plain**.

- **must not** have the value of other properties of the Textual Body inferred from similar properties on the Annotation resource. If any of the interpretations above are not correct, then the **TextualBody construction must** be used instead.

Example

EXAMPLE 6: String Body

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno6",
  "type": "Annotation",
  "bodyValue": "Comment text",
  "target": "http://example.org/target1"
}
```

Which is equivalent to:

EXAMPLE 7: Equivalent Textual Body

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno7",
  "type": "Annotation",
  "body": {
    "type": "TextualBody",
    "value": "Comment text",
    "format": "text/plain"
  },
  "target": "http://example.org/target1"
}
```

3.2.6 Cardinality of Bodies and Targets

Some Annotations may not have a Body at all, such as a simple highlight or bookmark without any accompanying text. It is also possible for an Annotation to have multiple Bodies and/or Targets. In this case, each Body is considered to be equally related to each Target individually, rather than to the set of all of the Targets.

Example Use Case: Gretchen highlights a particular region of her ebook in green and, knowing what such a highlight means, she does not give a comment. Her client associates a stylesheet with the Annotation, and does not create a body at all.

Example Use Case: Hannah associates a tag and a description with two images using a single annotation.

Example

EXAMPLE 8: Annotations without a Body

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno8",
  "type": "Annotation",
  "target": "http://example.org/ebook1"
}
```

EXAMPLE 9: Multiple Bodies or Targets

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno9",
  "type": "Annotation",
  "body": [
    "http://example.org/description1",
    {

```

```

    "type": "TextualBody",

    "value": "tag1"

  }

],

"target": [

  "http://example.org/image1",

  "http://example.org/image2"

]
}

```

3.2.7 Choice Between Bodies

A Choice has an ordered list of resources from which an application should select only one to process or display.

Term	Type	Description
Id	Property	The IRI that identifies the Choice. The Choice may have exactly 1 IRI that identifies it.
Type	Relationship	The type of the resource. The Choice must have exactly 1 type , and it must be the CHOICE class.
Choice	Class	A construction that conveys to a consuming application that it should select one of the listed resources to display to the user, and not render all of them.
Items	Relationship	A list of resources to choose from, with the default option listed first.

Table-3.5

Example Use Case: Irina writes up her discussion of a particular website in both French and English. As the two posts are equivalent, there is no need to display both, and instead she wants French speakers to see the French comment, and everyone else to see the English version. Her client creates a Choice with the English comment listed first.

Example

EXAMPLE 10: Choice

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno10",
  "type": "Annotation",
  "body": {
    "type": "Choice",
    "items": [
      {
        "id": "http://example.org/note1",
        "language": "en"
      },
      {
        "id": "http://example.org/note2",
        "language": "fr"
      }
    ]
  },
  "target": "http://example.org/website1"
}
```

3.3 Other Properties

It is often important to have information about the context in which the Annotation and any External Web Resources were created, modified and used. In particular,

- When was the resource created, modified or generated

- Who created, modified or generated the serialized form of the Annotation or other resource, and who is it intended for
- Why was the resource included in the annotation, or the annotation created
- What other identities the resource has
- How can the resource be used, according to its rights and licensing

3.3.1 Lifecycle Information

The person, organization or machine responsible for the Annotation or referenced resource deserves credit for their contribution, and the time at which those resources are created is useful for display ordering and filtering out old, potentially irrelevant content. The creator of the Annotation is also useful for determining the trustworthiness of the Annotation. The software used to create and serialize the Annotation, along with when that activity occurred, is useful for both advertising and debugging issues.

Example Use Case: Jane writes a review of a restaurant online, and wishes to be associated with that review so that her friends know that it was her review and can trust it. Her client adds her account's identity, and its own identity, to the Annotation and the appropriate timestamps for when the resources were created.

Example

EXAMPLE 11: Lifecycle Information

```
{  
  
  "@context": "http://www.w3.org/ns/anno.jsonld",  
  
  "id": "http://example.org/anno11",  
  
  "type": "Annotation",  
  
  "creator": "http://example.org/user1",  
  
  "created": "2015-01-28T12:00:00Z",  
  
  "modified": "2015-01-29T09:00:00Z",  
  
  "generator": "http://example.org/client1",  
  
  "generated": "2015-02-04T12:00:00Z",  
  
}
```



```

"body": {
  "id": "http://example.net/review1",
  "creator": "http://example.net/user2",
  "created": "2014-06-02T17:00:00Z",
  "target": "http://example.com/restaurant1"}

```

3.3.2 Agents

More information about the agents involved in the creation of an Annotation is normally required beyond an IRI that identifies them. This includes whether they are an individual, a group or a piece of software and properties such as real name, account nickname, and email address.

Example Use Case: Kelly wants to submit an Annotation to a system that does not manage her identity, and would like a pseudonym to be displayed. Her client adds this information to the Annotation to send to the service.

Example

EXAMPLE 12: Agents

```

{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno12",
  "type": "Annotation",
  "creator": {
    "id": "http://example.org/user1",
    "type": "Person",
    "name": "My Pseudonym",
    "nickname": "pseudo",
    "email_sha1": "58bad08927902ff9307b621c54716dcc5083e339"
  },

```

```
"generator": {  
  
  "id": "http://example.org/client1",  
  
  "type": "Software",  
  
  "name": "Code v2.1",  
  
  "homepage": "http://example.org/client1/homepage1"  
  
},  
  
"body": "http://example.net/review1",  
  
"target": "http://example.com/restaurant1"}
```

3.3.3 Intended Audience

Beyond the agents that are associated with the creation and management of the Annotation and other resources, it is also useful to know the audience or class of consuming agent that the resource is intended for. This allows for the roles (such as teacher versus student) or properties of the class (such as a suggested age range) of the intended audience to be recorded.

Example Use Case: Lynda writes some notes about using a particular textbook to teach a class. She adds that the intended audience of the Annotation is teachers (who are using the textbook), to distinguish from other Annotations that might have an audience of the students (who are also using the textbook, but to learn from).

Further properties that describe the audience are used from schema.org's [Audience](http://schema.org) classes. The properties and class names **must** be prefixed in the JSON with **schema:** to ensure that they are uniquely distinguished from any other properties or classes.

The use of **audience** does not imply nor enable any access restriction to prevent the annotation from being seen. Systems **should** use the information for filtering the display of Annotations based on their knowledge of the user, and not assume that the Annotation or other resources will require authentication and authorization.

Example

EXAMPLE 13: Audience

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno13",
  "type": "Annotation",
  "audience": {
    "id": "http://example.edu/roles/teacher",
    "type": "schema:EducationalAudience",
    "schema:educationalRole": "teacher"
  },
  "body": "http://example.net/classnotes1",
  "target": "http://example.com/textbook1"
}
```

3.3.4 Accessibility of Content

Access to information is recognized as a basic human right by the United Nations. The Web is able to remove barriers to communication and interaction regardless of various physical impediments. This supports social inclusion, but also increases the potential audience of the information. For resources that are used as the Body or Target of an Annotation, it is valuable to record the features of that resource that provide easier access for users with various and diverse ranges of ability.

