GMap3 Wicket Component

Table of Contents

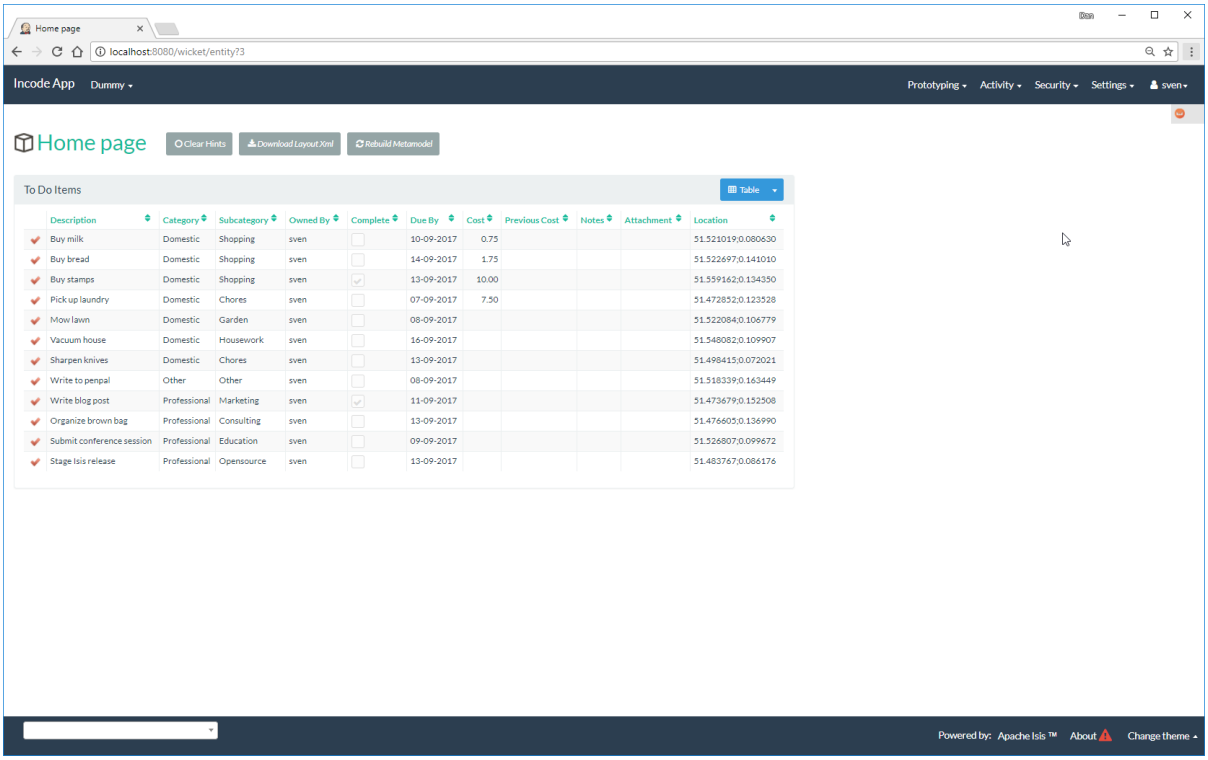
This component (`isis-wicket-gmap3`) allows an entity or collection of entities to be rendered within a map (using google's gmap3 API).

Screenshots

The module's functionality can be explored by running the quickstart with example usage using the `org.incode.domainapp.example.app.modules.ExampleDomWktGmap3AppManifest`.

