FullCalendar2 Wicket Component

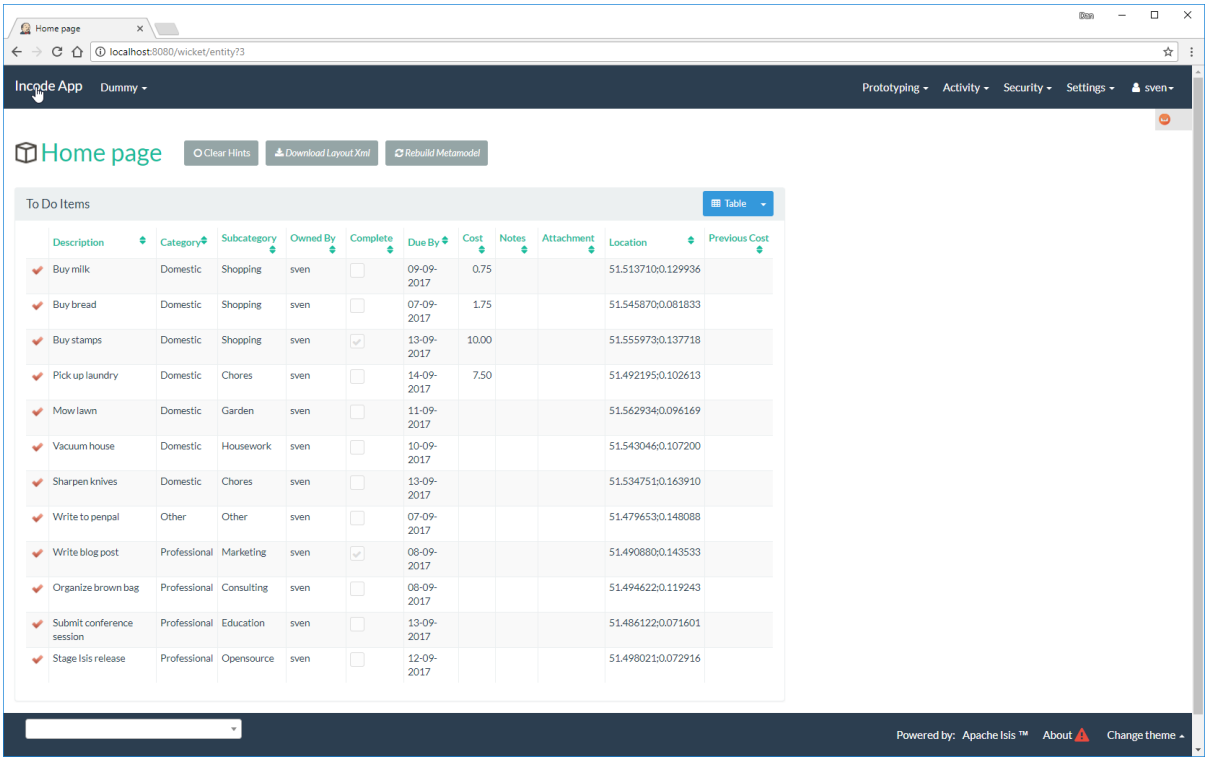
Table of Contents

This component (`isis-wicket-fullcalendar2`) renders events for a collection of entities within a fullpage calendar. Underneath the covers it uses this fullcalendar widget.