Example Use Case: Megan has very limited ability to hear sound, and prefers to read captions when interacting with videos. She uses her annotation client to make a comment on such a video, and to help others in the same situation, the client includes that the video has this accessibility feature.

Example

EXAMPLE 14: Accessibility

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno14",
  "type": "Annotation",
  "motivation": "commenting",
  "body": "http://example.net/comment1",
  "target": {
    "id": "http://example.com/video1",
    "type": "Video",
    "accessibility": "captions"
  }
}
```

3.3.5 Motivation and Purpose

In many cases it is important to understand the reasons why the Annotation was created, or why the Textual Body was included in the Annotation, not just the times and agents involved. These reasons are provided by declaring the motivation for the Annotation's creation or the purpose for the inclusion of the Textual Body in the Annotation; the "why" rather than the "who" and "when" described in the previous sections.

Example Use Case: Noelle annotates a resource intending to bookmark it for future reference, and provides a description and a tag to make it easier to find again. Her client adds the right motivations to the Annotation and the Textual Body resources to capture this.

Example

EXAMPLE 15: Motivation and Purpose

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno15",
  "type": "Annotation",
  "motivation": "bookmarking",
  "body": [
    {
      "type": "TextualBody",
      "value": "readme",
      "purpose": "tagging"
    },
    {
      "type": "TextualBody",
      "value": "A good description of the topic that bears further
investigation",
      "purpose": "describing"
    }
  ],
  "target": "http://example.com/page1"
}
```

3.3.6 Rights Information

It is common practice to associate a license or rights statement with a resource, in order to describe the conditions under which it may be used. This allows the user to make appropriate use of the resource, as well as allowing some automated systems to confirm that the usage is permitted. As the Annotation, Bodies, and Targets might be created with different licences or rights, each can be described separately. The rights of resources other than the Annotation itself are considered informative hints to a consuming client application.

Example Use Case: Ophelia writes a review of a product and wishes to be known as the author of the review, however does not mind how the Annotation that relates the review and the product together is used. She asserts these two separate rights statements with the Annotation and Body individually. She does not know the rights asserted on the target resource, so does not specify any.

Example

EXAMPLE 16: Rights

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno16",
  "type": "Annotation",
  "rights": "https://creativecommons.org/publicdomain/zero/1.0/",
  "body": {
    "id": "http://example.net/review1",
    "rights": "http://creativecommons.org/licenses/by-nc/4.0/"
  },
  "target": "http://example.com/product1"
}
```

3.3.7 Other Identities

In a massively distributed system such as the Web, information is often copied. In order to track the provenance of the Annotation and other related resources, it is possible to record additional IRIs that also identify the resource. These may be dereferencable "permalinks", identities assigned by a client offline without any knowledge of the web, or simply the location where the current harvesting system discovered the resource.

Example Use Case: Petra creates an Annotation and sends it to multiple systems to maintain, one personal and one public. She wants to ensure that the copies can be aligned, and so she sets a UUID as the canonical IRI, allowing the service to assign an HTTP IRI for it. A subsequent system then harvests the public copy, maintaining the canonical UUID as discovered, then moves the original HTTP IRI to **via**, replacing it with an IRI under its control.

Example

EXAMPLE 17: Other Identities

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno17",
  "type": "Annotation",
  "canonical": "urn:uuid:dbfb1861-0ecf-41ad-be94-a584e5c4f1df",
  "via": "http://other.example.org/anno1",
  "body": {
    "id": "http://example.net/review1",
    "rights": "http://creativecommons.org/licenses/by/4.0/"
  }, "target": "http://example.com/product1"}
```

Chapter-4

Specific Resources

While it is possible using only the constructions described above to create Annotations that reference parts of resources by using IRIs with a fragment component, there are many situations when this is not sufficient. For example, even a simple circular region of an image, or a diagonal line across it, are not possible. Selecting an arbitrary span of text in an HTML page, perhaps the simplest annotation concept, is also not supported by fragments. Furthermore, there are non-segment use cases that require a client to retrieve a specific state or representation of the resource, to style it in a particular way, to associate a role with the resource that is specific to the Annotation's use of it, or for the Annotation to only apply when the resource is used in a particular context.

The Web Annotation Data Model uses a new type of resource to capture these Annotation-specific requirements: a `SpecificResource`. The `SpecificResource` is used in between the Annotation and the Body or Target, as appropriate, to capture additional information about how it is used in the Annotation. The descriptions are typically referenced from the `SpecificResource` as separate entities and can be of various types to capture the different requirements. For example, if the Target of the Annotation is a circular region of an image, then the `SpecificResource` is the circular region, it is described by a Selector, and is also associated with the source Image resource.

Specific Resources and Specifiers **may** be External Web Resources with their own IRIs, such as in the example for the `Selector` construction, however it is **recommended** that they be included in the Annotation's representation to avoid requiring unnecessary network interactions to retrieve all of the information needed to process the Annotation.

