Formevents

Custom code, attatched to an entity, getting called at different times

Creating the codeunit

1. Make sure that **p2eShared.jar** is included in the project
2. Create a new class that extends

```
com.tsnocode.codeunit.CodeunitFormevents
```

LIST level event hooks

```
public String appendListPageHead() { return ""; }
public String appendListPageFoot() { return ""; }
```

```
public void beforeSelectList() throws Exception {}
public void beforeRenderList() throws Exception {}
```

LIST execution order

1. **beforeSelectList**
2. Gather form data from database
    1. dataFilterActive > dataFilterHandler
    2. listFilterActive > listFilterHandler
3. **beforeRenderList**
4. Return list to user

ITEM level event hooks

```
public String appendItemPageHead() { return ""; }
public String appendItemPageFoot() { return ""; }
```

```
public void beforeSelectItem() throws Exception {}
public void beforeChangeItem() throws Exception {}
public void beforeUpdateItem() throws Exception {}
public void beforeRenderItem() throws Exception {}
public void afterUpdateItem() throws Exception {}

public boolean afterUpdateRedirectActive() { return false; }
public String afterUpdateRedirectContent()   { return null; }
```

ITEM execution order: viewing data

1. **beforeSelectItem**
2. Gather form data from database
   - dataFilterActive > dataFilterHandler
   - itemFilterActive > itemFilterHandler
3. **beforeRenderItem**
4. Return form to user

ITEM execution order: posting data

1. **beforeSelectItem**
2. Gather form data from database
3. **beforeChangeItem**
4. Update field values
5. **beforeUpdateItem**
6. Write changes to database
7. **afterUpdateItem**
8. if NO OTHER ACTION:
   - **afterUpdateRedirectActive**
   - if TRUE
     - **afterUpdateRedirectContent**
9. Return content to user

FILTER event hooks

Filters will help you build customized permission schemes. They are called for **both** LIST and ITEM commands.

```
@Override
protected boolean dataFilterActive() {
    return true;
}
@Override
protected void dataFilterHandler(StringBuilder sql) {
    sql.append(" AND something");
}
```

In some cases you only want the filter to trigger for **either** LIST or ITEM commands

```
itemFilterActive() {}
itemFilterHandler(StringBuilder sql) { return sql; }
listFilterActive() {}
listFilterHandler(StringBuilder sql) { return sql; }
```

Examples

The xxxFilterActive tells if the filter is active

```
boolean dataFilterActive() { return !s.isAdministor(); }
```

The xxxFilterHandler modifies the SQL used to fetch data

```
void dataFilterHandler(StringBuilder sql) { sql.append(" AND CATEGORY NOT IN (123,456,789)"); }
```

Event firing details

Event firing global

The following events are ALLWAYS fired

- beforeSelectList
- beforeSelectItem
- beforeChangeItem
- beforeUpdateItem
- afterUpdateItem

Event firing in UI (reserved for normal users)

The following events will NOT be fired during imports etc.

- appendListPageHead
- appendListPageFoot
- beforeRenderList
- appendItemPageHead
- appendItemPageFoot
- beforeRenderItem

Event firing in UI depending on user actions

The following events are SOMETIMES be fired for normal users depending on navigation

- afterUpdateRedirectActive
  - Not executed in SUBFORM mode
  - Not executed in during imports etc.
- afterUpdateRedirectContent
  - Depends on a TRUE result from afterUpdateRedirectActive()