SessionLogger SPI Implementation

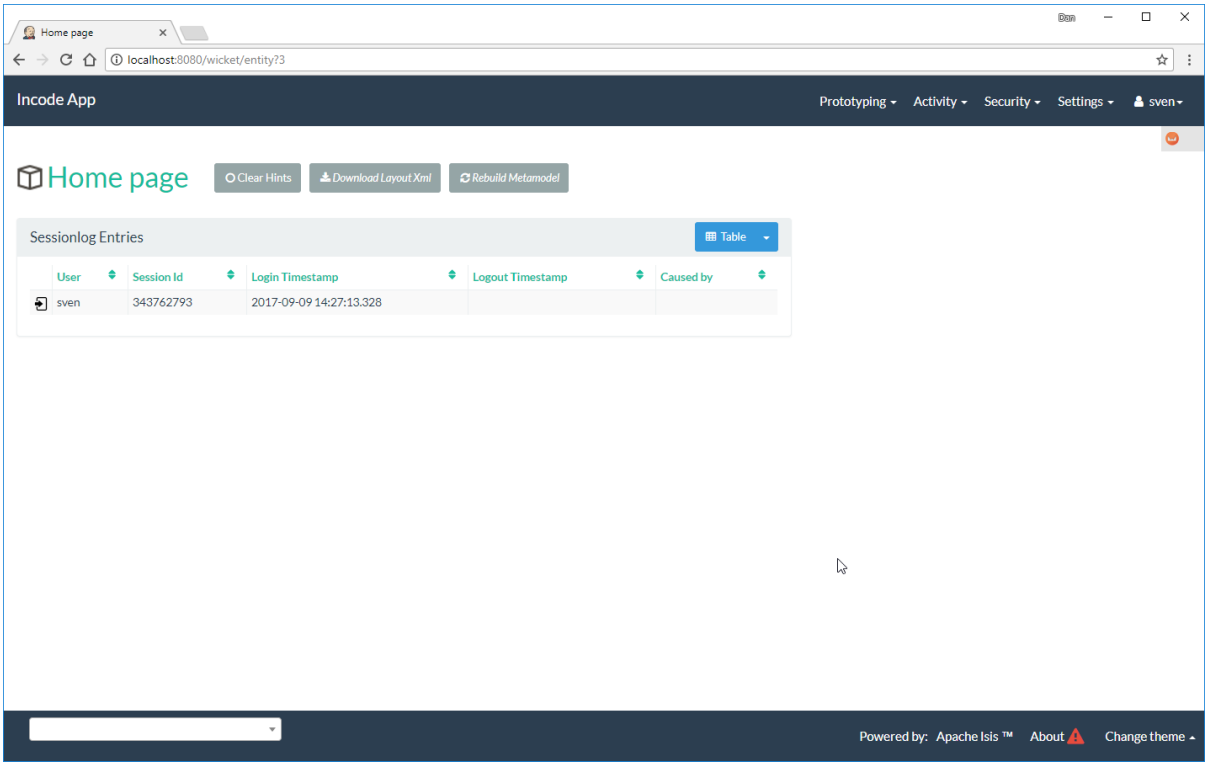
Table of Contents

This module (`isis-module-sessionlogger`) provides an implementation of Apache Isis' `SessionLoggingService` API that persists session entries (representing users logging in or out of the application) using Isis' own (JDO) objectstore. Typically this will be to a relational database; the module's `SessionLogEntry` entity is mapped to the "IsisSessionLogEntry" table.