The types of additional specificity that are defined by this document:

- **Purpose:** Describe the purpose of including the source resource in the Annotation
- **Selector:** Describe the desired segment of the source resource for the Annotation
- **State:** Describe the desired representation of the source resource for the Annotation
- **Style:** Describe the style in which the source resource should be presented for the Annotation
- **Rendering:** Describe the system used by the client for rendering the resource when the annotation was created

Model

Term	Type	Description
Id	Property	The identity of the Specific Resource. A Specific Resource may have exactly 1 IRI that identifies it.
Type	Relationship	The class of the Specific Resource. The Specific Resource should have the SpecificResource class.
SpecificResource	Class	The class for Specific Resources. The SpecificResource class should be associated with a Specific Resource to be clear as to its role as a more specific region or state of another resource.
source	Relationship	The relationship between a Specific Resource and the resource that it is a more specific representation of. There must be exactly 1 source relationship associated with a Specific Resource. The source resource may be described in detail, as defined above, or be just the resource's IRI.

Table-4.1

The same Specific Resource and Specifier classes are used for both Target and Body. The examples in this section only use one of these, however the same model applies for both.

4.1 Purpose for External Web Resources

As well as Textual Bodies, External Web Resources may also be given a Motivation as to their inclusion within the Annotation. This is done using the Specific Resource pattern, as the purpose specifies the way in which the resource is used in the context of the Annotation in the same way as a Selector describes the segment or a State describes the representation.

Example Use Case: Qitara wants to tag a photo with an identifier for a city, rather than just type the city's name which could be ambiguous. Her client uses a well-known IRI for the city having done a search for it, and creates a Specific Resource to manage that purpose assignment.

Model

Term	Type	Description
purpose	Relationship	The reason for including the Web Resource in the Annotation. There may be 0 or more Motivations associated with the SpecificResource using purpose .

Table-4.2

Example

EXAMPLE 18: Resource with Purpose

```
{  
  
  "@context": "http://www.w3.org/ns/anno.jsonld",  
  
  "id": "http://example.org/anno18",  
  
  "type": "Annotation",
```

```

"body": {

  "type": "SpecificResource",

  "purpose": "tagging",

  "source": "http://example.org/city1"

},

"target": {

  "id": "http://example.org/photo1""type": "Image"}}

```

4.2 Selectors

Many Annotations refer to part of a resource, rather than all of it, as the Target. We call that part of the resource a Segment (of Interest). A Selector is used to describe how to determine the Segment from within the Source resource. The nature of the Selector will be dependent on the type of resource, as the methods to describe Segments from various media-types will differ. Multiple Selectors can be given to describe the same Segment in different ways in order to maximize the chances that it will be discoverable later, and that the consuming user agent will be able to use at least one of the Selectors.

Example Use Case: Ramona wants to associate a selection of text in a web page, with a slice of a dataset. She selects both using her client, and creates the Annotation with a SpecificResource that has a Selector for each of the Body and the Target.

Model

Term	Type	Description
selector	Relationship	The relationship between a Specific Resource and a Selector. There may be 0 or more selector relationships associated with a Specific Resource. Multiple Selectors should select the same content, however some Selectors will not have the same precision as others. Consuming user agents must pick one of the described segments, if they are different.

Table-4.3

Example

EXAMPLE 19: Selectors

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno19",
  "type": "Annotation",
  "body": {
    "source": "http://example.org/page1",
    "selector": "http://example.org/paraselector1"
  },
  "target": {
    "source": "http://example.com/dataset1",
    "selector": "http://example.org/datasetselector1"
  }
}
```

4.2.1 Fragment Selector

As the most well understood mechanism for selecting a Segment is to use the fragment part of an IRI defined by the representation's media type, it is useful to allow this as a description mechanism via a Selector. This allows existing and future fragment specifications to be used with Specific Resources in a consistent way. To be clear about which fragment type is being used, the Selector may refer to the specification that defines it. It is **recommended** to use **FragmentSelector** as a consistent method compatible with other means of describing SpecificResources, rather than using the IRI with a fragment directly. Consuming applications **should** be aware of both.

The following IRIs are some of the specifications that define the semantics of fragments, and hence may be used with the **conformsTo** relationship. Other IRIs **may** also be used.

Example Use Case: Sally wants to associate part of a video as the description of an image. She selects the time range within the video and clicks that it is describing the target. Her client then creates the Annotation using a `SpecificResource` with a `FragmentSelector` and the **describing** Motivation.

Example

EXAMPLE 20: Fragment Selector

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno20",
  "type": "Annotation",
  "body": {
    "source": "http://example.org/video1",
    "purpose": "describing",
    "selector": {
      "type": "FragmentSelector",
      "conformsTo": "http://www.w3.org/TR/media-frags/",
      "value": "t=30,60"
    }
  },
  "target": "http://example.org/image1"}
```

4.2.2 CSS Selector

One of the most common ways to select elements in the HTML Document Object Model is to use CSS Selectors [[CSS3-selectors](#)]. CSS Selectors allow for a wide variety of well supported ways to describe the path to an element in a web page, and thus cover many of the basic use cases for Web Annotation. Results are not defined for when a CSS Selector is applied to a representation that does not conform to the Document Object Model.

Note that CSS may also be used for [styling](#) a resource within an annotation. This class is specifically to re-use the CSS Selector mechanism to select a segment of a resource that conforms to the Document Object Model.

Example Use Case: Teynika selects a paragraph in a web page that she wishes to write a note about. Her client calculates a CSS path that cleanly identifies that element and adds it to the annotation.

Example

EXAMPLE 21: CSS Selector

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno21",
  "type": "Annotation",
  "body": "http://example.org/notel",
  "target": {
    "source": "http://example.org/page1.html",
    "selector": {
      "type": "CssSelector",
      "value": "#elemid > .elemclass + p"
    }
  }
}
```

4.2.3 XPath Selector

Another common method of selecting elements and content within a resource that supports the Document Object Model (DOM), such as documents in XML or HTML, is to use an XPath selection [[DOM-Level-3-XPath](#)]. XPath allows a great deal of flexibility when describing the path through the structure to the selected content. Results are not defined for when an XPath Selector is applied to a representation that does not conform to the DOM.

Example Use Case: Ulrika selects a span within a table in an HTML page and writes a note about the content. To refer explicitly to this element, her client carefully constructs an XPath to identify it as the target of the Annotation.

Example

EXAMPLE 22: XPath Selector

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno22",
  "type": "Annotation",
  "body": "http://example.org/note1",
  "target": {
    "source": "http://example.org/page1.html",
    "selector": {
      "type": "XPathSelector",
      "value": "/html/body/p[2]/table/tr[2]/td[3]/span"
    }
  }
}
```

4.2.4 Text Quote Selector

This Selector describes a range of text by copying it, and including some of the text immediately before (a prefix) and after (a suffix) it to distinguish between multiple copies of the same sequence of characters.

