Linked Data Apps with React

Made easy
with Link-Lib + Link-Redux

Slides: http://bit.ly/solid-link

Where is Link used?

- Argu
  - Back-end and front-end frameworks Open Source
- RIVM
- Open Raadsinformatie Search
  - Open Source
- Volunteer platform (can’t show due to GDPR)
  - Due to be Open Source
- Mash Browser
  - Open Source; dynamically extendible

Ontola.io

What’s in the box?

- Data fetching
- View management
  - Smart lookups
  - Error and Loading handling
  - Adapts to data
- Rule-based inference engine
- Actions / Data manipulation
  - Move logic between client and server
- Developer tools

Ontola.io

Use-cases

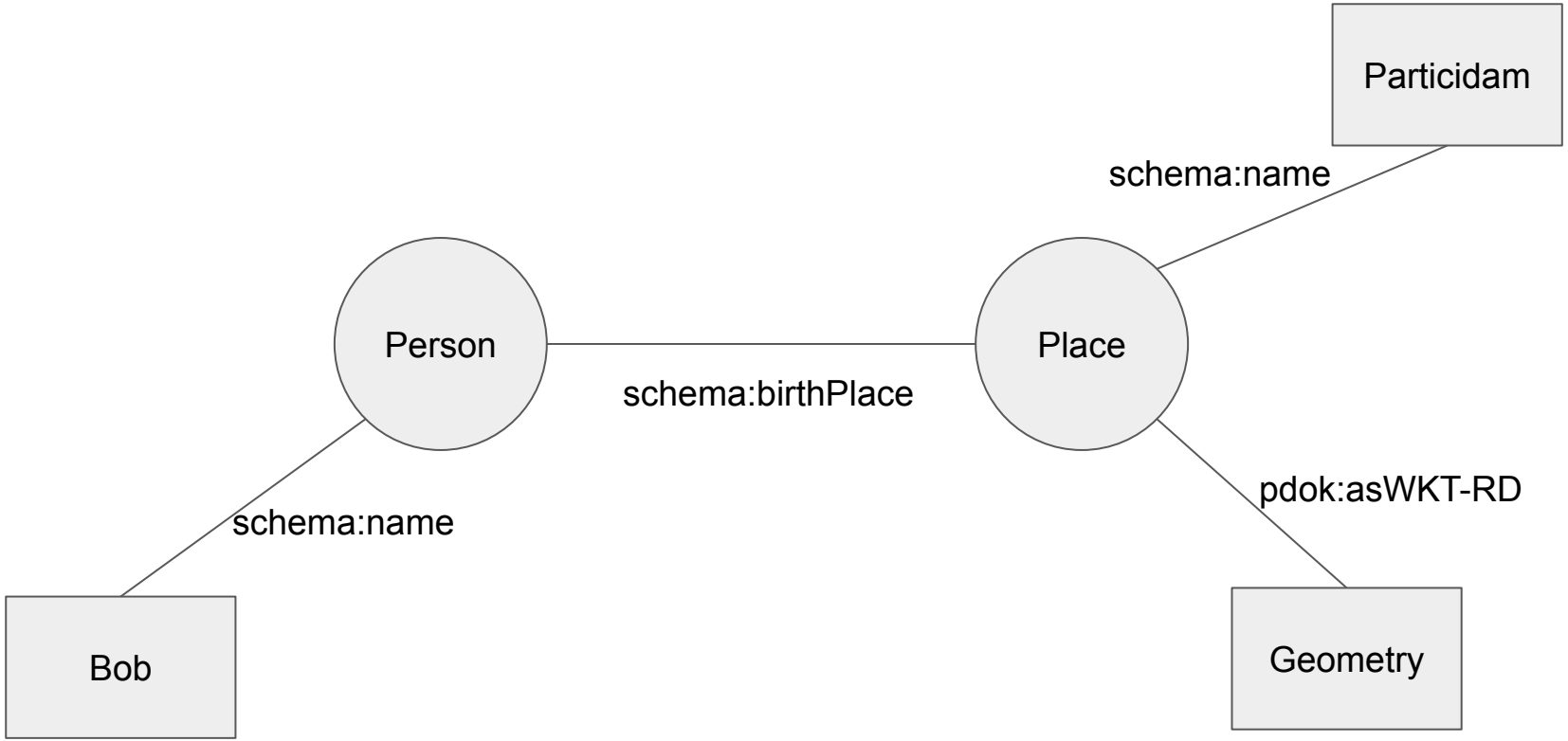
- Embedding Linked Data into your existing app
- SOLID apps
- Games
- Multi-domain linked-data browsers
- Full Rest-in-Rest hypermedia systems

