极客学院
jikexueyuan.com

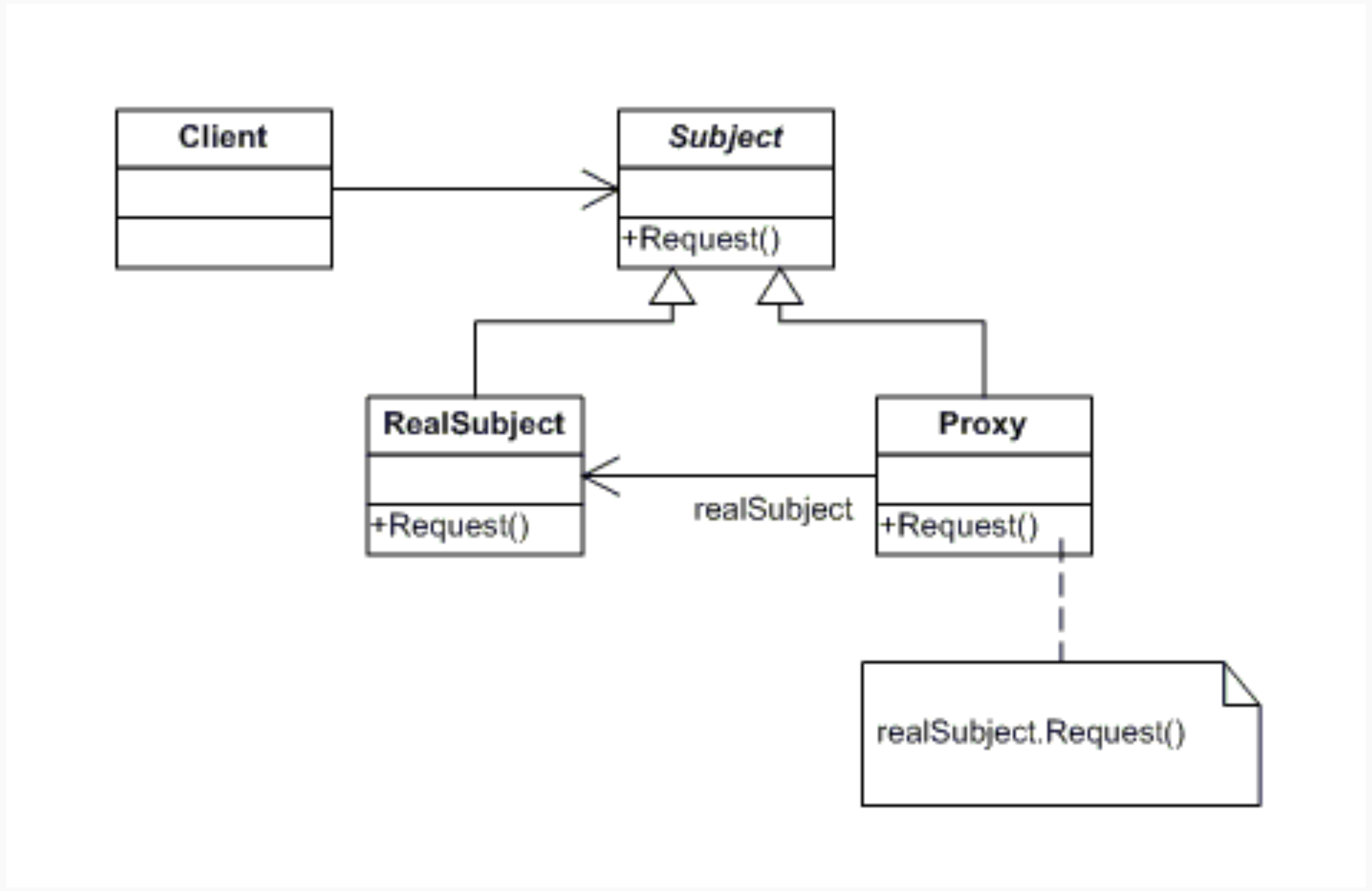
iOS设计模式 – 代理

iOS设计模式 – 代理 — **课程概要**

- 代理模式
- 代理与协议
- 用 NSProxy 实现代理模式

代理模式

代理模式


Client
Subject
+Request()
RealSubject
+Request()
Proxy
+Request()
realSubject
realSubject.Request()