For example, if the document were again "abcdefghijklmnopqrstuvwxyz", one could select "efg" by a prefix of "abcd", the match of "efg" and a suffix of "hijk".

Example Use Case: Valeria selects a typo ('anotation') in a web page and adds a comment that it should be replaced with the correct spelling ('annotation').

Model

Term	Type	Description
Type	Relationship	The class of the Selector. Text Quote Selectors must have exactly 1 type and the value must be TextQuoteSelector .
TextQuoteSelector	Class	The class for a Selector that describes a textual segment by means of quoting it, plus passages before or after it. The TextQuoteSelector must have this class associated with it.
Exact	Property	A copy of the text which is being selected, after normalization. Each TextQuoteSelector must have exactly 1 exact property.
Prefix	Property	A snippet of text that occurs immediately before the text which is being selected. Each TextQuoteSelector should have exactly 1 prefix property, and must not have more than 1.

Suffix	Property	<p>The snippet of text that occurs immediately after the text which is being selected.</p> <p>Each TextQuoteSelector should have exactly 1 suffix property, and must not have more than 1.</p>
--------	----------	---

Table-4.4

The selection of the text **must** be in terms of unicode code points (the "character number"), not in terms of code units (that number expressed using a selected data type). Selections **should not** start or end in the middle of a grapheme cluster. The selection **must** be based on the **logical order** of the text, rather than the visual order, especially for bidirectional text. For more information about the character model of text used on the web, see [[charmod](#)].

The text **must** be normalized before recording in the Annotation. Thus HTML/XML tags **should** be removed, and character entities **should** be replaced with the character that they encode. Note that this does not affect the state of the content of the document being annotated, only the way that the content is recorded in the Annotation document.

If, after processing the prefix, exact, and suffix, the user agent discovers multiple matching text sequences, then the selection **should** be treated as matching all of the matches.

Example

EXAMPLE 23: Text Quote Selector

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno23",
  "type": "Annotation",
  "body": "http://example.org/comment1",
  "target": {
    "source": "http://example.org/page1",
    "selector": {
      "type": "TextQuoteSelector",
```

```
"exact": "anotation",

"prefix": "this is an ",

"suffix": " that has some"} } }
```

4.2.5 Text Position Selector

This Selector describes a range of text by recording the start and end positions of the selection in the stream. Position 0 would be immediately before the first character, position 1 would be immediately before the second character, and so on. The start character is thus included in the list, but the end character is not.

For example, if the document was "abcdefghijklmnopqrstuvwxy^z", the start was 4, and the end was 7, then the selection would be "efg".

Example Use Case: Wendy writes a review of an ebook that does not allow its content to be extracted and copied. Her client describes the selection using its start and end position in the content.

Example

EXAMPLE 24: Text Position Selector

```
{

"@context": "http://www.w3.org/ns/anno.jsonld",

"id": "http://example.org/anno24",

"type": "Annotation",

"body": "http://example.org/review1",

"target": {

  "source": "http://example.org/ebook1",

  "selector": {

    "type": "TextPositionSelector",

    "start": 412,
```

```
"end": 795

}  }}
```

4.2.6 Data Position Selector

Similar to the Text Position Selector, the Data Position Selector uses the same properties but works at the byte in bitstream level rather than the character in text level.

Example Use Case: Xena writes comments about regions of online disk images for forensic purposes and describing emulation requirements. Her client generates the start and end positions from the binary stream, rather than the more human readable display she is using.

Example

EXAMPLE 25: Data Position Selector

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno25",
  "type": "Annotation",
  "body": "http://example.org/note1",
  "target": {
    "source": "http://example.org/diskimg1",
    "selector": {
      "type": "DataPositionSelector",
      "start": 4096,
      "end": 4104
    }
  }
}
```

4.2.7 SVG Selector

An SvgSelector defines an area through the use of the Scalable Vector Graphics [SVG11] standard. This allows the user to select a non-rectangular area of the content, such as a circle or polygon by describing the region using SVG. The SVG may be either embedded within the Annotation or referenced as an External Web Resource.

Note that the SvgSelector uses SVG to select an area of a resource. Segments of an SVG representation may also be selected using selectors, including the FragmentSelector or even an SvgSelector.

Example Use Case: Yadira is tagging an old map online with a diagonal region for a historical road. Her client creates SVG polygon to describe the region, relative to the image content.

Term	Type	Description
Type	Relationship	The class of the Selector. SVG Selectors must have exactly 1 type and the value must include SvgSelector .
SvgSelector	Class	The class for a Selector which defines a shape for the selected area using the SVG standard. The Selector must have this class associated with it.
Value	Property	The character sequence of the SVG content. There may be exactly 1 value property associated with the Selector, and if so the value of the property must be well-formed SVG XML.

Table-4.5

Example

EXAMPLE 26: SVG Selector

```
{  
  
  "@context": "http://www.w3.org/ns/anno.jsonld",  
  
  "id": "http://example.org/anno26",
```

```

"type": "Annotation",

"body": "http://example.org/road1",

"target": {

  "source": "http://example.org/map1",

  "selector": {

    "id": "http://example.org/svg1",

    "type": "SvgSelector"

  }

}

```

EXAMPLE 27: SVG Selector, embedded

```

{

  "@context": "http://www.w3.org/ns/anno.jsonld",

  "id": "http://example.org/anno27",

  "type": "Annotation",

  "body": "http://example.org/road1",

  "target": {

    "source": "http://example.org/map1",

    "selector": {

      "type": "SvgSelector",

      "value": "<svg:svg> ... </svg:svg>"

    }

  }

}

```

4.2.8 Range Selector

Selections made by users may be extensive and/or cross over internal boundaries in the representation, making it difficult to construct a single selector that robustly describes the correct content. A Range Selector can be used to identify the beginning and the end of the selection by using other Selectors. In this way, two points can be accurately identified using the most appropriate selection mechanisms, and then linked together to form the selection. The selection consists of everything from the beginning of the starting selector through to the beginning of the ending selector, but not including it.

