Issue
Inspector-Hook

Forum

Common Lisp Document Repository (CDR)

Status

Final.

References

`inspect` (function), `*debugger-hook*` (variable)

Category

Addition.

Edit History

20-Mar-2008 by Rittweiler (Draft)
30-Sept-2008 by Rittweiler (Typesetting)
03-Oct-2008 by Rittweiler (CDR: Initial)
16-Nov-2008 by Rittweiler (CDR: Final)

Problem Description

Cross-implementation Lisp programming environments such as Slime or Climacs face the problem that the ANSI Common Lisp standard does not specify any way to hook into the inspection machinery of an implementation. Specifically, there is no portable way to install an inspector that is run when the function `inspect` is invoked.

Proposal (Inspector-Hook:`*inspector-hook*`-Variable)

Add a variable `*inspector-hook*` that is called by the `inspect` function to delegate inspection. See the appendix of this document for a detailed specification.

Rationale

This proposal is closely modelled on the specification of the `*debugger-hook*` variable, and allows programs to install their own inspectors that are adapted to their respective domains.

Notes

The proposal does deliberately say nothing about the home-package of the symbol `*inspector-hook*`. However, implementors are encouraged to export this symbol from their extensions package (often called "`EXT`") or another appropriate package—unless a later CDR document specifies a more explicit location.

Cost to implementators

Negligible.

Current Practice

On at least SBCL, Clisp, and Clozure CL, it's possible to straightforwardly hook into the inspection machinery, albeit by mostly unexported and undocumented means.

The documentation of Allegro CL states that an invocation of `inspect` runs a GUI inspector on the MS Windows operating system, and a text-based inspector on other operating systems, so it's very likely that they employ some hook mechanism internally.

Appendix

`*INSPECTOR-HOOK*` **[Special Variable]**

Value Type

A designator for a function of one argument (the object to be inspected), or `nil`.

Initial Value

`nil`.

Description

When the value of `*inspector-hook*` is non-`nil`, an invocation of `inspect` in the dynamic extent of this value results in calling the value with the object that was originally passed to `inspect`. The function `inspect` delegates its work to the value of `*inspector-hook*` this way.

`*inspector-hook*` is not rebound before calling the function denoted by the value, i.e. the function is executed in the same dynamic environment as at the invocation of `inspect`; this is to allow inspectors to be defined recursively.

The return value of `inspect` is independent of the return value of the denoted function, and remains implementation-dependent.

Affected By

`inspect`

See Also

`*debugger-hook*`

Notes

It's possible to invoke the implementation's inspection machinery from within `*inspector-hook*` by binding it to `nil` before invoking `inspect`.