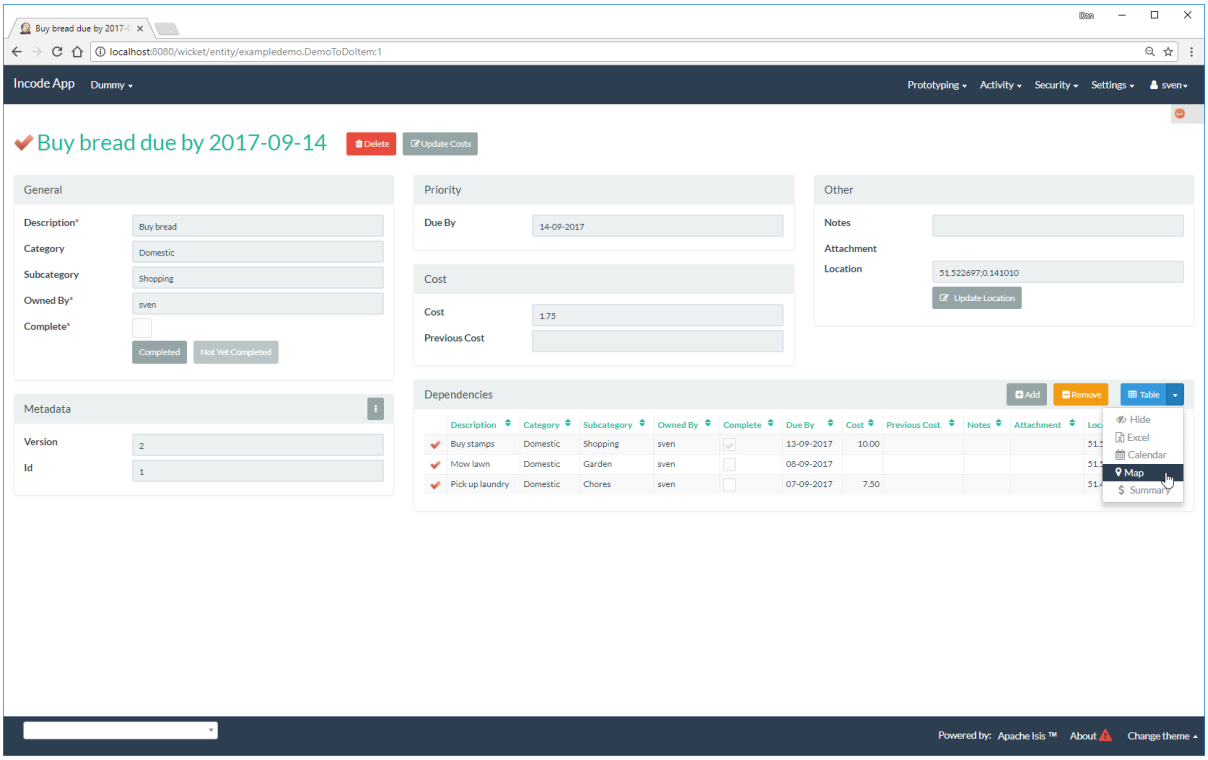
Note that the `isis.viewer.wicket.gmap3.apiKey` must be set to a valid value; this is most easily done using a system property.

A home page is displayed when the app is run:

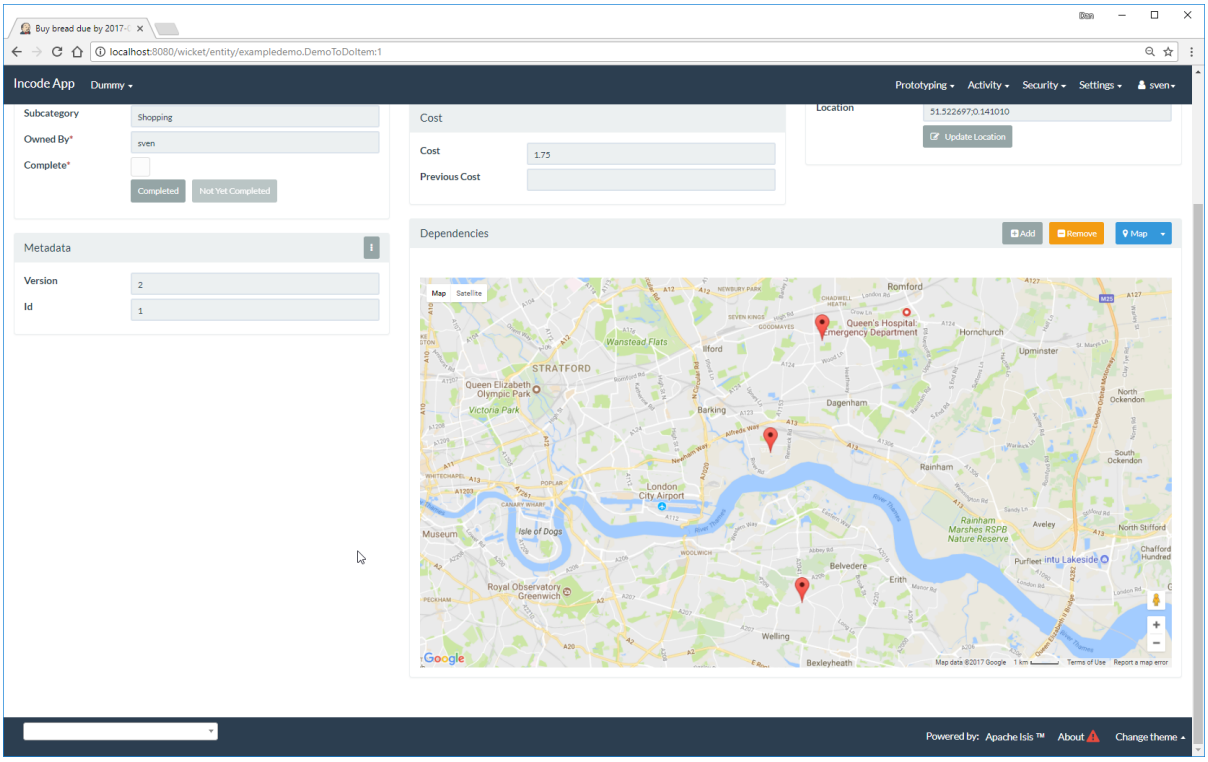


Parented collection as gmap

The todo item's collection contains a list of `Locatable` entities (also todo items); this is indicated through a button to switch the view:



Clicking the button shows the same entities on a gmap3:

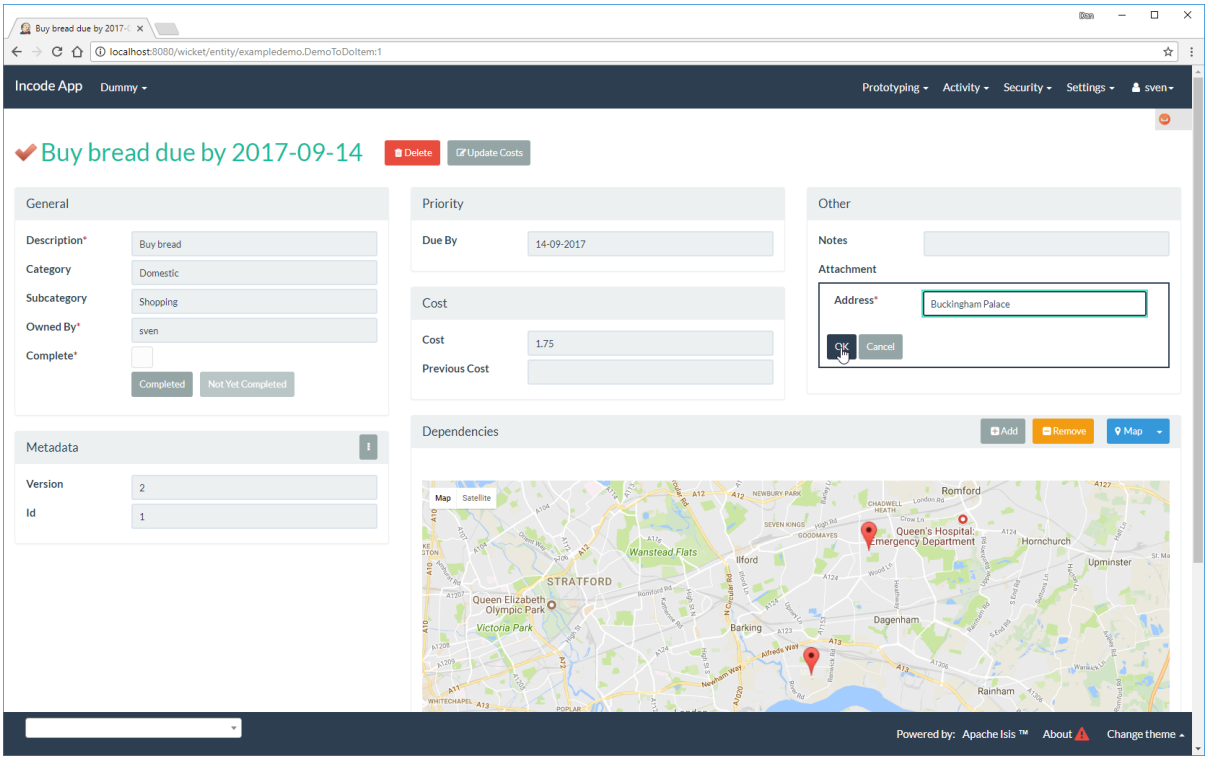


Update location using service

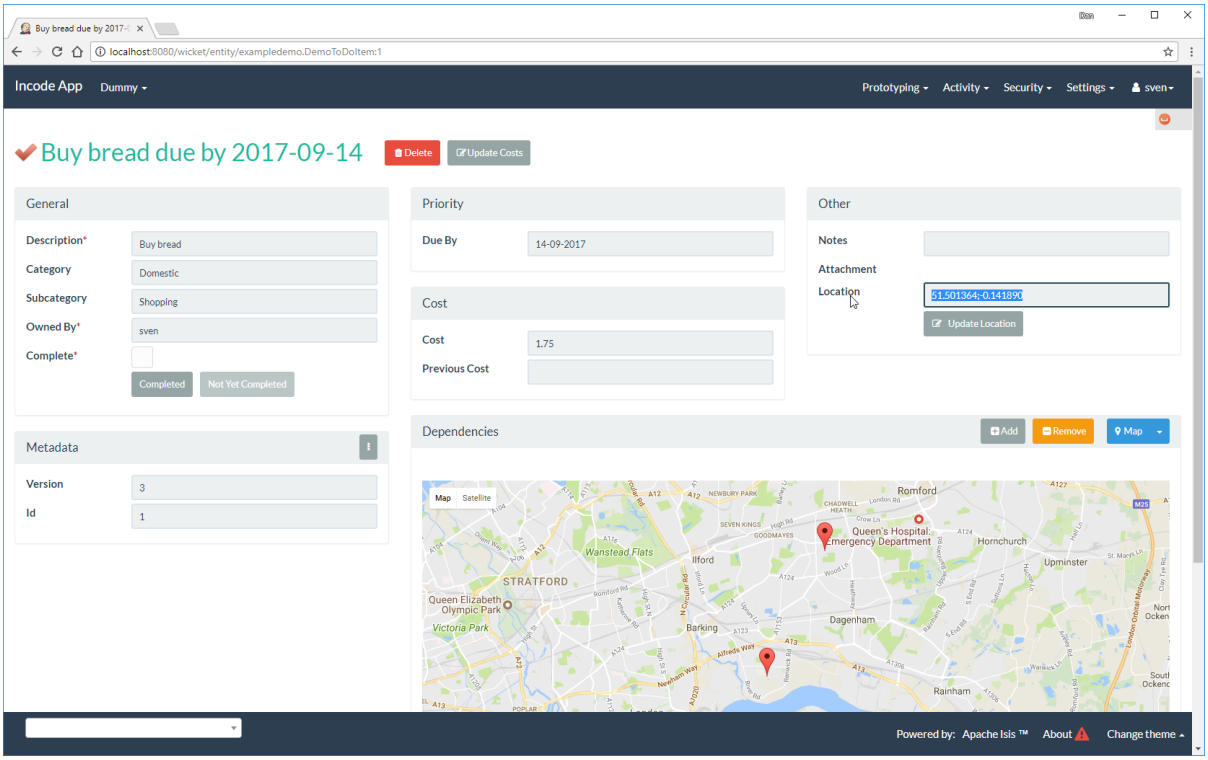
The previous screenshot shows the todo item also provides an "update location" action:

```
public Gmap3WicketToDoItem updateLocation(
        @ParameterLayout(named="Address")
        final String address) {
    final Location location = this.locationLookupService.lookup(address);
    setLocation(location);
    return this;
}
```

When invoked:

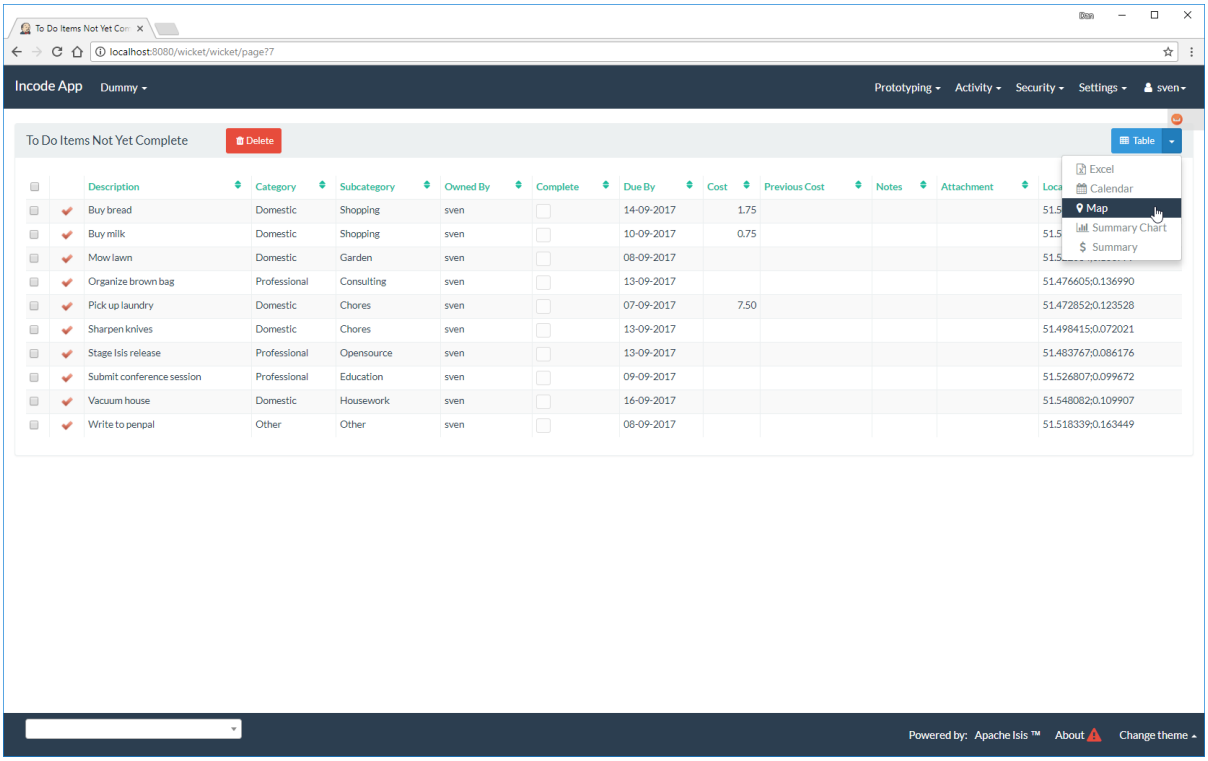


... it updates the location:

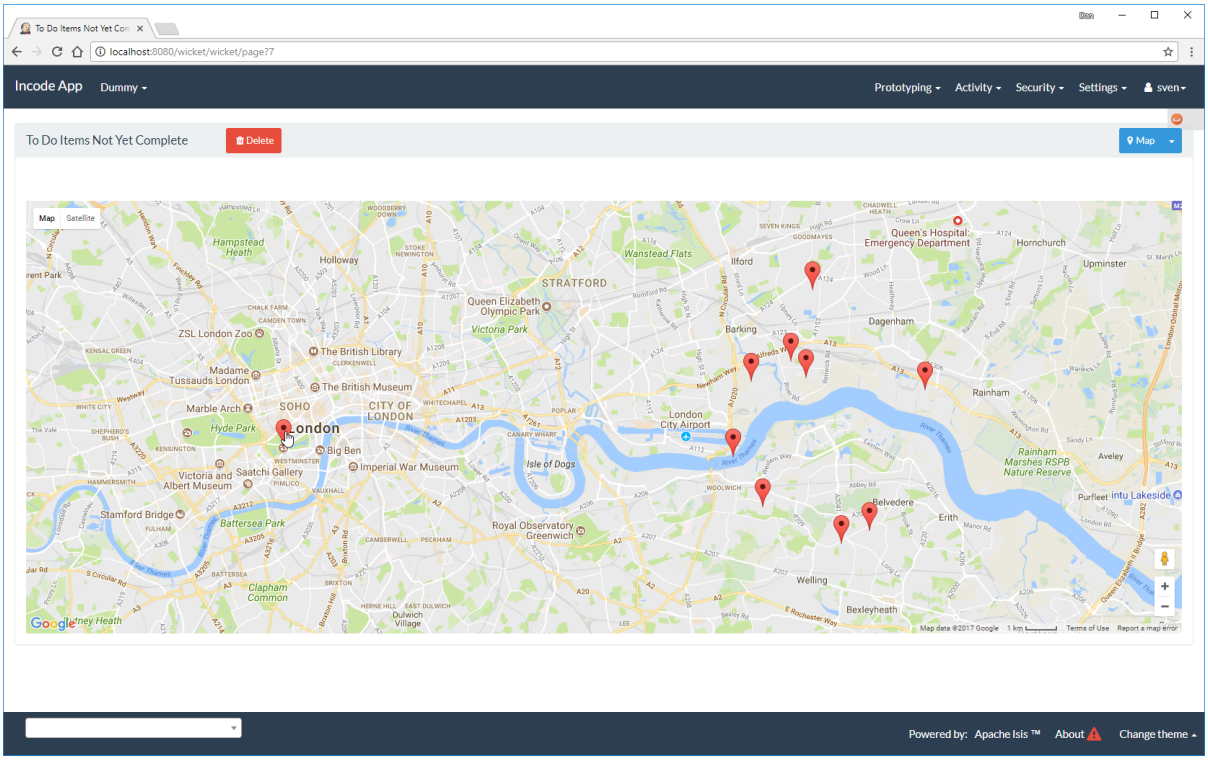


Standalone location as gmap

Invoking an action that returns a list of `Locatable` entities also results in the button to view in a gmap3:

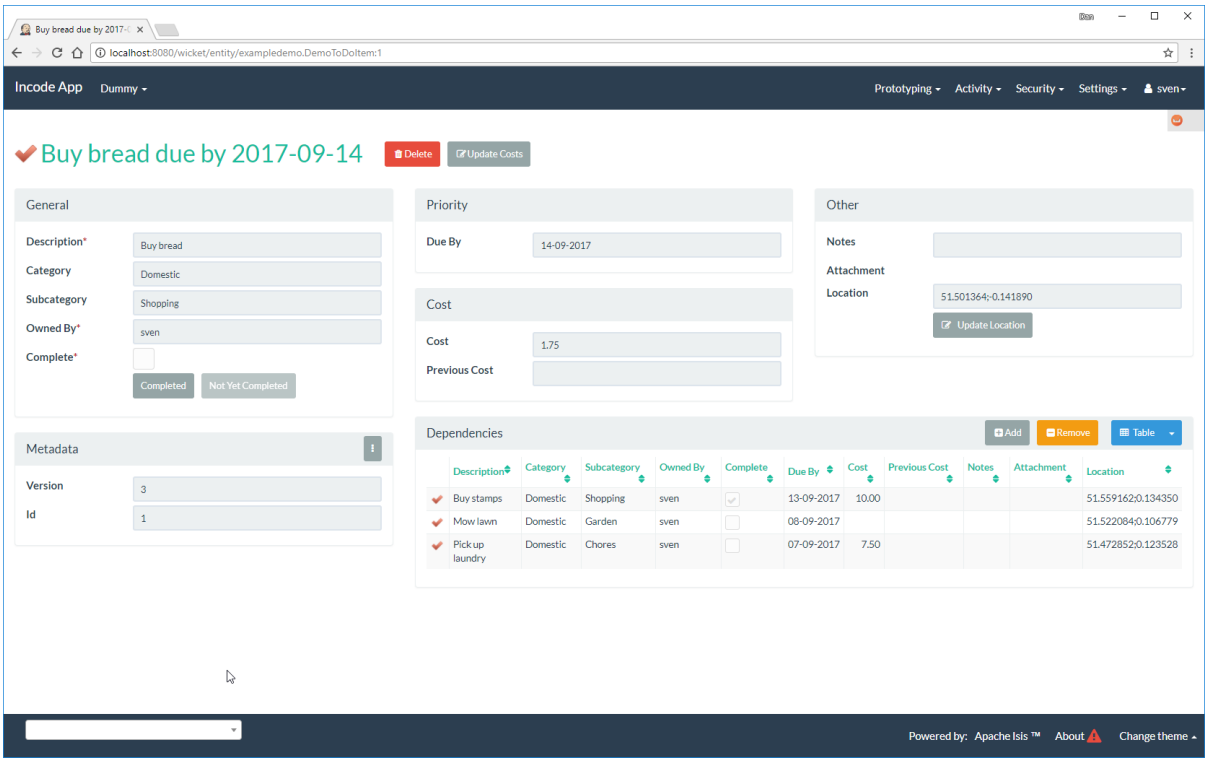


... which then renders the items in a map. Note the tooltips:



Click through

Clicking on a map marker drills down to the entity:



API & Usage

Rendering objects on a map

Make your entity implement `org.isisaddons.wicket.gmap3.applib.Locatable`, such that it provides a `Location` property of type `org.isisaddons.wicket.gmap3.applib.Location`.