Example Use Case: Zara wants to comment on two adjacent cells in a table that is part of a web page. She selects the two cells and her client constructs XPaths to the the first cell, and the cell that immediately follows the second. Her client then creates a Range Selector with the first XPath Selector as the start, and the second XPath selector as the end.

Example

EXAMPLE 28: Range Selector

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno28",
  "type": "Annotation",
  "body": "http://example.org/comment1",
  "target": {
    "source": "http://example.org/page1.html",
    "selector": {
      "type": "RangeSelector",
      "startSelector": {
        "type": "XPathSelector",
        "value": "//table[1]/tr[1]/td[2]"
      },
    },
  },
}
```

```

    "endSelector": {

      "type": "XPathSelector",

      "value": "//table[1]/tr[1]/td[4]"

    }

  }

}

```

4.2.9 Refinement of Selection

It may be easier, more reliable or more accurate to specify the segment of interest of a resource as a selection of a selection, rather than as a selection of the complete resource. Particularly for resources that contain other resources, such as various packaging formats, this also allows decomposition of the selection mechanisms when the components do not have unique identifiers. This is accomplished by having selectors chained together, where each refines the results of the previous one.

Example Use Case: Alexandra selects a paragraph of text and then a short phrase within it to comment on. Her client records the phrase as a TextQuoteSelector that further modifies a FragmentSelector used to identify the paragraph that the phrase is part of.

Model

Term	Type	Description
refinedBy	Relationship	<p>The relationship between a broader selector and the more specific selector that should be applied to the results of the first.</p> <p>A Selector may be refinedBy 1 or more other Selectors. If more than 1 is given, then they are considered to be alternatives that will result in the same selection.</p>

Table-4.6

Example

EXAMPLE 29: Refinement of Selection

```
{  
  
  "@context": "http://www.w3.org/ns/anno.jsonld",  
  
  "id": "http://example.org/anno29",  
  
  "type": "Annotation",  
  
  "body": "http://example.org/comment1",  
  
  "target": {  
  
    "source": "http://example.org/page1",  
  
    "selector": {  
  
      "type": "FragmentSelector",  
  
      "value": "para5",  
  
      "refinedBy": {  
  
        "type": "TextQuoteSelector",  
  
        "exact": "Selected Text",  
  
        "prefix": "text before the ",  
  
        "suffix": " and text after it"      }    }  }  }
```

4.3 States

A State describes the intended state of a resource as applied to the particular Annotation, and thus provides the information needed to retrieve the correct representation of that resource. Web resources change over time, and a State might be used to describe how to recover the intended previous version. Web resources also have multiple formats, and a State might equally be used to describe how to retrieve that particular format. Multiple States may be given to describe the same representation in order to maximize the chances that the representation will be retrievable by the consuming user agent.

Example Use Case: Britney makes a comment about a web page that changes frequently. Her client records information to allow other clients to hopefully reconstruct the original target of the annotation.

Example

EXAMPLE 30: State

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno30",
  "type": "Annotation",
  "body": "http://example.org/note1",
  "target": {
    "source": "http://example.org/page1",
    "state": {
      "id": "http://example.org/state1"
    }
  }
}
```

4.3.1 Time State

A Time State resource records the time at which the resource is appropriate for the Annotation, typically the time that the Annotation was created and/or a link to a persistent copy of the current version. The timestamp for the resource could be resolved via the Memento protocol, described in RFC 7089 [[rfc7089](#)].

Example Use Case: Carla makes a note about the current state of the front page of a news website, and flags that the page is likely to change often. Her client adds in a State with the current time to describe the version of the page being annotated.

Example

EXAMPLE 31: Time State

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno31",
  "type": "Annotation",
  "body": "http://example.org/note1",
  "target": {
    "source": "http://example.org/page1",
    "state": {
      "type": "TimeState",
      "cached": "http://archive.example.org/copy1",
      "sourceDate": "2015-07-20T13:30:00Z"
    }
  }
}
```

4.3.2 Request Header State

As there are potentially many representations that can be delivered from a resource with a single IRI, and a Specific Resource may only apply to one of them, it is important to be able to record the HTTP Request headers that need to be sent to retrieve the correct representation. The `HttpRequestState` resource maintains a copy of the headers to be replayed when obtaining the representation.

Term	Type	Description
------	------	-------------

Type	Relationship	The class of the State. Request Header States must have exactly 1 type and the value must be HttpRequestState .
HttpRequestState	Class	A description of how to retrieve an appropriate representation of the Source resource for the Annotation, based on the HTTP Request headers to send on the request. The State must have this class associated with it.
Value	Property	The HTTP request headers to send as a single, complete string. An HttpRequestState must have exactly 1 value property.

Table-4.7

Example Use Case: Devina retrieves a PDF representation of a web resource that can deliver HTML, PDF or plain text and then writes a description about it. She signals that her description is only about the PDF representation. Her client then includes a State to describe how to retrieve the target representation.

Example

EXAMPLE 32: HTTP Request State

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno32",
  "type": "Annotation",
  "body": "http://example.org/description1",
  "target": {
    "source": "http://example.org/resource1",
    "state": {
      "type": "HttpRequestState",
```

```

    "value": "Accept: application/pdf"

  }

}

}

```

4.3.3 Refinement of State

Similar to the [refinement of selection](#), it may be easier, more reliable or more accurate to specify the appropriate state of the resource as a hierarchy of atomic State resources. This is particularly appropriate for representing the combination of a State that reflects an internal transformation along with the results of a State that describes an external request. This decomposition is accomplished by having the states chained together in the same way as Selectors.

Further, given that the State(s) will likely result in a specific representation, there may be specific Selectors that are appropriate for describing the segment of the representation. In order to accommodate this, States may also be refined by Selectors.

Example Use Case: Erin writes a comment about a travel e-book which has many versions available over time, and is available in different formats. She is particularly commenting on a specific version and format, so her client adds both a TimeState to capture the time and an HttpRequestState to capture the format, and then a particular FragmentSelector that is appropriate to that format.

Model

Term	Type	Description
refinedBy	Relationship	The relationship between a broader State and either a more specific State or a Selector that should be applied to the results of the first. Each State may be refinedBy 1 or more other States or Selectors.

