Akka-demo

Подключаете в Maven

```
<repository>
        <id>akka.repository</id>
        <name>Akka Maven Repository</name>
        <url>akka.io/repository</url>
</repository>
<dependency>
        <groupId>se.scalablesolutions.akka</groupId>
        <artifactId>akka-actor</artifactId>
        <version>1.1.3</version>
</dependency>
<dependency>
        <groupId>se.scalablesolutions.akka</groupId>
        <artifactId>akka-typed-actor</artifactId>
        <version>1.1.3</version>
</dependency>
```

TestUntypedActor

```
import akka.actor.UntypedActor;

public class TestUntypedActor extends UntypedActor {
	public void onReceive(Object message) throws Exception {
		if (message instanceof MySuperMessage) {
			System.out.println("We've got super message - " + message);
		} else if (message instance of String) {
			System.out.println("Process string - " + message);
		}
	}
}
```

Для создания UntypedActor

```
ActorRef testActor = Actors.actorOf(TestUntypedActor.class);
```

TypedActor

```
interface RegistrationService {
        void register(User user, Credentials cred);
        User getUserFor(String username);
}
```

```
import akka.actor.TypedActor;
public class RegistrationServiceImpl extends TypedActor implements RegistrationService {
        public void register(User user, Credentials cred) {
                ... // Регистрация пользователя
        }
        public User getUserFor(String username) {
                ... // Реализация поиска пользователя
                return user;
        }
}
```

Для создания TypedActor

```
// Первый параметр - интерфейс, второй - реализация
UserService service =
(UserService) TypedActor.newInstance(UserService.class, UserServiceImpl.class);

// Или если у TypedActor нет конструктора без параметров
UserService service = TypedActor.newInstance(
        UserService.class,
        new TypedActorFactory() {
                public TypedActor create() {
                        return new UserServiceImpl("default user storage"));
                }
        }
);
```

Akka реализует 3 типа сообщений actor'ам

- fire-and-forget («отправил и забыл», асинхронная отправка сообщения без ожидания результата),
- request-reply («вопрос-ответ», отправка сообщения и ожидание ответа, синхронный режим),
- request-reply-with-future («отправить и получить ответ в будущем», отправка сообщения и получения ответа дальше по коду с помощью специального объекта).

fire-and-forget («отправил и забыл»)

fire-and-forget («отправил и забыл», асинхронная отправка сообщения без ожидания результата),

```
actorRef.sendOneWay("Hello"); // Посылает в качестве сообщения строчку

// Отправить объект в качестве сообщения.
// и context текущего actor'a (из которого происходит вызов)
// чтобы можно было отправить ответ с помощью context().reply()
actorRef.sendOneWay(new MySuperObject(), context());
```

request-reply («вопрос-ответ»)

request-reply («вопрос-ответ», отправка сообщения и ожидание ответа, синхронный режим),

```
// Отправить строчку как сообщение и ждать ответ.
// В методе можно также передать context()
// и таймаут ожидания ответа (при истечении таймаута - ActorTimeoutException)
Object result = actorRef.sendRequestReply("get result");
```

request-reply-with-future («отправить и получить ответ в будущем»)

request-reply-with-future («отправить и получить ответ в будущем», отправка сообщения и получения ответа дальше по коду с помощью специального объекта),

```
// Отправить сообщение и получить объект для извлечения результатов.
// Можно передать context() и таймаут.
// У самого объекта Future есть методы await (ожидание результата),
// result (получить результат),
// exception (получить исключение если было)

Future<MyResult> future = actorRef.sendRequestReplyFuture("Hello");
```

Demo

```
import akka.actor.*;
public class Hello1 {
    public static void main(String[] args) {
        ActorSystem system = ActorSystem.create("actor-demo-java");
        ActorRef hello = system.actorOf(Props.create(Hello.class));
        hello.tell("Bob", ActorRef.noSender());
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            /* ignore */
        }
        system.shutdown();
    }
}
```

```
class Hello extends UntypedActor {
    public void onReceive(Object message) throws Exception {
        if (message instanceof String) {
            System.out.println("Hello " + message); }
        }
    }
}
```

Getting Started Tutorial (Java): First Chapter

https://doc.akka.io/docs/akka/2.0.1/intro/getting-started-first-java.html