代理模式

- 代理模式基本原理
- 使用代理模式需要注意的地方

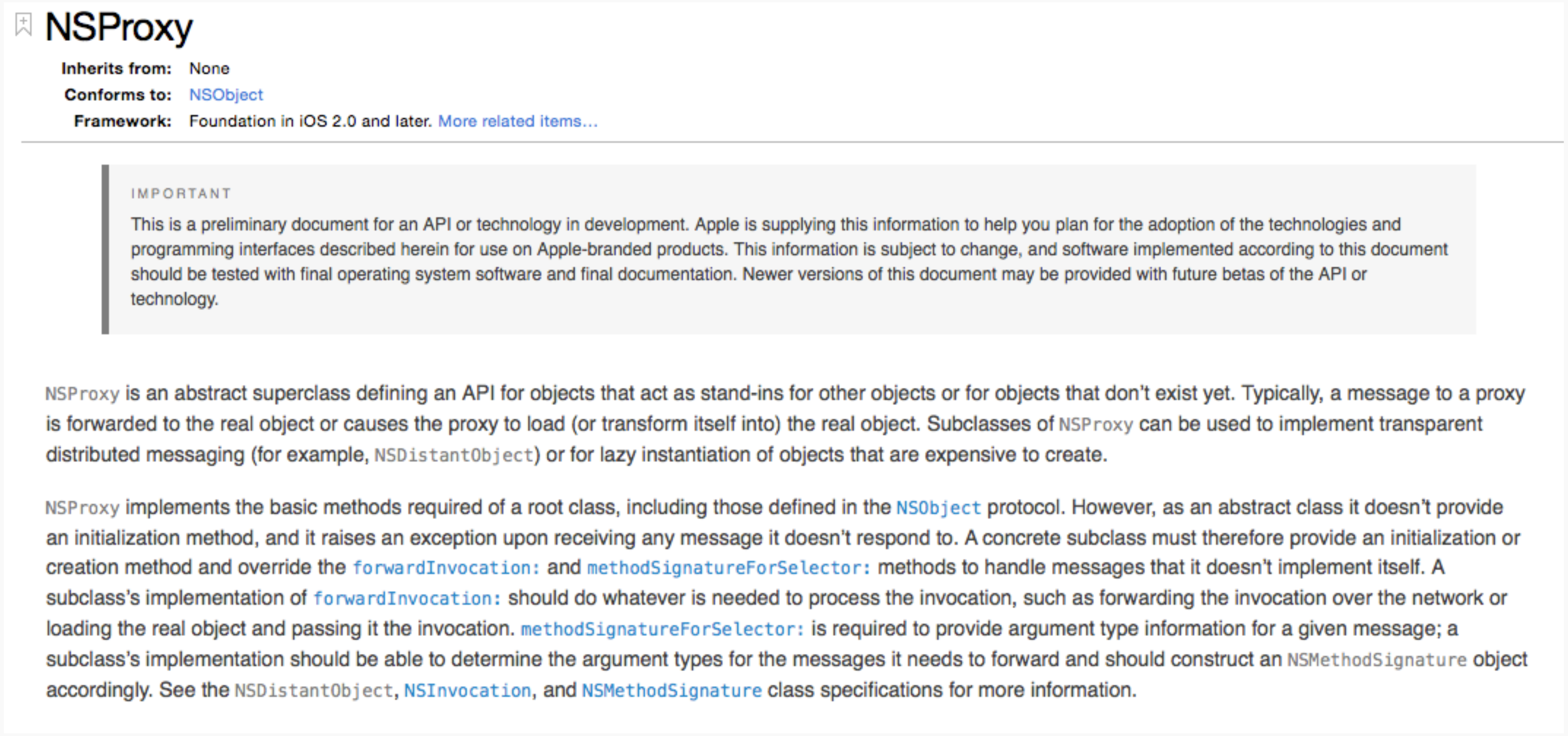
代理与协议

代理与协议

- 代理的职能
- 协议的功能
- 代理与协议的相似性
- 代理与协议的不同点

用 NSProxy 实现代理模式

用 NSProxy 实现代理模式



NSProxy

Inherits from:	None
Conforms to:	NSObject
Framework:	Foundation in iOS 2.0 and later. More related items...

IMPORTANT

This is a preliminary document for an API or technology in development. Apple is supplying this information to help you plan for the adoption of the technologies and programming interfaces described herein for use on Apple-branded products. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. Newer versions of this document may be provided with future betas of the API or technology.

`NSProxy` is an abstract superclass defining an API for objects that act as stand-ins for other objects or for objects that don't exist yet. Typically, a message to a proxy is forwarded to the real object or causes the proxy to load (or transform itself into) the real object. Subclasses of `NSProxy` can be used to implement transparent distributed messaging (for example, `NSDistantObject`) or for lazy instantiation of objects that are expensive to create.

`NSProxy` implements the basic methods required of a root class, including those defined in the `NSObject` protocol. However, as an abstract class it doesn't provide an initialization method, and it raises an exception upon receiving any message it doesn't respond to. A concrete subclass must therefore provide an initialization or creation method and override the `forwardInvocation:` and `methodSignatureForSelector:` methods to handle messages that it doesn't implement itself. A subclass's implementation of `forwardInvocation:` should do whatever is needed to process the invocation, such as forwarding the invocation over the network or loading the real object and passing it the invocation. `methodSignatureForSelector:` is required to provide argument type information for a given message; a subclass's implementation should be able to determine the argument types for the messages it needs to forward and should construct an `NSMethodSignature` object accordingly. See the `NSDistantObject`, `NSInvocation`, and `NSMethodSignature` class specifications for more information.

用 NSProxy 实现代理模式

- NSProxy 中的消息传递机制
- NSProxy 的用途
- 用 NSProxy 实现代理模式

iOS设计模式 – 代理

本套课程中我们学习了 iOS 设计模式 – 代理模式，你应当掌握了以下知识：

- 代理模式
- 代理与协议
- 用 NSProxy 实现代理模式

你可以用所学的知识来重新 review 以前所写的代码，进行适当的改进，如果想继续提高，你可以继续在极客学院学习iOS设计模式 – 单例。