CS 525 - ASD

Advanced Software Development

MS.CS Program

Department of Computer Science

Rene de Jong, MsC.

Maharishi University
OF MANAGEMENT

CS 525 - ASD

Advanced Software Development

Maharishi University
OF MANAGEMENT

Event publisher and listener

```
@Service
public class CustomerServiceImpl implements CustomerService {
  @Autowired
  private ApplicationEventPublisher publisher;

  public void addCustomer() {
    publisher.publishEvent(new AddCustomerEvent("New customer is added"));
  }
}
```

Inject a publisher

```
@Service
public class Listener {

  @EventListener
  public void onEvent(AddCustomerEvent event) {
    System.out.println("received event :" + event.getMessage());;
  }
}
```

Listen to AddCustomer events

Events

```
public class AddCustomerEvent {
  private String message;

  public AddCustomerEvent(String message) {
    this.message = message;
  }

  public String getMessage() {
    return message;
  }
}
```

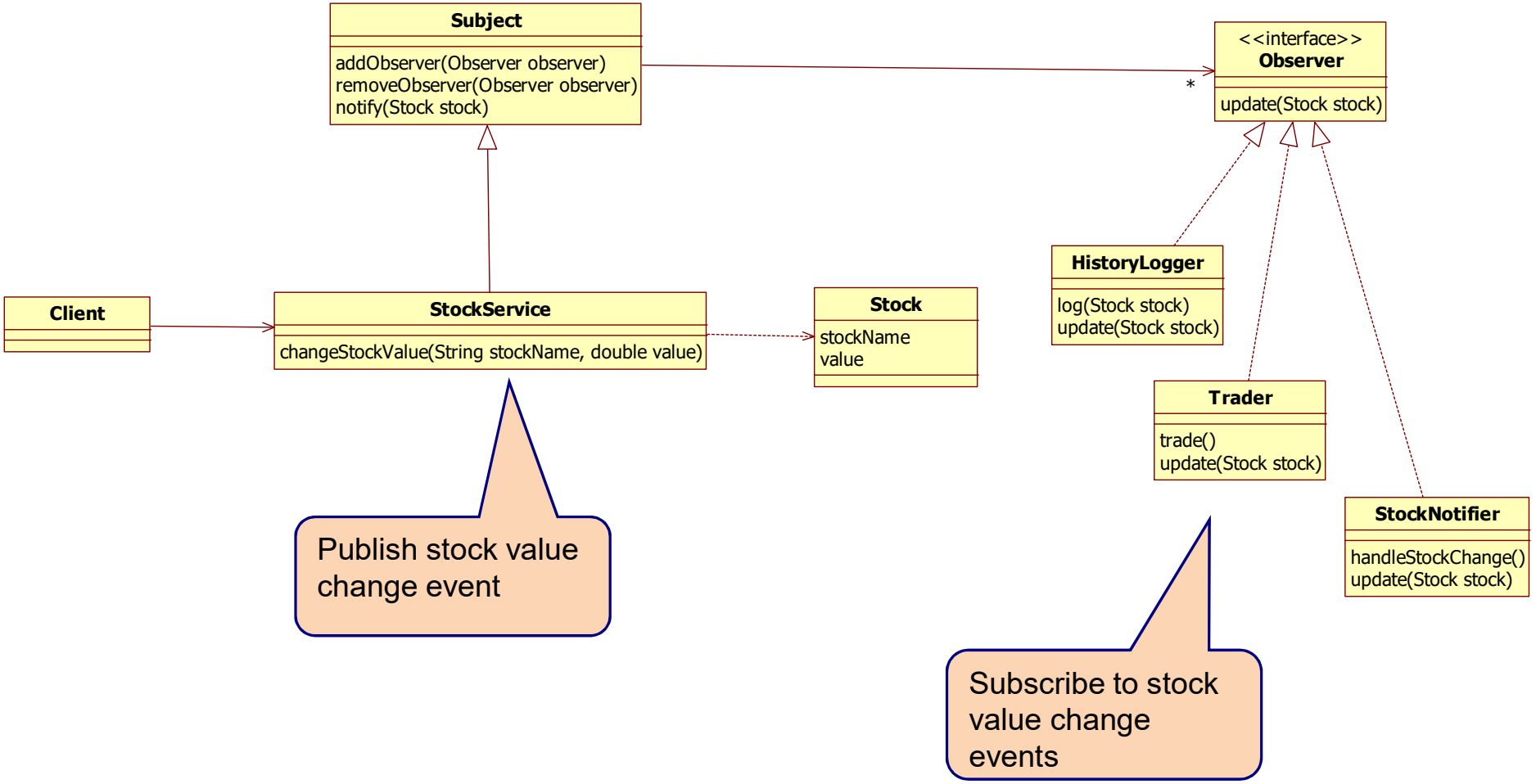
A simple event class

Asynchronous events

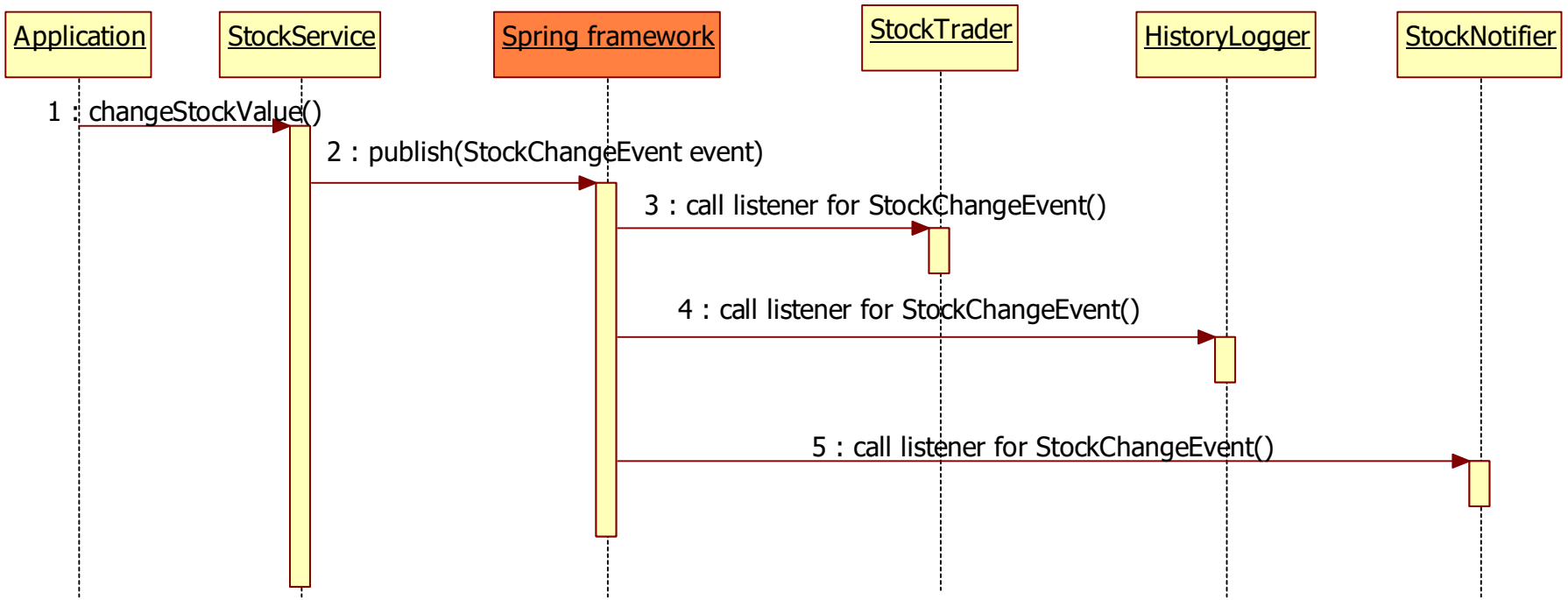
```
@Service
@EnableAsync
public class Listener {

  @Async
  @EventListener
  public void onEvent(AddCustomerEvent event) {
    System.out.println("received event :" + event.getMessage());;
  }
}
```

Observer pattern



Spring publish-subscribe



Publisher

```
public class StockChangeEvent {
  private String stockName;
  private double newValue;

  public StockChangeEvent(String stockName, double newValue) {
    this.stockName = stockName;
    this.newValue = newValue;
  }

  @Override
  public String toString() {
    return "StockChangeEvent [stockName=" + stockName + ", newValue=" + newValue + "]";
  }
}
```

```
@Service
public class StockService {
  @Autowired
  private ApplicationEventPublisher publisher;

  public void changeStockValue(String stockName, double value) {
    publisher.publishEvent(new StockChangeEvent(stockName, value));
  }
}
```

Subscribers

```
@Component
public class HistoryLogger {
  @Async
  @EventListener
  public void log(StockChangeEvent stockChangeEvent) {
    System.out.println("HistoryLogger received event :" + stockChangeEvent);;
