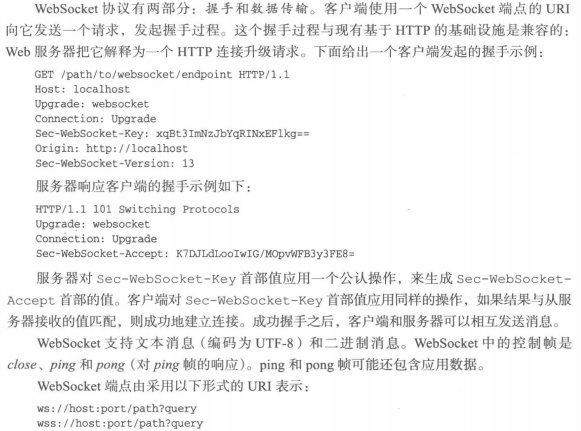
在websocket应用中，服务器发布一个Websocket端点，客户端使用这个端点的URI来连接服务器。

建立连接后，Websocket协议是对称的；客户端和服务器可以在连接打开的任何时间相互发送消息， 而且它们可以在任何时间关闭连接。

WebSocket协议有2部分；握手和数据传输。客户端使用一个WebSocket端点的URI向它发送一个请求，发起握手过程。



Ws模式表示一个未加密的WebSocket连接，wss模式表示一个加密的连接。port是可选的；对于未加密的连接，默认的端口是80，对于加密连接，默认的端口与是443.

WebSocket Java Api:

javax.websocket.server包含创建和配置服务器端点的注解，类和接口。

javax.websocket包含客户端和服务器端点公共的注解，类，接口和异常。

WebSocket端点是javax.websocket.Endpoint类的实例。

创建端点有2点方式

1.扩展Endpoint类并覆盖其生命周期方法。

2.注解端点。

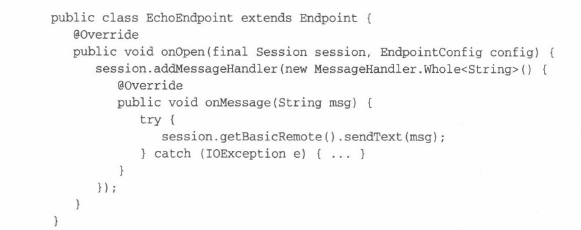
创建和部署WebSocket端点的过程如下：

1.创建一个端点类

2.实现这个端点的生命周期方法。

3.为端点增加业务逻辑

4.将这个端点部署在一个Web应用中。



EndPoint定义了三个生命周期的方法：onOpen,onClose和onError

Session参数表示这个端点和远程端点之间的一个会话。addMessageHandler方法注册消息处理器，getBasicRemote方法返回一个表示远程端点的对象。

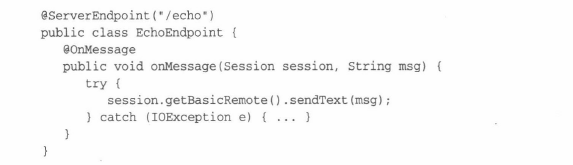
要部署这个可编程端点，需要在java ee应用中使用以下代码：

ServerEndpointConfig.Builder.create(EchoEndpoint.class, "/echo").build();

部署应用时，这个端点可以在ws://<host>:<port>/<application>/echo得到vk：

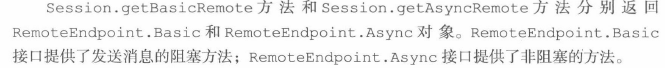
ws://localhost:8080/echoapp/echo

注解端点：、

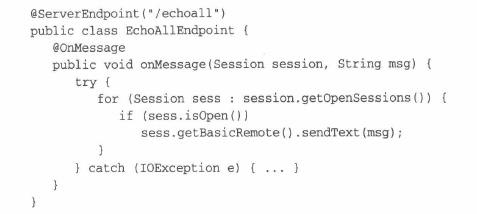




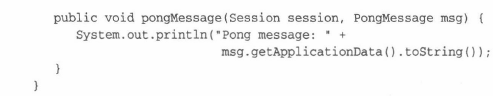
WebSocket端点可以发送和接收文本和二进制消息。它们还可以发送ping帧和接收pong帧。



发送消息









实现编码器将java对象转换为WebSocket消息

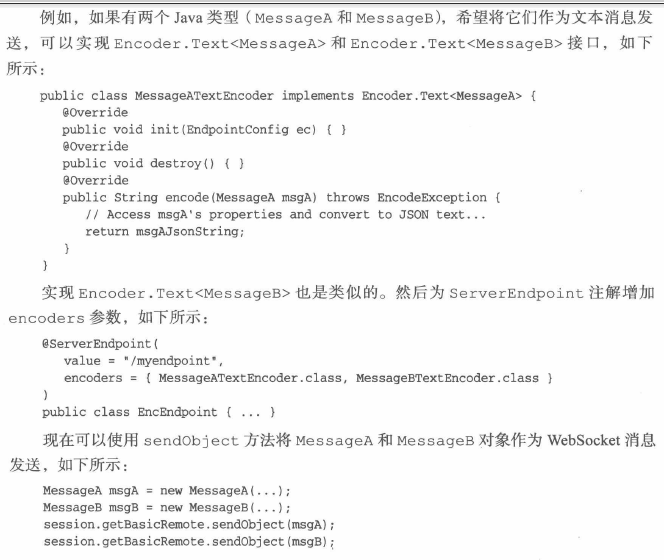
1.实现以下某个接口（这些接口实现了encode方法）：

对于文本消息实现Encoder.Text<T>

对于二进制消息实现Encoder.Binary<T>

2.将编码器实现的名字增加到ServerEndpoint注解的可选参数encoders中。

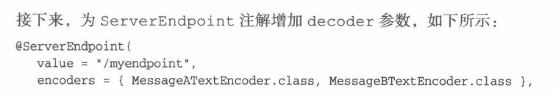
3.使用RemoteEndpoint.Basic或RemoteEndpoint.Async接口的sendObject(Object data)方法将对象作为消息发送。容器会自动找到相应的编码器，用它将对象转换为一个websocket消息。



如果有编码器，同样有解码器。实现接口Decoder.Text<T>.

将解码器实现的名字增加到ServerEndpoint注解的可选参数decoders中。





ServerEndpoint注解允许使用URI模板将一个端点部署URI的各个部分指定为应用参数。

@ServerEndpoing("/chatrooms/{room-name}")

public class ChatEndpoint {...}



当存在连接问题或消息处理运行时错误，或者解码消息时出现转换错误时，可以在@onError中捕捉。

指定端点配置器类：

利用websocket java api可以配置容器如何创建服务器端点实例。通过提供定制的端点配置逻辑可以完成以下工作：

访问一个WebSocket连接的初始http请求细节。

对origin http首部完成定制检查

修改websocket握手响应

从客户端请求协议中选择一个websocket子协议

控制端点实例化和初始化

要提供定制端点配置逻辑，需要扩展ServerEndpointConfig.Configuration类，并覆盖它的一些方法。