Screenshots

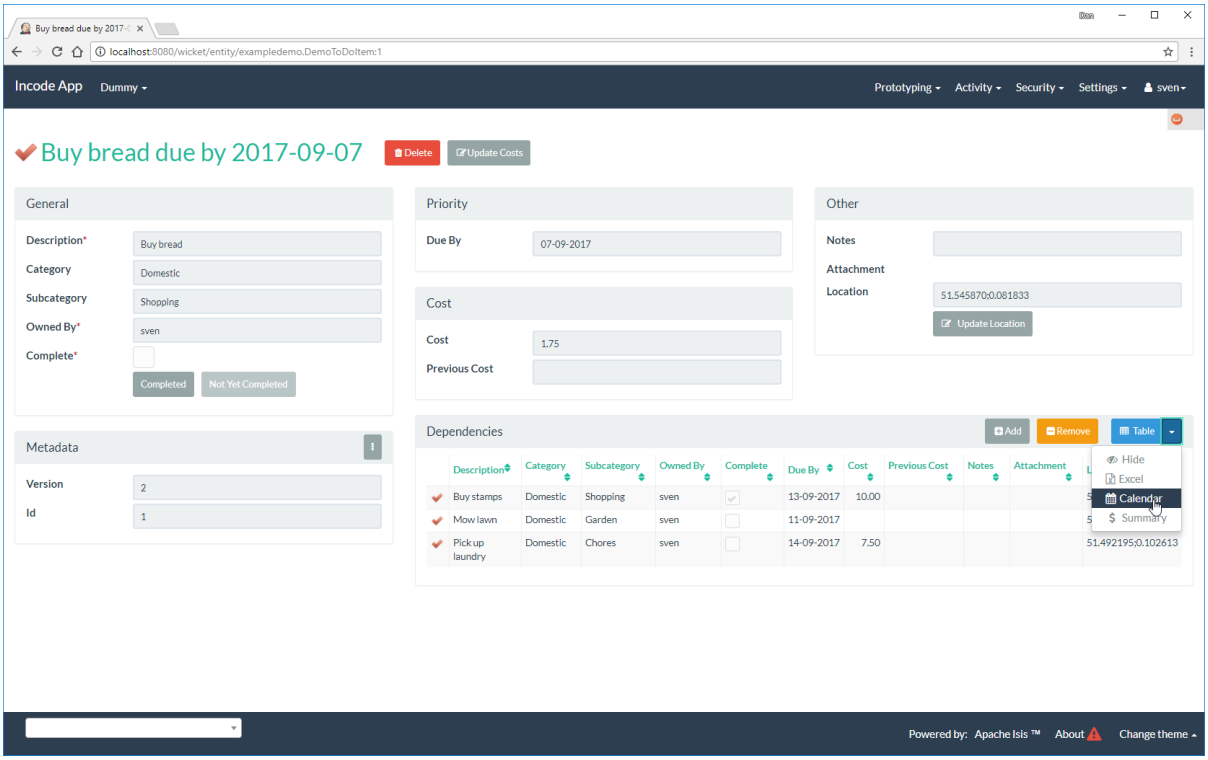
The module's functionality can be explored by running the quickstart with example usage using the `org.incode.domainapp.example.app.modules.ExampleDomWktFullCalendar2AppManifest`.

A home page is displayed when the app is run:

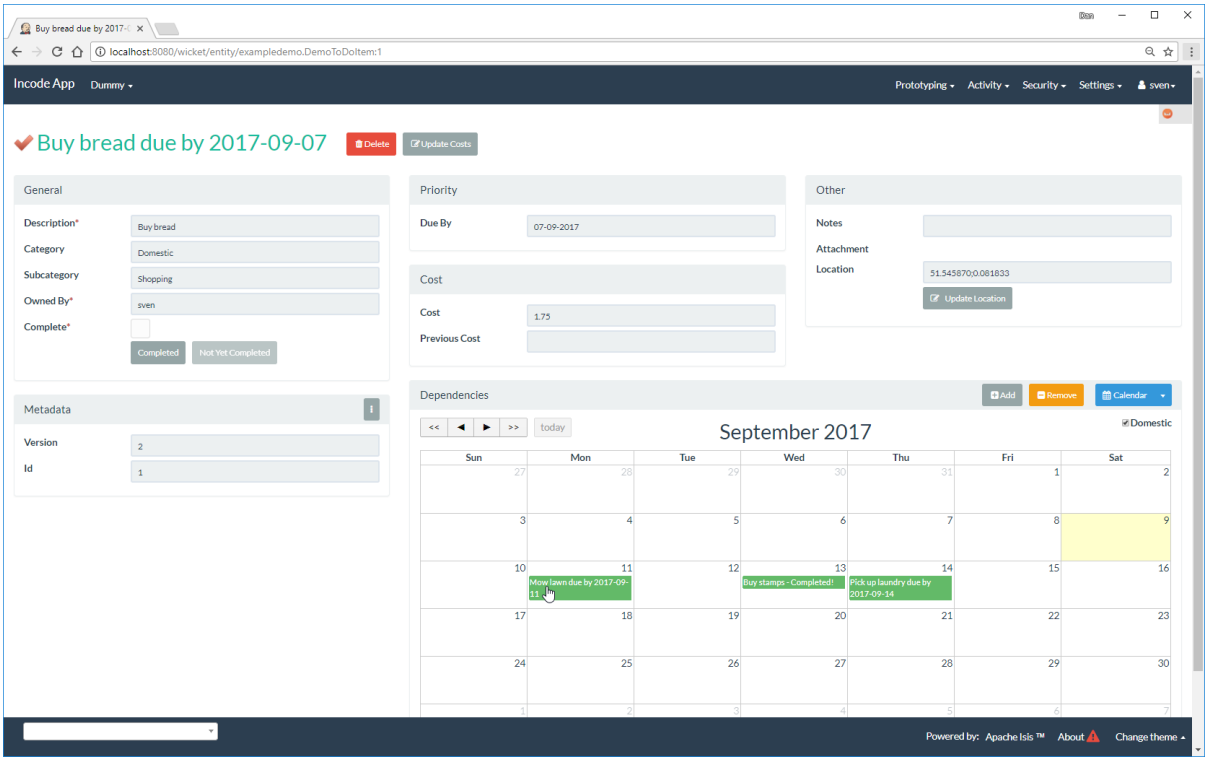


Parented collection as calendar

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

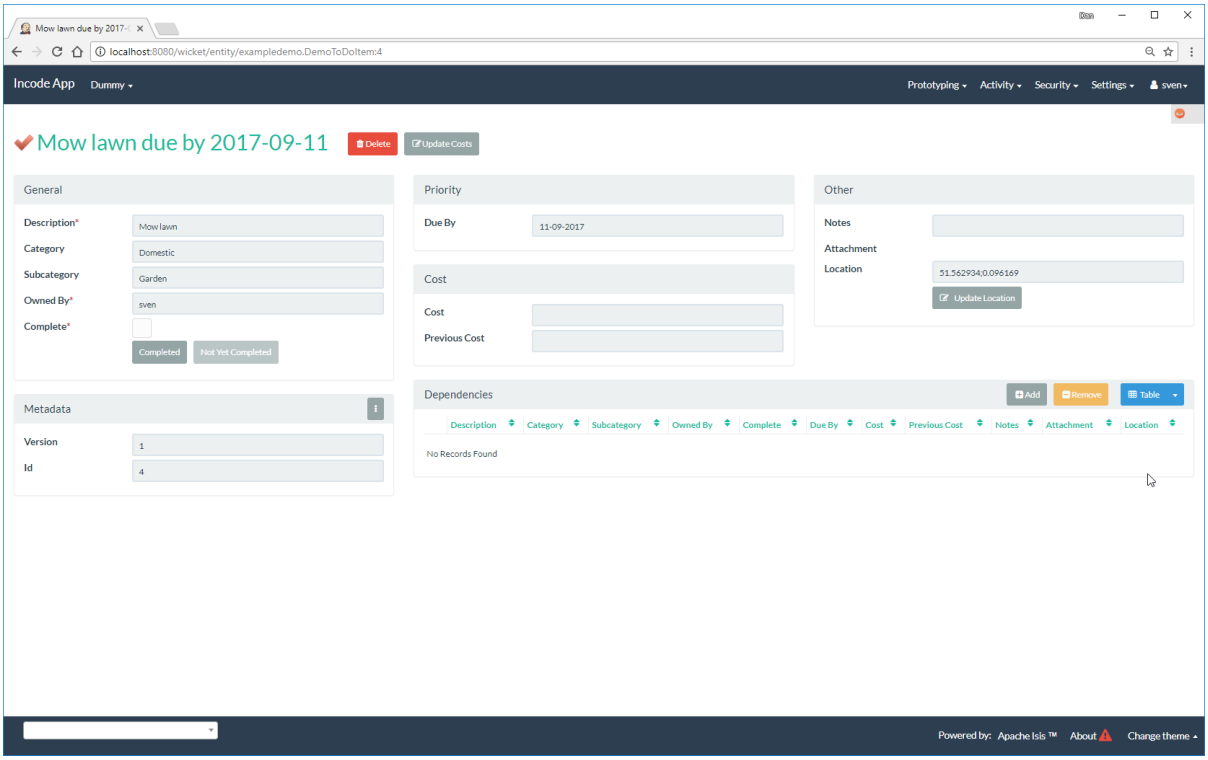


Clicking the button shows the same entities on a fullpage calendar:



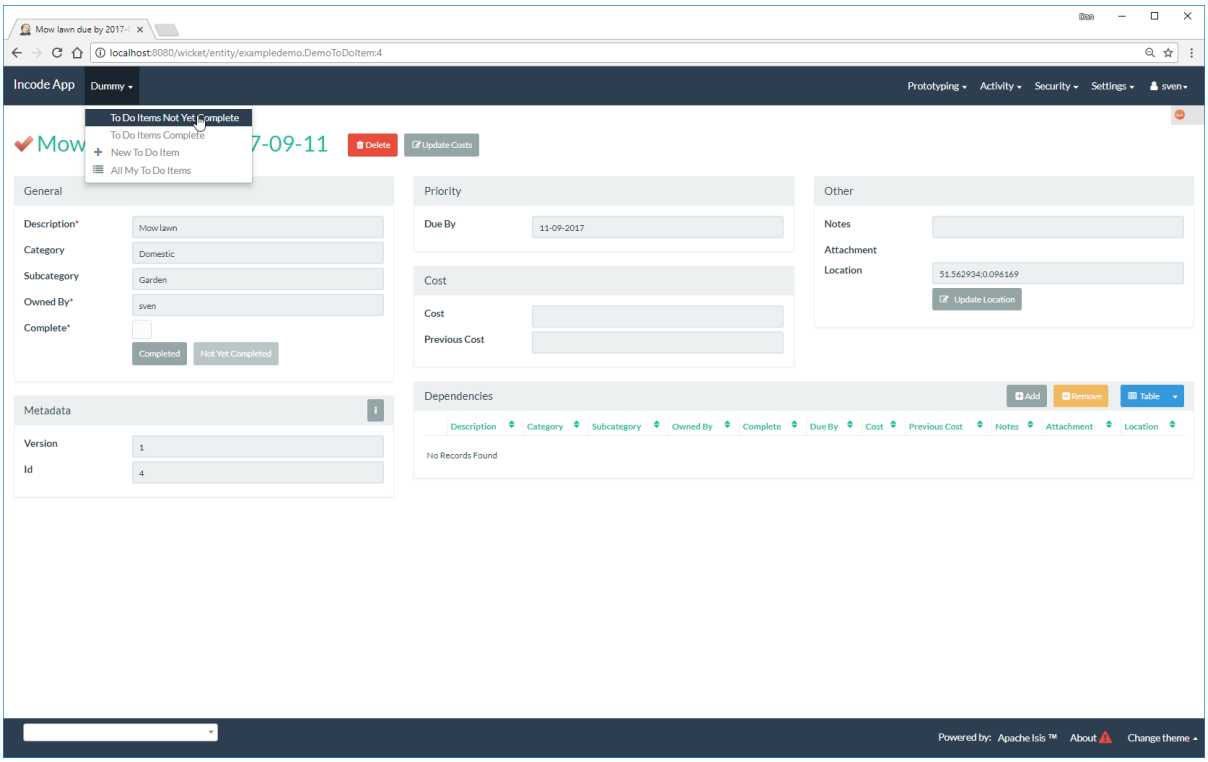
Drill down

Clicking on the event in the calendar drills down to the corresponding entity:

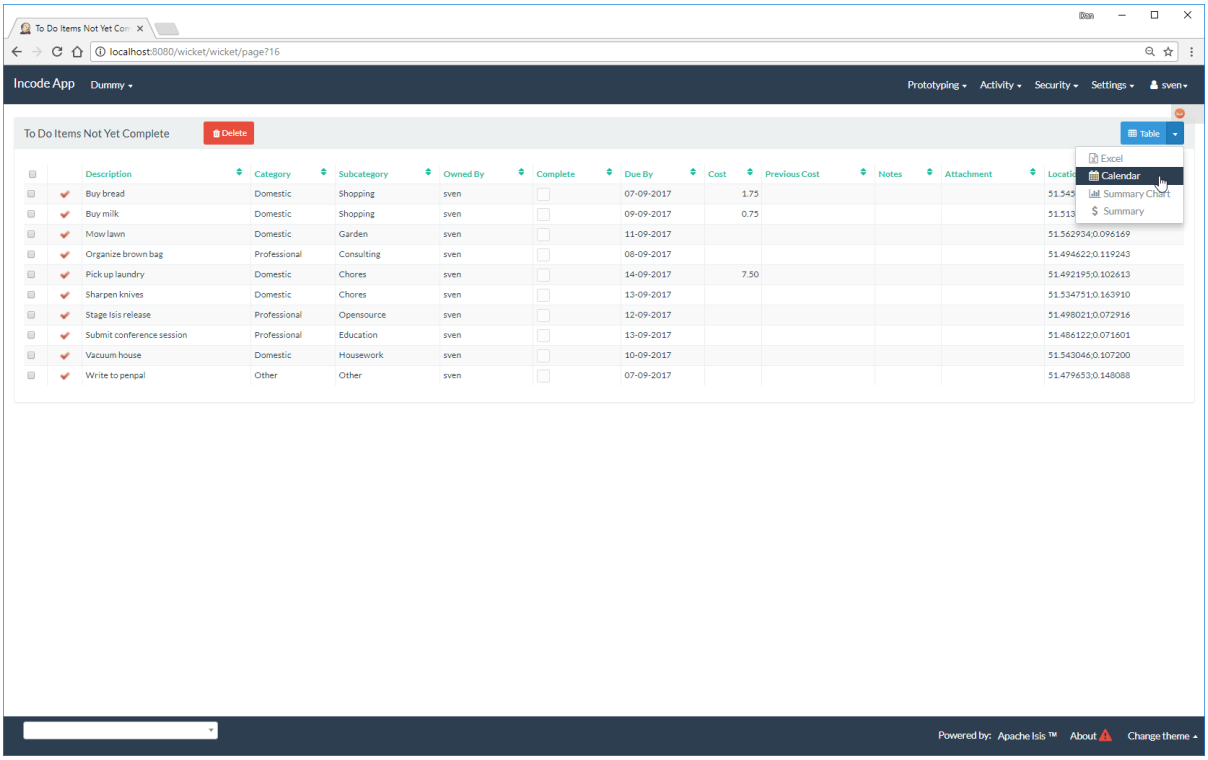


Standalone collection as calendar

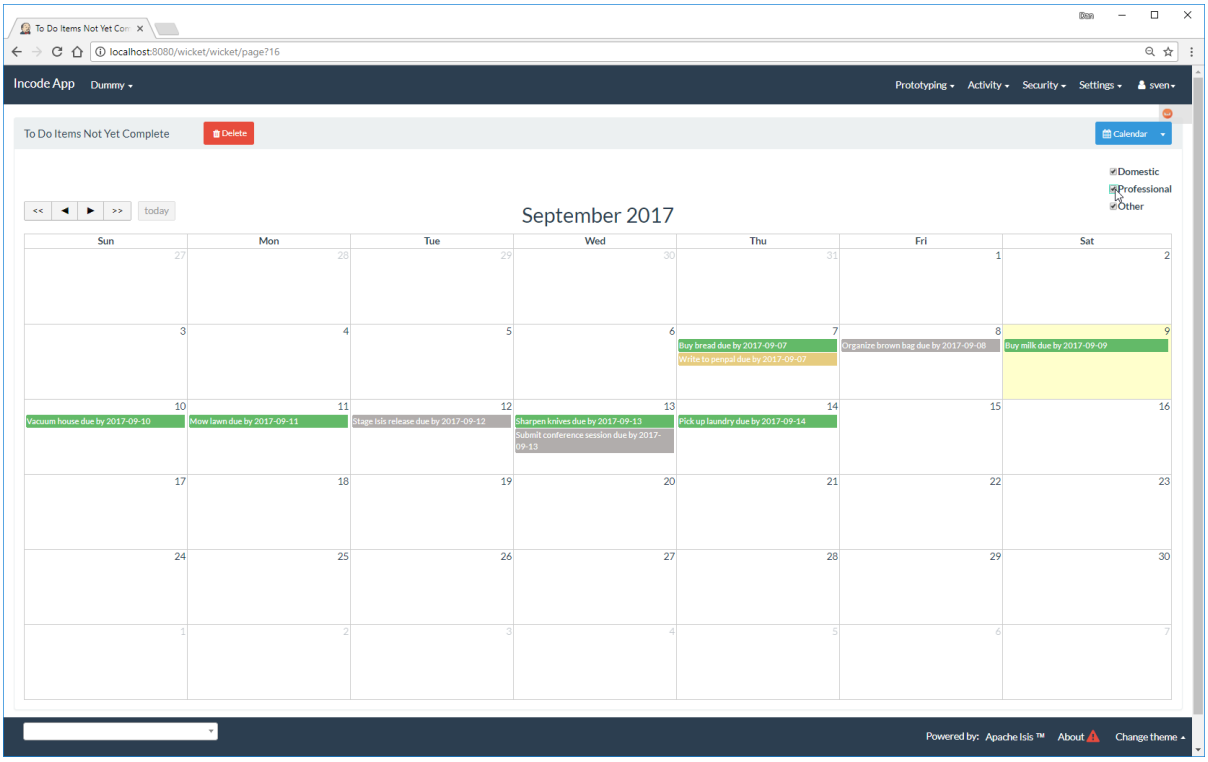
Invoking an action that returns a list of `Calendarable` entities:



... also results in the button to view in a fullpage calendar:



Each item is shown in the calendar view:

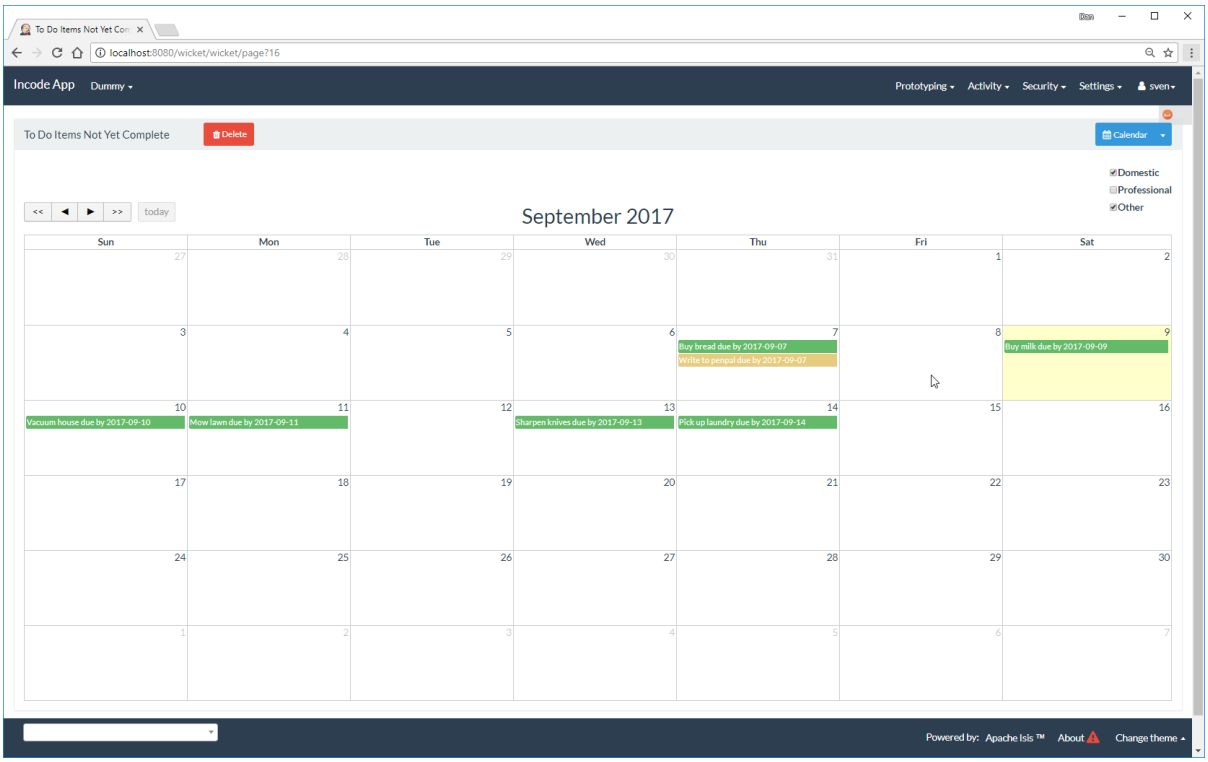


Calendars

Each entity can provides dates to either a single calendar or to multiple calendars. In the example app each todo item exposes its `dueBy` date to a single calendar, named after its `category`:

```
@Programmatic
@Override
public String getCalendarName() {
    return getCategory().name();
}
@Programmatic
@Override
public CalendarEvent toCalendarEvent() {
    if(getDueBy() == null) {
        return null;
    }
    return new CalendarEvent(getDueBy().toDateTimeAtStartOfDay(), getCalendarName(),
container.titleOf(this));
}
```

The full page calendar uses colour coding to indicate the calendars, as well as checkboxes to show/hide each calendar. Unchecking the calendar toggle hides all events in that calendar:



API & Usage

Each entity must implement either the `CalendarEventable` interface or the `Calendarable` interface:

CalendarEventable` interface

Of the two interfaces, `CalendarEventable` interface is the simpler, allowing the object to return a single `CalendarEvent`:

```
public interface CalendarEventable {
    String getCalendarName();           ①
    CalendarEvent toCalendarEvent();    ②
}
```

① groups similar events together; in the UI these correspond to checkboxes rendered near the top.

② returns a `CalendarEvent` value type representing the data to be rendered on the calender.

`CalendarEvent` itself is:

```
public class CalendarEvent implements Serializable {
    private final DateTime dateTime;
    private final String calendarName;
    private final String title;
    private final String notes;
    public CalendarEvent(
            final DateTime dateTime,
            final String calendarName,
            final String title) {
        this(dateTime, calendarName, title, null);
    }
    public CalendarEvent(
            final DateTime dateTime,
            final String calendarName,
            final String title,
            final String notes) {
        this.dateTime = dateTime;
        this.calendarName = calendarName;
        this.title = title;
        this.notes = notes;
    }
    ...
}
```

In the demo app, the `ToDoItem` implements `CalendarEventable`.

`Calendarable` interface