This property will need to be annotated as `@javax.jdo.annotations.Persistent`.

For example:

```
import org.isisaddons.wicket.gmap3.cpt.applib.Locatable;
import org.isisaddons.wicket.gmap3.cpt.applib.Location;

public class ToDoItem implements Locatable {
    ...
    @javax.jdo.annotations.Persistent
    private Location location;

    @MemberOrder(name="Detail", sequence = "10")
    @Optional
    public Location getLocation() {
        return location;
    }
    public void setLocation(Location location) {
        this.location = location;
    }
}
```

You should then find that any collections of entities that have `Locatable` properties (either returned from an action, or as a parented collection) will be rendered in a map.

`LocationLookupService`

By injecting the provided `LocationLookupService` domain service, you can write an action to lookup `Location`s.

For example:

```
public void lookupLocation(
        @ParameterLayout(named="Description")
        final String description) {
    setLocation(locationLookupService.lookup(description));
}
```

To use this the `LocationLookupService` needs to be registered; see below.

Alternatively, the `Location` can also be specified directly as a string. The format is `mmm.mmm;nnn.nnn`, where `mmm.mmm` is the latitude, and `nnn.nnn` is the longitude

LocationDereferencingService

Sometimes the domain object that implements `Locatable` will be a supporting object such as an `Address`, belonging to a `Customer`, say. When the location marker is clicked in the map, we would rather that the UI opens up the `Customer` rather than the associated `Address` (in other words, saving a click).

This requirement is supported by providing an implementation of the `LocationDereferencingService`:

```
public interface LocationDereferencingService {
    @Programmatic
    Object dereference(final Object locatable);
}
```

for example, one might have:

```
public class LocationDereferencingServiceForAddress implements
LocationDereferencingService {
    @Programmatic
    public Object dereference(final Object locatable) {
        if (!(locatable instanceof Address)) {
            return null;
        }
        final Address address = (Address) locatable;
        return address.getCustomer();
    }
}
```

Note that there can be multiple implementations of this service; the component will check all that are available. The order in which they are checked depends upon the `@DomainServiceLayout(menuOrder=…)` attribute.

How to configure/use

Classpath

Add the component to your project's `dom` module's `pom.xml`:

```
<dependency>
    <groupId>org.isisaddons.wicket.gmap3</groupId>
    <artifactId>isis-wicket-gmap3-cpt</artifactId>
    <version>1.14.0</version>
</dependency>
```

Check for later releases by searching Maven Central Repo.

For instructions on how to use the latest `-SNAPSHOT`, see the contributors guide.

Bootstrapping

In the `AppManifest`, update its `getModules()` method, eg:

```
@Override
public List<Class<?>> getModules() {
    return Arrays.asList(
            ...
            org.isisaddons.wicket.gmap3.cpt.applib.Gmap3ApplibModule.class,
            org.isisaddons.wicket.gmap3.cpt.service.Gmap3ServiceModule.class,
    );
}
```

This will register the `LocationLookupService`.

Configuration Properties

gmap3 API Key

In order to use the component an API key is required. See the google documentation for instructions as to how to do this; a free key (with quite generous daily limits) can be used.

Configure the key in `WEB-INF/viewer_wicket.properties` (or `WEB-INF/isis.properties`):

```
isis.viewer.wicket.gmap3.apiKey=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX-XXXXXX
```

Known issues

None known at this time.

Dependencies

Maven can report modules dependencies using:

```
mvn dependency:list -o -pl modules/wkt/gmap3/impl -D excludeTransitive=true
```

which, excluding Apache Isis itself, returns these compile/runtime dependencies:

```
org.jdom:jdom:jar:2.0.2
org.wicketstuff:wicketstuff-gmap3:jar:7.8.0
org.apache.httpcomponents:httpclient:jar:4.5.2
```

For further details on 3rd-party dependencies, see:

- 42Lines/wicket-fullcalendar

In addition to Apache Isis, this component depends on:

- wicketstuff/core (gmap3 component)

  which integrates the Google Maps Javascript API

- JDOM
- Apache HttpComponents