The context with a presence pattern

Joop Ringelberg 23-05-22 Version: 1

Introduction

When we say a user 'enters a context', we can mean two things: either he takes up a role in that context, or he opens it on screen (the latter on condition of the former). The first meaning is a matter of state in Perspectives: the role instance is part of the context, filled by the user, or it is not. The second meaning, however, is a state that is much more ephemereal. We could describe it as GUI-state. But for some types of context, it would be convenient to know whether a user actually can *see* it at a particular moment, or not. Think of a chat room, or a board game: participants would like to know whether their chat partners or game opponents are currently available, or not.

ContextWithScreenState

`model:System` contains the context type `ContextWithScreenState`:

```
case ContextWithScreenState
  external
    property IsOnScreen (Boolean)
```

The value of the property IsOnScreen reflects whether, at any moment, the context is actually visible in the InPlace application. This depends on specific code in the Screen React component.

InPlace will actually try to set this property on *every context that is opened*. However, the PDR refuses to set a non-declared property on a role and fails silently to do so.

In order to actually use this facility, provide your context type with `sys:ContextWithScreenState` as an Aspect (and its external role with the Aspect external role).

Limitations

Note that as soon as more than one user play a role in such a context, the semantics of `IsOnScreen` changes to *at least one user can see it on screen.*

In order to monitor *who* actually sees the context, we'd have to modify the mechanism to change a property on a user role. InPlace must then be adapted to try to modify that property for the current user opening the screen (Note: this will not work for calculated user roles!).