Table-4.8

Example

EXAMPLE 33: Refinement of States

```

{

  "@context": "http://www.w3.org/ns/anno.jsonld",

```

```

    "id": "http://example.org/anno33",

    "type": "Annotation",

    "body": "http://example.org/comment1",

    "target": {

        "source": "http://example.org/ebook1",

        "state": {

            "type": "TimeState",

            "sourceDate": "2016-02-01T12:05:23Z",

            "refinedBy": {

                "type": "HttpRequestState",

                "value": "Accept: application/pdf",

                "refinedBy": {

                    "type": "FragmentSelector",

                    "value": "page=10",

                    "conformsTo": "http://tools.ietf.org/rfc/rfc3778"

                }

            }

        }

    }

}

```

4.4 Styles

The interpretation of a particular Annotation, or the Annotation's Body, may rely on the rendering style of the Annotation being consistent across implementations. For Annotations on binary content such as Images or Video, the background color of the Target may not be accessible to the annotation client, and the default coloring may be difficult to perceive, such as a black rectangle rendered as the target area on an image of the night sky. Rendering

information is recorded using CSS stylesheets and references to classes defined in those stylesheets.

Example Use Case: Felicity highlights two paragraphs in a document, and selects in her client that one should be highlighted in red and the other in yellow. She then makes a comment that the yellow part contradicts the red part. Her client records that she selected the red and yellow coloration of the targets.

The CSS Stylesheet is associated with the Annotation itself, and the content provides the rendering hints about the Annotation's constituent resources. It **may** have its own dereferenceable IRI that provides the information, or it may be embedded within the Annotation. This is to avoid having single line stylesheets each associated with different resources, and instead to allow reference to a single IRI that governs the full set of styles for a particular implementation.

Publishing systems **must not** assume that they will be processed; they are only provided as hints rather than requirements.

When rendering a Specific Resource, consuming applications **should** check to see if it has a **styleClass** property. If it does, then the application **should** attempt to locate the appropriate selector in the CSS document, and then apply the css-value block. If a Specific Resource has a **styleClass** value, but no such class is described by a **stylesheet** attached to the Annotation, then the **styleClass** **must** be ignored.

Example

EXAMPLE 34: CSS Style

```
{  
  
  "@context": "http://www.w3.org/ns/anno.jsonld",  
  
  "id": "http://example.org/anno34",  
  
  "type": "Annotation",  
  
  "stylesheet": "http://example.org/style1",  
  
  "body": "http://example.org/comment1",  
  
  "target": {
```

```

    "source": "http://example.org/document1",

    "styleClass": "red"

  }
}

```

EXAMPLE 35: CSS Style, embedded

```

{

  "@context": "http://www.w3.org/ns/anno.jsonld",

  "id": "http://example.org/anno35",

  "type": "Annotation",

  "stylesheet": {

    "type": "CssStylesheet",

    "value": ".red { color: red }"

  },

  "body": "http://example.org/body1",

  "target": {

    "source": "http://example.org/target1",

    "styleClass": "red"

  }

}

```

4.5 Rendering Software

It may be valuable to know the software that was used to process and/or render the Target resource when the annotation was created. This information can be used by later systems to potentially recreate the environment to ensure that the annotation can be more easily and more accurately reconnected with the appropriate part of the Target's representation. This life cycle information is associated with the Specific Resource, as it is very likely to change between

Annotations for the same Target, and thus cannot be associated with the Target resource directly.

Example Use Case: Gabrielle is using a browser based client to render a PDF of a scholarly article. Her browser uses a particular library to render the PDF as HTML. She annotates a paragraph in the view that she sees, the HTML rendering, and her client records that the library that was used for rendering in the annotation, along with her comment and the target PDF.

Example

EXAMPLE 36: Rendering Software

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/anno36",
  "type": "Annotation",
  "body": "http://example.org/comment1",
  "target": {
    "source": "http://example.edu/article.pdf",
    "selector": "http://example.org/selectors/html-selector1",
    "renderedVia": {
      "id": "http://example.com/pdf-to-html-library",
      "type": "Software",
      "schema:softwareVersion": "2.5"
    }
  }
}
```

4.6 Scope of a Resource

It is sometimes important for an Annotation to capture the context in which it was made, in terms of the resources that the annotator was viewing or using at the time. This does not imply

an assertion that the annotation is only valid for the image in the context of that page, it just records that the page was being viewed.

Example Use Case: Heather makes a comment about an image in a particular web page to say that it is not the right organization's logo. Her client includes the page that the image is being rendered in, however the annotation is associated with the image resource itself.

Example

EXAMPLE 37: Scope

```
{  
  
  "@context": "http://www.w3.org/ns/anno.jsonld",  
  
  "id": "http://example.org/anno37",  
  
  "type": "Annotation",  
  
  "body": "http://example.org/note1",  
  
  "target": {  
  
    "source": "http://example.org/image1",  
  
    "scope": "http://example.org/page1"}}
```

Chapter-5

5.Collections

It is often useful to be able to collect Annotations together into a list, called an Annotation Collection. This list, which is always ordered, serves as a means to refer to the Annotations that are contained within it, and to maintain any information about the Collection itself.

The Collection model is divided into two sections: the Annotation Collection that manages the identity of the list and its description, and Annotation Pages that list the Annotations which are members of the Collection.

Example Use Case: Ingeborg works for a publishing house and has transformed the author's commentary on their steampunk novel into a set of annotations for sale. The company wishes to have them available as an add-on for customers that have already bought the novel, and also in a bundle for new sales.

5.1 Annotation Collection

As Annotation Collections might get very large, the model distinguishes between the Collection itself and sequence of component pages that in turn list the Annotations. The Collection maintains information about itself, including creation or descriptive information to aid with discovery and understanding of the Collection, and also references to at least the first Page of Annotations. By starting with the first Annotation in the first Page, and traversing the Pages to the last Annotation of the last Page, all Annotations in the Collection will have been discovered. Annotations **may** be within multiple Collections at the same time, and Collection **may** be created or maintained by agents other than those that create or maintain the included Annotations.

Other properties **may** be added to the Collection to describe its use, intellectual property rights, provenance and any other features that are considered useful. These properties **should** come from those described in this specification if possible, but **may** come from any appropriate vocabulary.

Example

EXAMPLE 38: Annotation Collection

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/collection1",
  "type": "AnnotationCollection",
  "label": "Steampunk Annotations",
  "creator": "http://example.com/publisher",
  "total": 42023,
  "first": "http://example.org/page1",
  "last": "http://example.org/page42"}
```

5.2 Annotation Page

An Annotation Page is part of an Annotation Collection, and has an ordered list of some or all of the Annotations that are within the Collection. Each Collection may have multiple pages, and these are traversed by following the **next** and **prev** links between the pages.