  }
}
```

```
@Component
public class StockTrader {
  @Async
  @EventListener
  public void trade(StockChangeEvent stockChangeEvent) {
    System.out.println("StockTrader received event :" + stockChangeEvent);;
  }
}
```

```
@Component
public class StockNotifier {
  @Async
  @EventListener
  public void handleChangeEvent(StockChangeEvent stockChangeEvent) {
    System.out.println("StockNotifier received event :" + stockChangeEvent);;
  }
}
```

The application

```
@SpringBootApplication
@EnableAsync
public class Application implements CommandLineRunner {

  @Autowired
  private StockService stockService;

  public static void main(String[] args) {
    SpringApplication.run(Application.class, args);
  }

  @Override
  public void run(String... args) throws Exception {
    stockService.changeStockValue("AMZN", 2310.80);
    stockService.changeStockValue("MSFT", 890.45);
  }
}
```

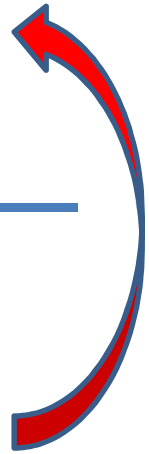
```
HistoryLogger received event :StockChangeEvent [stockName=AMZN, newValue=2310.8]
HistoryLogger received event :StockChangeEvent [stockName=MSFT, newValue=890.45]
StockTrader received event :StockChangeEvent [stockName=AMZN, newValue=2310.8]
StockTrader received event :StockChangeEvent [stockName=MSFT, newValue=890.45]
StockNotifier received event :StockChangeEvent [stockName=AMZN, newValue=2310.8]
StockNotifier received event :StockChangeEvent [stockName=MSFT, newValue=890.45]
```

Main point

- Publish-Subscribe is a very powerful technique that allows for loosely coupled publishers and subscribers.
- The nervous system of a human being is able to subscribe to the intelligence of pure consciousness.

Connecting the parts of knowledge with the wholeness of knowledge

1. With AOP you can avoid code tangling and code scattering
2. Spring events is a very easy way to implement the observer pattern.



3. **Transcendental consciousness** is the source of all activity
4. **Wholeness moving within itself:** In Unity Consciousness, one experiences that one self (rishi), and all other objects (chhandas) and the operations between oneself and all other objects (devata) are expressions of one's own Self.