```
public $pic = null;
public $current_location = null;


public function __construct($vals=null) {
  if (is_array($vals)) {
    if (isset($vals['uid'])) {
       $this->uid = $vals['uid'];
     }
     if (isset($vals['name'])) {
       $this->name = $vals['name'];
     }
     if (isset($vals['books'])) {
       $this->books = $vals['books'];
     }
     if (isset($vals['pic'])) {
       $this->pic = $vals['pic'];
     }
     if (isset($vals['current_location'])) {
       $this->current_location = $vals['current_location'];
    }
   // ...
  }
 // ...
}
```

All internal methods returning the type user (such as the internal implementation of users_getInfo) create all needed fields and end with something like the line in Example 6-9.

EXAMPLE 6-9. Consistent use of generated type

```
return new api_10_user($field_vals);
```

For example, if the current_location is present in this user object, then $field_vals['current_location'] is set to new api_10_location(...) somewhere before Example 6-9 is executed.

The names of the fields and types themselves actually generate the schema for the XML output, as well as the accompanying XML Schema Document (XSD). Example 6-10 shows an example of the actual XML output of the whole RPC flow.

EXAMPLE 6-10. XML output from web service call

```
<users_getInfo_response list="true">
 <users type="list">
  <user>
   <name>Dave Fetterman</name>
   <books>Zen and the Art, The Brothers K, Roald Dahl</books>
   <pic></pic>
   <current_location>
    <city>San Francisco</city>
    <state>CA</state>
    <zip>94110</zip>
```