Example

EXAMPLE 39: Annotation Page

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/page1",
  "type": "AnnotationPage",
  "partOf": {
    "id": "http://example.org/collection1",
    "label": "Steampunk Annotations",
```

```

    "total": 42023

  },

  "next": "http://example.org/page2",

  "startIndex": 0,

  "items": [

    {

      "id": "http://example.org/anno1",

      "type": "Annotation",

      "body": "http://example.net/comment1",

      "target": "http://example.com/book/chapter1"

    },

    {

      "id": "http://example.org/anno2",

      "type": "Annotation",

      "body": "http://example.net/comment2",

      "target": "http://example.com/book/chapter2"

    }

  ]

}

```

EXAMPLE 40: Annotation Collection with Embedded Page

```

{

  "@context": "http://www.w3.org/ns/anno.jsonld",

  "id": "http://example.org/collection1",

  "type": "AnnotationCollection",

  "label": "Two Annotations",

}

```

```
"total": 2,

"first": {

  "id": "http://example.org/page1",

  "type": "AnnotationPage",

  "startIndex": 0,

  "items": [

    {

      "id": "http://example.org/anno1",

      "type": "Annotation",

      "body": "http://example.net/comment1",

      "target": "http://example.com/book/chapter1"

    },

    {

      "id": "http://example.org/anno2",

      "type": "Annotation",

      "body": "http://example.net/comment2",

      "target": "http://example.com/book/chapter2"

    }

  ]

}

}
```

Chapter-6

6. Correspondence Among Media Types and Selectors

The table below shows the relationships among major media types and selector types. It is relevant to the 1.3 [Conformance](#) section of this document.

	Fragment	CSS	XPath	Text Quote	Text Position	Data Position	Svg
HTML (text/html)	✓	✓	✓	✓	✓	X	X
CSV (text/csv)	✓	X	X	✓	✓	X	X
Plain Text (text/plain)	✓	X	X	✓	✓	X	X
Other text files (text/*)	?	X	X	✓	✓	X	X
EPUB2, EPUB3 (application/epub+zip)	✓	X	X	✓	X	X	X
PDF (application/pdf)	✓	X	X	✓	✓	X	X
XML (application/xml, application/*+xml)	✓	✓	✓	✓	✓	X	X
SVG (image/svg+xml)	✓	✓	✓	✓	✓	X	✓
Image, other than SVG (image/gif, image/jpeg, image/png, image/tiff)	✓	X	X	X	X	?	✓
Video (video/*)	✓	X	X	X	X	?	✓
Binary Data Files	?	X	X	X	X	✓	X

Table-6.1

Chapter-7

7.Sets of Bodies and Targets.

While it is possible to annotate multiple targets, the meaning of that annotation is that each Body applies independently to each of the Targets. This might not be the intent of the annotator, such as when all of the targets are required for the annotation to be correctly understood. In order to allow annotators to capture these requirements, a resource similar to Choice could be used, such as a Composite (unordered) or List (ordered).

The technical implementation of this pattern is not difficult, as it is practically identical to Choice, however the implementation of a user interface that can manage a human user's interactions such that the client can recognize the distinctions has proven to be very challenging. As such, the pattern is noted in this appendix for future consideration.

Example Use Case: Karin comments on a set of web pages as, together, providing evidence towards her research hypothesis. Her client creates a Composite, as there is no inherent order to the set of web pages.

Example Use Case: Lana tags a list of pages within a book as being important. As the pages have an order in the book, her client creates a List to maintain that order.

Example Use Case: Melanie annotates a set of images to classify them as portraits. As the classification applies to each image independently, her client creates a Independents resource to group them.

7.1 Proposed Model

Term	Type	Description
Id	Property	The IRI that identifies the set. The set resource may have exactly 1 IRI that identifies it.
Type	Relationship	The type of the resource. The set must have exactly 1 type class, taken from the options below.

Composite	Class	A set of resources, all of which are required for the Annotation to be correctly interpreted.
List	Class	An ordered list of resources, all of which are required in order for the Annotation to be correctly interpreted.
Independents	Class	A set of resources, each of which is being annotated separately with the same interpretation as having multiple bodies or targets directly associated with the Annotation.
items	Relationship	The list of resources in the Composite , List , or Independents .

Table-7.1

7.2 Candidate Recommendation Exit Criteria

For this specification to be advanced to Proposed Recommendation, there must be at least two independent implementations of each feature described below. Each feature may be implemented by a different set of products, and there is no requirement that any single product implement every feature.

Features

For the purposes of evaluating exit criteria, the following are considered as features:

- The Annotation class and required properties.
- The Agent class and required properties, as related to an Annotation.
- The Agent class and required properties, as related to a resource used as the Body of an Annotation.
- Embedded TextualBody class and required properties.
- External web resources, used as the Body of an Annotation.
- A Choice of resources, used as the Body of an Annotation.
- The SpecificResource class and required properties, used as the Body of an Annotation.
- External web resources, used as the Target of an Annotation.
- The SpecificResource class and required properties, used as the Target of an Annotation.
- The AnnotationCollection class and required properties.
- The AnnotationPage class and required properties.

Conclusion

These are conclusions reached from the project are Considerable interest has been shown in standardizing Web annotations , especially among publishers and those in the e-learning field; there has also been interest from some e-reading system implementers in providing client-side functionality. Based on this interest, The formation of a Web Annotation Working Group will depend upon the interest shown by our members during this review.

While there has been less overt interest from browser vendors in implementing client-side annotation functionality, there is interest in solving certain parts of the robust anchoring issue, such as standardized selection, ranging, a find-in-page API, and styling selections. To address these parts of the robust anchoring.

Appendix

```
<div class="container">

  <div class="page-header">
    <h1>This is a search front-end for YaCy!</h1>
    <p class="lead">Retrieval of search results using YaCys search API and display using AJAX technology.</p>
  </div>