- Open- and closed-world compatible

Ontola.io

Rendering

Rendering data



Rendering data

```
import schema from “@ontologies/schema”

const MyPlaceView = ({ geometry, name }) => (
  <div>
    <h1>{name.value}<h1>
    <GeometryViewer shape={geometry} >
  </div>
)

MyPersonView.mapDataToProps = {
   geometry: new NamedNode(‘http://data.pdok.nl/def/pdok#asWKT-RD’),
   name: schema.name
}
```

Ontola.io

Rendering data

```
import schema from “@ontologies/schema”

const MyPersonView = ({ birthPlace, name }) => (
  <div>
    <h1>{name.value}<h1>
    <MyPlaceView place={birthPlace}>
  </div>
)

MyPersonView.type = schema.Person


MyPersonView.mapDataToProps = {
   birthPlace: schema.birthPlace,
   name: schema.name
}
```

Ontola.io

Rendering data

```
import schema from “@ontologies/schema”

const MyPersonView = ({ birthPlace, name }) => (
  <div>
    <h1>{name.value}<h1>
    <LinkedResourceContainer subject={birthPlace}>
  </div>
)

MyPersonView.type = schema.Person


MyPersonView.mapDataToProps = {
   birthPlace: schema.birthPlace,
   name: schema.name
}
```

Ontola.io

Rendering data - open world assumption

```
import schema from “@ontologies/schema”

const MyPersonView = () => (
  <div>
    <Property label={schema.name} />
    <Property label={schema.birthPlace} />
  </div>
)

MyPersonView.type = schema.Person
```

Ontola.io

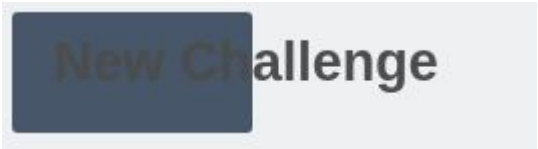
Rendering data - open world assumption

```
import schema from “@ontologies/schema”

const MyNameView = ({ linkedProp }) => (
  <h1>{linkedProp.value}</h1>
)

MyNameView.type = schema.Thing

MyNameView.property = schema.name
```

Ontola.io

Rendering data - Constraints

```
<ButtonComponent>
  <Property label={schema.name} />
</ButtonComponent>
```



Ontola.io

Rendering data - Constraints

```
const ButtonComponent = ({ children }) => (
  <button className=”pretty”>
    {children}
  </button>
)
```

Ontola.io

Rendering data - Constraints

```
const ButtonComponent = ({ children }) => (
  <button className=”pretty”>
    {children}
  </button>
)

ButtonComponent.providesTopology = <example:topology/button>
```

(simplified)

Ontola.io

Rendering data - Smarts

<foaf:name> <owl:sameAs> <schema:name>

<vcard:name> <owl:sameAs> <foaf:name>

Future:

<dcterms:label> <skos:closeMatch> <foaf:name>

Hypermedia determined rendering

Rendering data - Constraints

```
import { TopologyProvider } from “link-redux”

const buttonTopology = <example:topology/button>

class ButtonComponent extends TopologyProvider {
  constructor() {
    this.element = “button”
    this.className = “pretty”
    this.topology = buttonTopology
  }
}
```