Screenshots

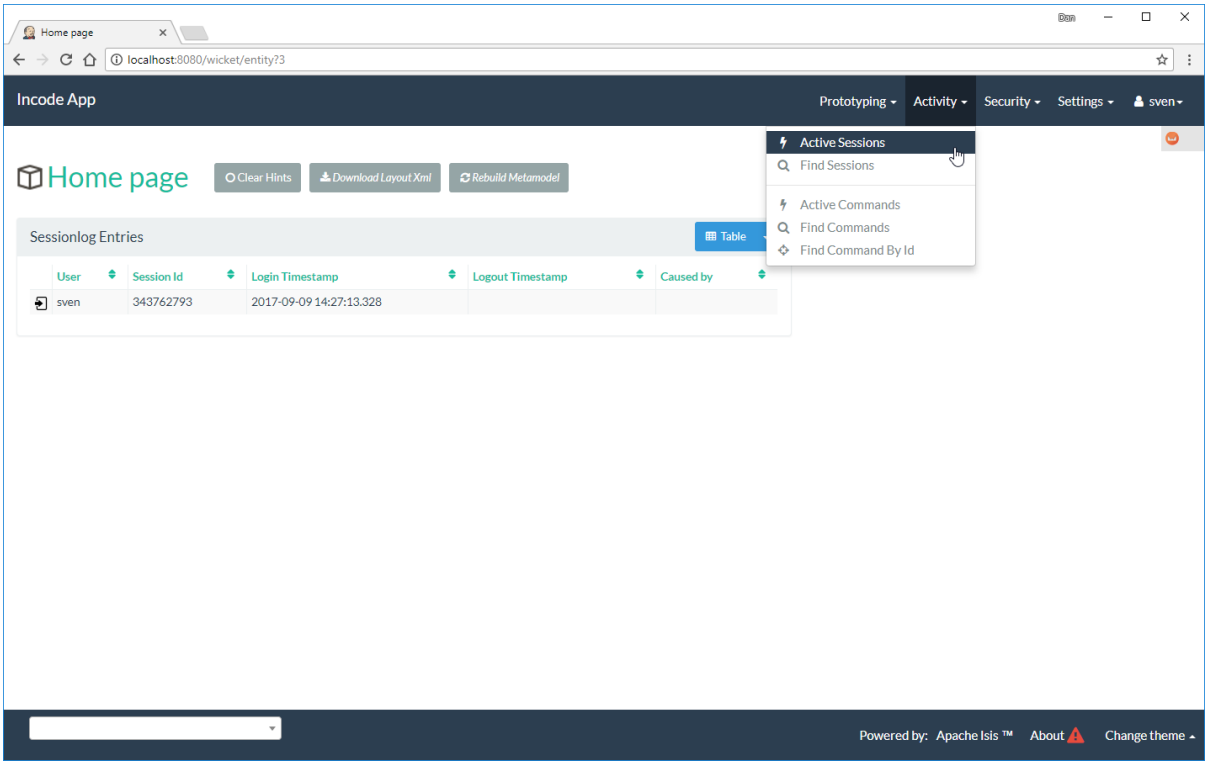
The module's functionality can be explored by running the quickstart with example usage using the `org.incode.domainapp.example.app.modules.ExampleDomSpiSessionLoggerAppManifest`.

A home page is displayed when the app is run:

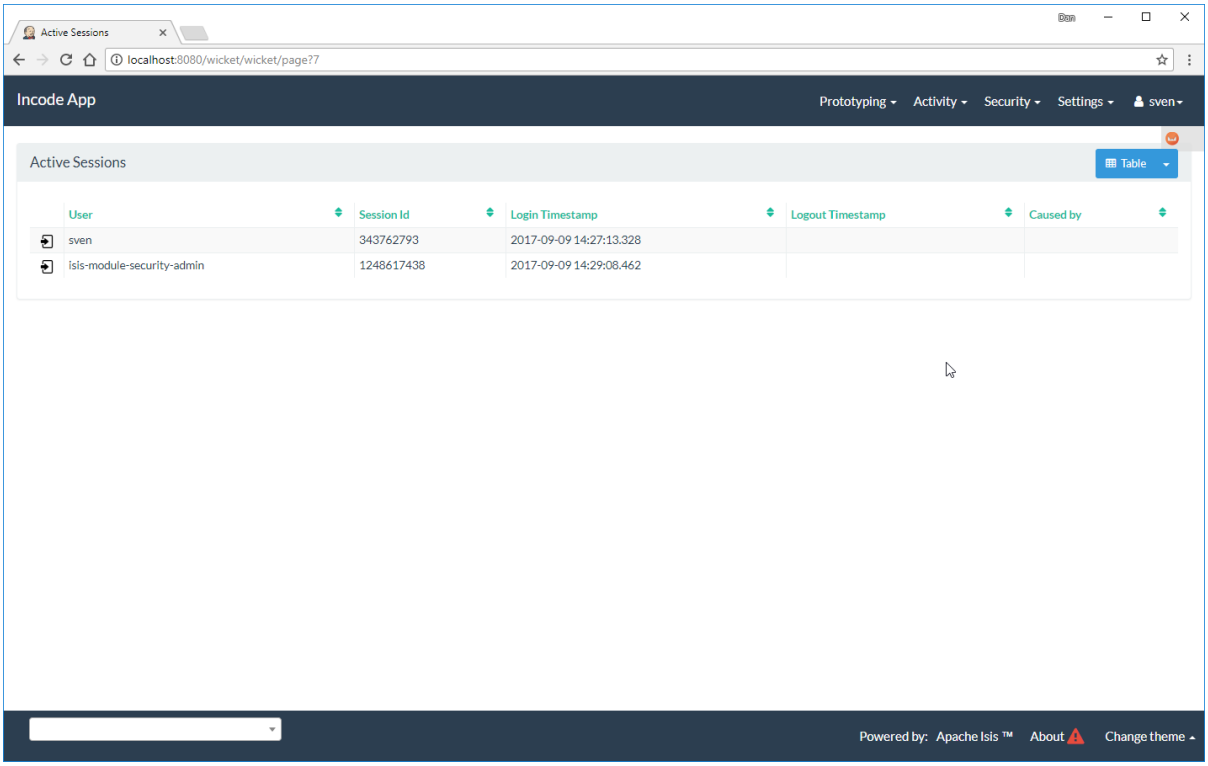


The sessionlogger module automatically creates log entries whenever a user logs on or logs out. The home page shows all session log entries, so shows one initially for the current user.

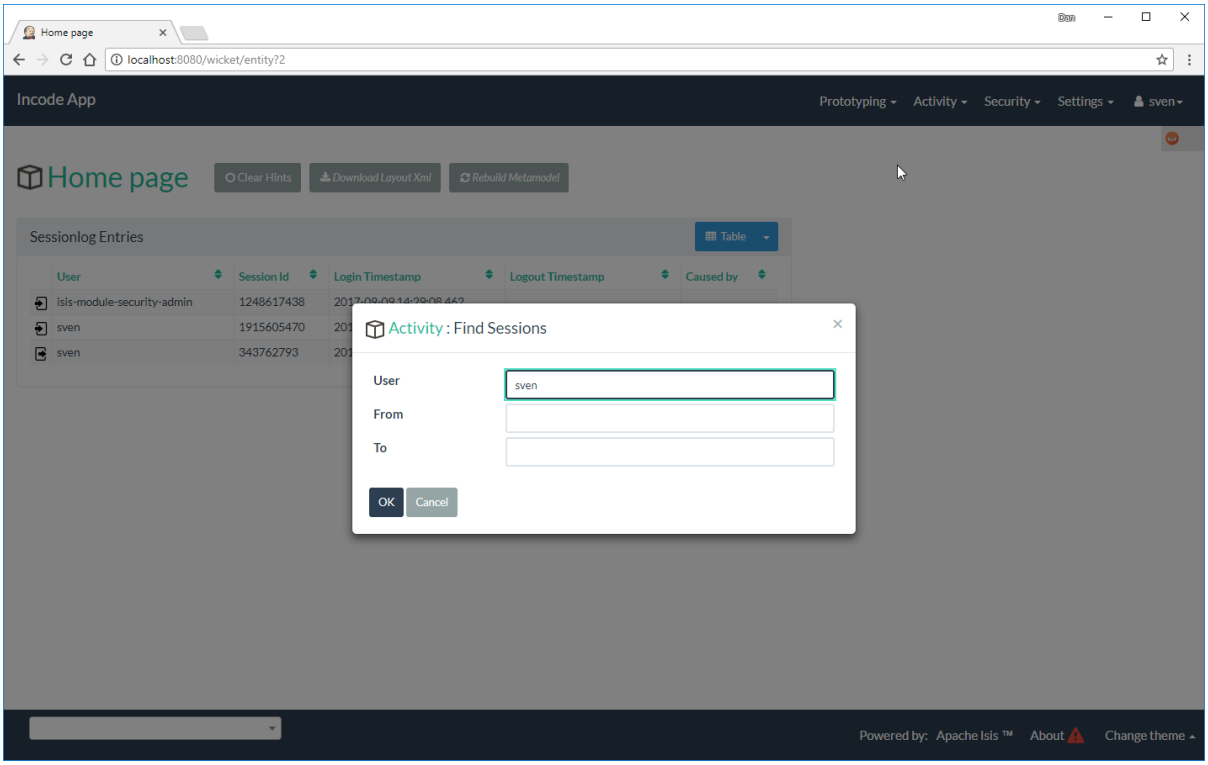
The currently logged on users of the application (that is: those for whom there is a valid non-expired HTTP session) can be found from the activity menu:



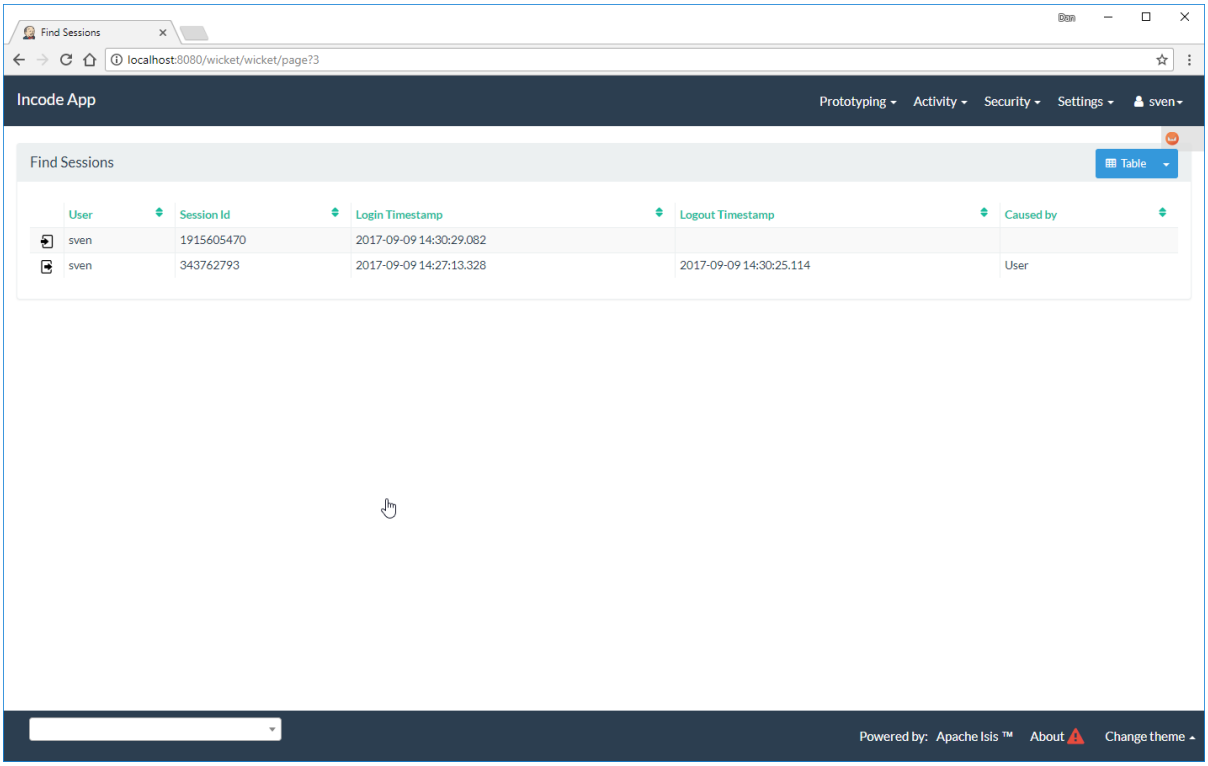
If on another browser another user logs in, then this will show two currently active users:



The list of sessions can optionally be filtered by user and date range:



returning matching sessions:



How to Configure/Use

Classpath

Update your classpath by adding this dependency in your project's `dom` module's `pom.xml`:

```
<dependency>
    <groupId>org.isisaddons.module.sessionlogger</groupId>
    <artifactId>isis-module-sessionlogger-dom</artifactId>
</dependency>
```

Check for releases by searching Maven Central Repo.

