inter-process communication (IPC) system called Thrift (*http://developers.facebook.com/thrift*) that accomplishes this cleanly.

Diving right in, Example 6-7 shows an example "dot thrift" file for our sample API version 1.0, which the Thrift package turns into much of the machinery of the API.

EXAMPLE 6-7. Web service definition through Thrift

```
xsd_namespace http://api.facebook.com/1.0/
/***
* Definition of types available in api.facebook.com version 1.0
*/
typedef i32 uid
typedef string uid_list
typedef string field_list

struct location {
 1: string street xsd_optional,
 2: string city,
 3: string state,
 4: string country,
 5: string zip xsd_optional
}

struct user {
 1: uid uid,
 2: string name,
 3: string books,
 4: string pics,
 5: location current_location
}

service FacebookApi10 {

 list<uid> friends_get()
  throws (1:FacebookApiException error_response),

 list<user> users_getInfo(1:uid_list uids, 2:field_list fields)
  throws (1:FacebookApiException error_response),
}
```

Each type in this example is a primitive (`string`), a structure (`location`, `user`), or a generic-style collection (`list<uid>`). Because each method declaration has a well-typed signature, code defining the reused types can be directly generated in any language. Example 6-8 shows part of the generated output for PHP.

EXAMPLE 6-8. Thrift-generated service code

```
class api10_user {

public $uid = null;
public $name = null;
public $books = null;
```