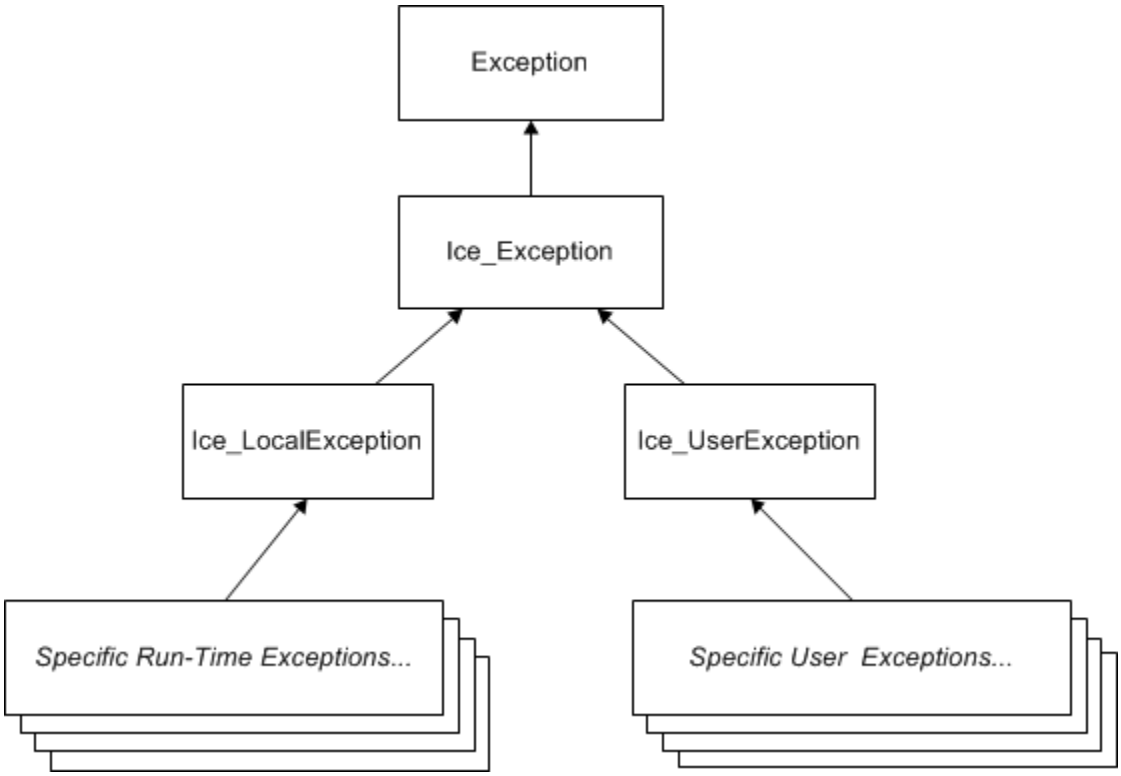
PHP Mapping for Exceptions

On this page:

Inheritance Hierarchy for Exceptions in PHP

The mapping for exceptions is based on the inheritance hierarchy shown below:



Inheritance structure for Ice exceptions.

The ancestor of all exceptions is `Exception`, from which `Ice_Exception` is derived. `Ice_LocalException` and `Ice_UserException` are derived from `Ice_Exception` and form the base for all run-time and user exceptions.

PHP Mapping for User Exceptions

Here is a fragment of the Slice definition for our world time server once more:

Slice

```
exception GenericError {
    string reason;
};
exception BadTimeVal extends GenericError {};
exception BadZoneName extends GenericError {};
```

These exception definitions map to the abbreviated PHP class definitions shown below:

PHP

```
class GenericError extends Ice_UserException
{
    public function __construct($reason='');
    public function ice_name();
    public function __toString();

    public $reason;
}

class BadTimeVal extends GenericError
{
    public function __construct($reason='');
    public function ice_name();
    public function __toString();
}

class BadZoneName extends GenericError
{
    public function __construct($reason='');
    public function ice_name();
    public function __toString();
}
```

Each Slice exception is mapped to a PHP class with the same name. The inheritance structure of the Slice exceptions is preserved for the generated classes, so `BadTimeVal` and `BadZoneName` inherit from `GenericError`.

Each exception member corresponds to an instance variable of the instance, which the constructor initializes to a default value appropriate for its type. You can also declare different default values for members of primitive and enumerated types. For derived exceptions, the constructor has one parameter for each of the base exception's data members, plus one parameter for each of the derived exception's data members, in base-to-derived order. As an example, although `BadTimeVal` and `BadZoneName` do not declare data members, their constructors still accept a value for the inherited data member `reason` in order to pass it to the constructor of the base exception `GenericError`.

Optional data members use the same mapping as required data members, but an optional data member can also be set to the marker value `Ice_Unset` to indicate that the member is unset. A well-behaved program must compare an optional data member to `Ice_Unset` before using the member's value:

PHP

```
try {
    ...
} catch($ex) {
    if($ex->optionalMember == Ice_Unset)
        echo "optionalMember is unset\n";
    else
        echo "optionalMember = " . $ex->optionalMember . "\n";
}
```

Each exception also defines the `ice_name` method to return the exception's type name, as well as the `__toString` magic method to return a stringified representation of the exception and its members.

All user exceptions are derived from the base class `Ice_UserException`. This allows you to catch all user exceptions generically by installing a handler for `Ice_UserException`. Similarly, you can catch all Ice run-time exceptions with a handler for `Ice_LocalException`, and you can catch all Ice exceptions with a handler for `Ice_Exception`.

PHP Mapping for Run-Time Exceptions

The Ice run time throws run-time exceptions for a number of pre-defined error conditions. All run-time exceptions directly or indirectly derive from `Ice_LocalException` (which, in turn, derives from `Ice_Exception`).

By catching exceptions at the appropriate point in the inheritance hierarchy, you can handle exceptions according to the category of error they indicate:

- `Ice_LocalException`
  This is the root of the inheritance tree for run-time exceptions.
- `Ice_UserException`
  This is the root of the inheritance tree for user exceptions.
- `Ice_TimeoutException`
  This is the base exception for both operation-invocation and connection-establishment timeouts.
- `Ice_ConnectTimeoutException`
  This exception is raised when the initial attempt to establish a connection to a server times out.

For example, `Ice_ConnectTimeoutException` can be handled as `Ice_ConnectTimeoutException`, `Ice_TimeoutException`, `Ice_LocalException`, or `Ice_Exception`.

You will probably have little need to catch run-time exceptions as their most-derived type and instead catch them as `Ice_LocalException`; the fine-grained error handling offered by the remainder of the hierarchy is of interest mainly in the implementation of the Ice run time. Exceptions to this rule are the exceptions related to facet and object life cycles, which you may want to catch explicitly. These exceptions are `Ice_FacetNotExistException` and `Ice_ObjectNotExistException`, respectively.

See Also

- User Exceptions
- Run-Time Exceptions
- PHP Mapping for Identifiers
- PHP Mapping for Modules
- PHP Mapping for Built-In Types
- PHP Mapping for Enumerations
- PHP Mapping for Structures
- PHP Mapping for Sequences
- PHP Mapping for Dictionaries
- PHP Mapping for Constants
- Optional Data Members
- Facets and Versioning
- Object Life Cycle