Bootstrapping

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

```
@Override
public List<Class<?>> getModules() {
    return Arrays.asList(
        ...
        org.isisaddons.module.sessionlogger.SessionLoggerModule.class,
        ...
    );
}
```

API

The `SessionLoggingService` defines the following API:

```
public interface SessionLoggingService {
    public enum Type {
        LOGIN,
        LOGOUT
    }
    public enum CausedBy {
        USER,
        SESSION_EXPIRATION,
        RESTART
    }
    void log(Type type, String username, Date date, CausedBy causedBy);
}
```

The framework will automatically call the `log(…)` method on the service implementation if configured to run the Wicket viewer.

The framework only ever calls `log(…)` with a `CausedBy` value of either "USER" (the user has explicitly logged in or logged out), or with "SESSION_EXPIRATION" (the Wicket viewer session has timed out).

The "RESTART" value is provided for implementations (such as the sessionlogger spi module) which automatically "tidy-up" and mark as complete and sessions that were in-progress if the webserver is restarted.

Implementation

The `SessionLoggingService` API is implemented in this module by the `org.isisaddons.module.sessionlogger.SessionLoggingServiceDefault` class. This implementation simply inserts a session log entry (`SessionLogEntry`) when either a user logs on, logs out or if their session expires.

The `SessionLogEntry` properties directly correspond to parameters of the `SessionLoggingService` `log()` API:

```
public class SessionLogEntry
    ...
    private String sessionId;                           ①
    private String username;                            ②
    private SessionLoggingService.Type type;            ③
    private Timestamp loginTimestamp;                   ④
    private Timestamp logoutTimestamp;                  ⑤
    private SessionLoggingService.CausedBy causedBy;    ⑥
    ...
}
```

① `sessionId` identifies the user's session. Primary key. (**Note**: it is not the http session id!)

② `username` identifies the user that has logged in/out

③ `type` determines whether this was a login or logout.

④ `loginTimestamp` is the date that the login of the session event occurred

⑤ `logoutTimestamp` is the date that the logout of the session event occurred

⑥ `causedBy`indicates whether the session was logged out due to explicit user action, by session expiry, or by the server restarting

The `SessionLogEntry` entity is designed such that it can be rendered on an Isis user interface if required.

Supporting Services

As well as the `SessionLoggingServiceDefault` service (that implements the `SessionLoggingService` API), the module also provides two further domain services:

- `SessionLogEntryRepository` provides the ability to search for persisted (`SessionLogEntry`) entries.

  None of its actions are visible in the user interface (they are all `@Programmatic`) and so this service is automatically registered.

- `SessionLoggingServiceMenu` provides the secondary "Activity" menu for listing all active sessions and for searching for session entries by user and by date.

  The `SessionLoggingServiceMenu` is automatically registered as a domain service; as such its actions will appear in the user interface. If this is not required, then either use security permissions or write a vetoing subscriber on the event bus to hide this functionality, eg:

```
@DomainService(nature = NatureOfService.DOMAIN)
public class HideIsisAddonsSessionLoggerFunctionality extends AbstractSubscriber {
    @Subscribe
    public void on(final SessionLoggerModule.ActionDomainEvent<?> event) { event
.hide(); }
}
```

Known issues or Limitations

The Restful Objects viewer currently does not support this service.

Related Modules/Services

There is some overlap with the`AuditingService3` API, which audits changes to entities by end-users. Implementations of this service are referenced by the Isis Add-ons website.

Dependencies

Maven can report modules dependencies using:

```
mvn dependency:list -o -pl modules/spi/sessionlogger/impl -D excludeTransitive=true
```

which, excluding Apache Isis itself, returns no direct compile/runtime dependencies.