The Guest pattern

Joop Ringelberg 27-08-21 Version: 1

Introduction

An end user can only access a context that (s)he has a perspective on. Yet, sometimes users will want to acquaint themselves with the requirements that taking a role in a context brings with it, before they do so[1]. We can handle that with object state: the perspectives of others may depend on the state of the role in question, so we might introduce a 'consent' Boolean property, for example.

Here we give another pattern, that depends on a particularity of deciding what role an end user plays in a context.

The role the user plays

When the end user opens a context instance, the InPlace client asks the Perspectives Distributed Runtime (PDR) what the type of the role is that the current user plays in it. The PDR applies this algorithm:

1. Return the type of the Enumerated role instance that is (ultimately) filled by the current user. If there is none,
2. Compute all Calculated User role instances and return the types of those that are (ultimately) filled by the current user.

Notice that more than one type may be returned. In that case, the user is presented with a choice in the user interface.

The pattern

A context that wants to offer an end user the opportunity to explore a context without committing to it, may introduce a calculated role like so:

```
User Guest = sys:Me
  Perspective on CommittedRole
    only (Create, Fill)
```

Where `CommittedRole` is the role that brings commitment and requirements with it.

An end user first visits the context and assumes the Guest role (the outcome of the algorithm given above). He then creates an instance of `CommittedRole` and fills it with

[1] E.g., the role may require filling with details that the user does not part with without serious consideration.

himself. From that moment, the algorithm no longer returns the type of `Guest`, but that of `CommittedRole`. His perspectives change accordingly.