While the `CalendarEventable` interface will fit many requirements, sometimes an object will have several dates associated with it. For example, one could imagine an object with start/stop dates, or optionExercise/optionExpiry dates.

The `Calendarable` interface therefore allows the object to return a number of `CalenderEvent`s; each is qualified (identified) by a `calendarName`:

```
public interface Calendarable {
    Set<String> getCalendarNames();
    ImmutableMap<String, CalendarEventable> getCalendarEvents();
}
```

`CalendarableDereferencingService`

Sometimes the domain object that implements `Calendarable` or `CalendarEventable` will be a supporting object such as a `Note` attached to an `Order`, say. When the marker is clicked in the calendar, we would rather that the UI opens up the `Order` rather than the associated `Note` (in other words, saving a click).

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

```
public interface CalendarableDereferencingService {
    @Programmatic
    Object dereference(final Object calendarableOrCalendarEventable);
}
```

for example, one might have:

```
public class LocationDereferencingServiceForNote implements
CalendarableDereferencingService {
    @Programmatic
    public Object dereference(final Object calendarableOrCalendarEventable) {
        if (!(locatable instanceof Note)) {
            return null;
        }
        final Note note = (Note) calendarableOrCalendarEventable;
        return note.getOwner();
    }
}
```

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 this component to your project's `dom` module's `pom.xml`, eg:

```
<dependency>
    <groupId>org.isisaddons.wicket.fullcalendar2</groupId>
    <artifactId>isis-wicket-fullcalendar2-cpt</artifactId>
    <version>1.15.1.1</version>
</dependency>
```

Check for later releases by searching Maven Central Repo.

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

Known issues

None known at this time.

Dependencies

Maven can report modules dependencies using:

```
mvn dependency:list -o -pl modules/wkt/fullcalendar2/impl -D excludeTransitive=true
```

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

```
net.ftlines.wicket-fullcalendar:wicket-fullcalendar-core:jar:2.2.0
```

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

- 42Lines/wicket-fullcalendar

  In turn, this uses Javascript components:

  - http://arshaw.com/fullcalendar/ (MIT License)
  - http://jquery.com (MIT License)