  <h2>Solr, JSON(P) and JavaScript / backbone.js - driven</h2>
  <p>Search results are displayed using AJAX-technology from a Solr server which is embedded into <a href="http://yacy.net">YaCy</a>. All search results must be provided by a YaCy search server which includes a Solr with a specialized JSON result writer. When a search request is made in one of the search templates, a http request is made to YaCy. The response is done in JSON because of the <a href="http://en.wikipedia.org/wiki/same_origin_policy">same origin policy</a> in JavaScript, the result is not directly available. To overcome the <a href="http://en.wikipedia.org/wiki/same_origin_policy">same origin policy</a> in JavaScript, the result is not directly available. This enables you to run YaCy anywhere and to use the results from this server somewhere else, maybe in static web pages, even if you like, then you can also get search results from the same query url by replacing the "wt=yjson"-parameter by "wt=opensearch". We implemented a proper model view of search results using the <a href="http://en.wikipedia.org/wiki/Model_View_ViewModel">Model View ViewModel</a> pattern.
  </p>
  <h2>Industry-Strength Search Efficiency</h2>
  <p>Because the search results come right from a <a href="http://lucene.apache.org/solr/">Solr</a> instance using a specialized JSON result writer, this is an unique combination of Solr, JSON(P), flexible JavaScript presentation the beautiful YAMML4 CSS Framework and the elegant Backbone.js.
  </p>
  <h2>Standard Compliance</h2>
  <p>There are standards for search request queries (i.e. <a href="http://www.loc.gov/standards/sru/">SRU</a>) and search request responses (i.e. <a href="http://www.loc.gov/standards/sru/">SRU</a>). YaCy provides both! Actually these search templates send SRU requests to YaCy and the jsonp result writer in Solr (inside YaCy) returns the results in JSON(P) format. If you like, then you can also get search results from the same query url by replacing the "wt=yjson"-parameter by "wt=opensearch".
  </p>
  <h2>Beautiful CSS Framework</h2>
  <p>These pages are made with the <a href="https://github.com/yacy/searchpage-template-yaml4">YAML4 CSS Framework</a> and it is highly customizable. Just use the template as provided in the git repository (see below: 'Clone This!') and create your own search portal.
  </p>
</div>
```

Screenshot of Aboutus.html

```
<script type="text/javascript">
function handleArrowKeys(evt) {
  evt = (evt) ? evt : ((window.event) ? event : null);
  if (evt) {
    switch (evt.keyCode) {
      case 9:
      case 33:
        window.location.href = document.getElementById("nextpage").href;
        break;
      case 34:
        window.location.href = document.getElementById("prevpage").href;
        break;
      case 40:
      case 38:
        break;
    }
  }
}
document.onkeydown = handleArrowKeys;
</script>
<script type="text/javascript" src="../bootstrap/js/typeahead.jquery.min.js"></script>
<script type="text/javascript">
var suggestMatcher = function() {
  return function opensearch(q, cb) {
    $.getJSON(suggestUrl + "&q=" + q, function(data) {
      var parsed = [];
      for (var i = 0; i < data[0][1].length; i++) {
        var row = data[0][1][i];
        if (row) {
          parsed[parsed.length] = {
            data: [row],
            value: row,
            result: row
          };
        }
      }
    });
  };
};
</script>
```

Screenshot of Index.html

References

1.[CSS3-selectors]

Selectors Level 3. Tantek Çelik; Erika Etemad; Daniel Glazman; Ian Hickson; Peter Linss; John Williams et al. W3C. 29 September 2011. W3C Recommendation. URL: <https://www.w3.org/TR/css3-selectors/>

2.[DOM-Level-3-XPath]

Document Object Model (DOM) Level 3 XPath Specification. Ray Whitmer. W3C. 26 February 2004. W3C Note. URL: <https://www.w3.org/TR/DOM-Level-3-XPath/>

3.[JSON-LD]

JSON-LD 1.0. Manu Sporny; Gregg Kellogg; Markus Lanthaler. W3C. 16 January 2014. W3C Recommendation. URL: <https://www.w3.org/TR/json-ld/>

4.[RFC2119]

Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF. March 1997. Best Current Practice. URL: <https://tools.ietf.org/html/rfc2119>

5.[SVG11]

Scalable Vector Graphics (SVG) 1.1 (Second Edition). Erik Dahlström; Patrick Dengler; Anthony Grasso; Chris Lilley; Cameron McCormack; Doug Schepers; Jonathan Watt; Jon Ferraiolo; Jun Fujisawa; Dean Jackson et al. W3C. 16 August 2011. W3C Recommendation. URL: <https://www.w3.org/TR/SVG11/>

6.[annotation-protocol]

Web Annotation Protocol. Robert Sanderson. W3C. W3C Recommendation. URL: <http://www.w3.org/TR/annotation-protocol/>

7.[annotation-vocab]

Web Annotation Vocabulary. Robert Sanderson; Paolo Ciccicarese; Benjamin Young. W3C. W3C Recommendation. URL: <http://www.w3.org/TR/annotation-vocab/>

8.[bcp47]

Tags for Identifying Languages. A. Phillips; M. Davis. IETF. September 2009. IETF Best Current Practice. URL: <https://tools.ietf.org/html/bcp47>

9.[cfi]

EPUB Canonical Fragment Identifiers. Peter Sorotokin; Garth Conboy; Brady Duga; John Rivlin; Don Beaver; Kevin Ballard; Alastair Fettes; Daniel Weck. IDPF. Recommended Specification. URL: <http://www.idpf.org/epub/linking/cfi/epub-cfi-20140628.html>

Team Members

Anubha Srivastava(161B038)

B.Tech 4th Year

Computer Science & Engineering

Jaypee University of Engineering & Technology, Guna, M.P. (473226)

Mob : 9335630301



Email: Anubha1998@gmail.com

Kartikae Mishra161B107)

B.Tech 4th Year

Computer Science & Engineering

Jaypee University of Engineering & Technology, Guna, M.P. (473226)

Mob : 8707014579

Email: Karikae.mishra@gmail.com

Ritik Vishwakarma(161B179)

B.Tech 4th Year

Computer Science & Engineering

Jaypee University of Engineering & Technology, Guna, M.P. (473226)

Mob : 9807779333

Email: aarvii@gmail.com