Ontola.io

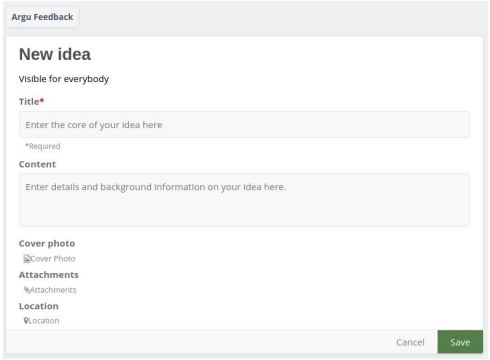
Rendering data - Topology

```
const MyNameButtonView = ({ linkedProp }) => (
  <p>{linkedProp.value}</p>
)

MyNameButtonView.property = schema.name

MyNameButtonView.topology = <example:topology/button>
```

Ontola.io

Topology

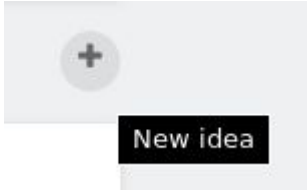
New action; one resource, many representations



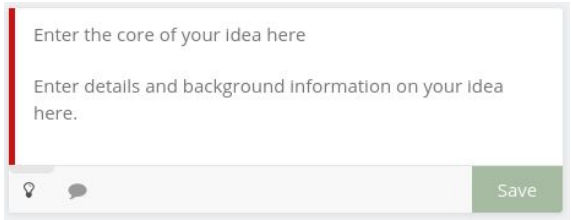
argu:page

(form argu:cardMain)

argu:widget



argu:containerFloat



argu:omniformFields

(form argu:omniformFields)

