FQLUserPic and FQLUserBooks differ only in their internal property $this->name, set by their constructor during processing. Note that underneath, we call user_get_info for every evaluation we need in the expression; this performs well only if the system caches these results in process memory. Facebook's implementation does just that, and the whole query executes in time on the order of a standard platform API call.

Here is a more complex field representing current_location, which takes the same input and exhibits the same usage pattern, but outputs a struct-type object we've seen earlier (Example 6-19).

EXAMPLE 6-19. A more complex FQL field mapping

```
// complex object data field
class FQLUserCurrentLocation extends FQLStringUserField {
  public function evaluate($id) {
    $info = user_get_info($id);
    if ($info && isset($info['current_location'])) {
      $location = new api1O_location($info['current_location']);
    } else {
      $location = new api1O_location();
    }
    return $location;
  }
}
```

Objects such as api1O_location are the generated types from "Data: Creating an XML Web Service," which Thrift and the Facebook data service know how to return as well-typed XML. Now we're seeing why even with a new input style, FQL's output does not need to be incompatible with that of the Facebook API.

The main evaluation loop of FQLStatement in the following example provides a high-level idea of FQL's implementation. Throughout this code we reference FQLExpressions, but in a simple query, we're mostly talking about FQLFieldExpressions, which wrap internal calls to the FQLField's own evaluate and can_see methods, as in Example 6-20.

EXAMPLE 6-20. A simple FQL expression class

```
class FQLFieldExpression {

  // instantiated with an FQLField in the "field" property
  public function evaluate($id) {
    if ($this->field->can_see($id))
      return $this->field->evaluate($id);
    else
      return new FQLCantSee(); // becomes an error message or omitted field
  }

  public function get_name() {
    return $this->field_name;
  }
}
```