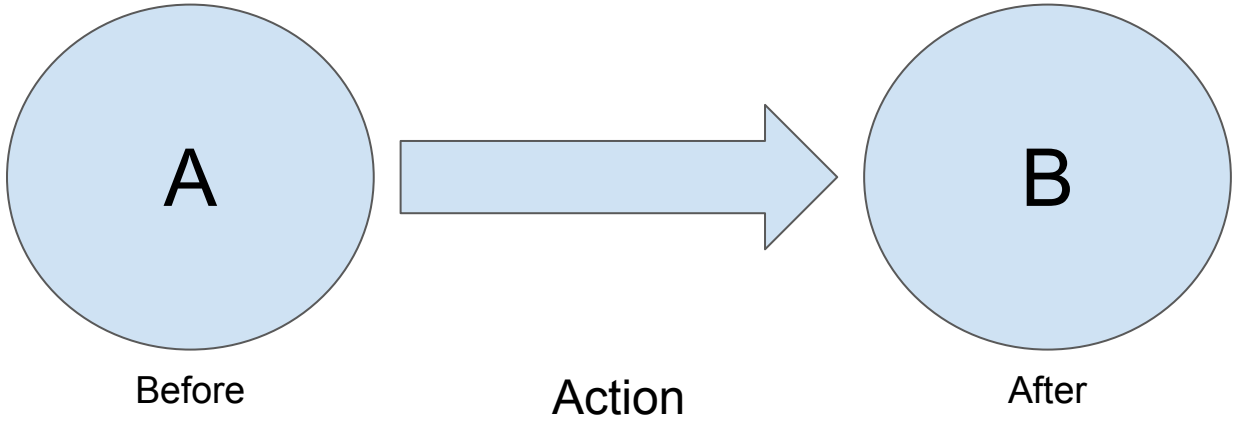
Ontola.io

Topology

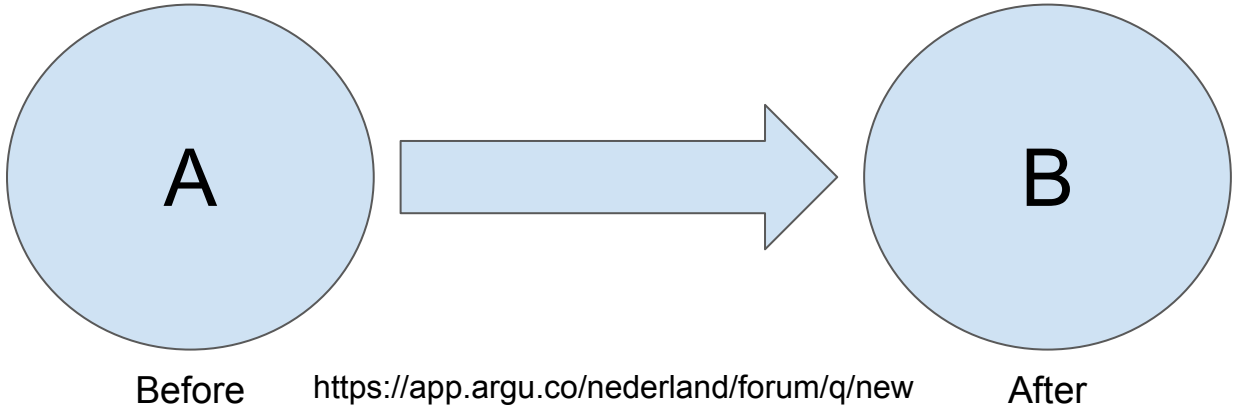
- Define the constraints the component should adhere to
- High coupling to UI framework
- Resources
  - Possible inheritance
  - Shareable

Actions

Change State



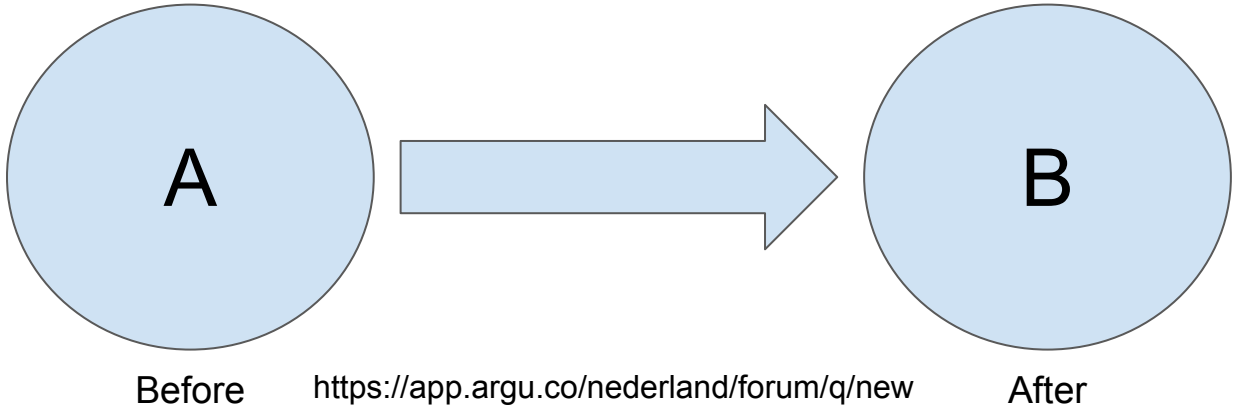
Actions are Resources



argu:nederland/forum/q/new a schema:CreateAction, ontola:Create::Question ;
schema:name “Nieuwe uitdaging” ;
schema:result argu:Question ;
schema:httpMethod “POST” ;
linklib:actionBody <argu:nederland/forum/q/new#shape> .

Ontola.io

Actions are Executable




```
const actionIRI = new NamedNode(“https://app.argu.co/nederland/forum/q/new”)

const actionBody = [Statement[], File[]]


exec(actionIRI, actionBody)
```

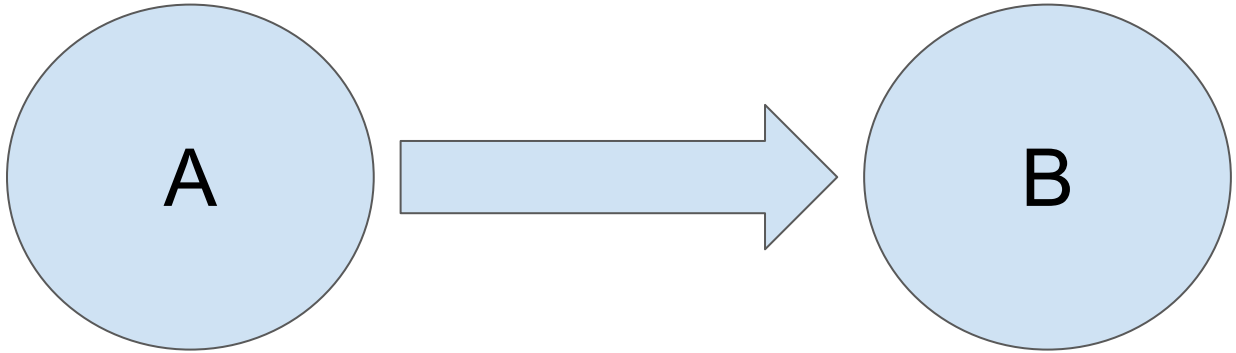
Ontola.io

Middleware

```
(store) => (next) => (iri: NamedNode, opts: any) => {
  if (iri === myAction) {
    // Todo: Manipulate the store
    return store.actions.solid.save(resource)
  }
  return next(iri, opts)
}
```

Ontola.io

The action can bee seen as the difference in state



$$A + \Delta = B$$

Solve Δ

Ontola.io

Solving Δ

- Usually done by some application logic
- Traditionally done by the server
- SOLID moves this to the client (where possible)

Linked Delta

One interface for changing state;

- Usable by servers
- Usable by clients

```
<argu:q/5> <rdf:type>        <argu:Question>   <ld:add>
<argu:q/5> <schema:name>     “Title”           <ld:replace>
<argu:q/5> <schema:creator>  <argu:fletcher91> <ld:replace>
<argu:q>   <sp:Variable>     <sp:Variable>     <ld:purge>
```

In practice: Open Raadsinformatie parses these to fill the REST API and elastic store

Ontola.io

How to get started?

Building SOLID/Linked Data apps

- Add it to your existing React App
  - https://github.com/fletcher91/link-redux
- Clone and customize the Mash Browser
  - https://github.com/ontola/mash
- Use the boilerplate
  - https://github.com/ontola/link-solid-boilerplate

Building a RESTful linked data service

- Linked Rails
  - Rails with RDF serialization, schema:Actions, SHACL forms, ActivityStreams Collections
  - https://github.com/ontola/linked_rails

Ontola.io

Future

- Deep queries
  - SPARQL
  - REST
  - Triple Pattern Fragments
  - Custom extensions
- Context aware view lookups
- Improved developer experience
  - Expanding the available @ontologies packages
  - Streamline boilerplate
  - React: better naming & more hooks
  - Additional documentation
- Server-side rendering
  - Already possible, but has some stability issues.

Ontola.io

Questions

Slides: http://bit.ly/solid-link

Resources (1/2)

- Slides
  - http://bit.ly/solid-link
- Data browser
  - https://ontola-mash.herokuapp.com/
  - https://github.com/ontola/mash
- Link Reference
  - https://github.com/fletcher91/link-redux/wiki
  - https://github.com/ontola/link-devtools
- Ontologies
  - @ontologies/core on NPM
    - Available: rdf, rdfs, schema, dcterms, as, foaf, owl, shacl, prov, xsd
- Example apps
  - https://github.com/fletcher91/link-minesweeper
  - https://github.com/fletcher91/link-redux-todo

Ontola.io

Resources (2/2)

- Linked Delta
  - Spec: https://github.com/ontola/linked-delta
  - JavaScript: https://github.com/fletcher91/link-lib/blob/master/src/store/deltaProcessor.ts
  - Kotlin: https://github.com/ontola/ori_api/tree/master/src/main/kotlin/io/ontola/linkeddelta
- Linked Rails (Create Linked Data apps in Rails)
  - https://github.com/ontola/linked_rails

Ontola.io