AGENDA

1. SOA Recalled
2. Hi !!! Microservices
3. Welcome !!! Reactive Microservices
4. Conclude
SOA Principles

- Reusability
- Contract
- Loose Coupling
- Abstraction
- Autonomy
- Composability
- Statelessness
- Discoverability
Typical SOA Architecture
Micro Services Design Considerations

Small, and Single Responsibility

- Smart endpoints and dumb pipes
- Organized Around Business Capabilities
- Independent and Autonomous
- Decentralized Data management
- Resilient and Fault tolerant
- Automated Infrastructure Environment
# Microservices comparison with SOA

<table>
<thead>
<tr>
<th>Comparison Basis</th>
<th>Microservices</th>
<th>SOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Taxonomy</td>
<td>Functional and Infrastructure Services</td>
<td>Business, Enterprise Application and Infrastructure Services</td>
</tr>
<tr>
<td>Service Ownership and co-ordination</td>
<td>Cross functional teams organized around capabilities, Requires less co-ordination</td>
<td>Concept of Service Owners, Teams organized around Service Taxonomy and requires more coordination</td>
</tr>
<tr>
<td>Service Granularity</td>
<td>Fine Grained</td>
<td>Coarse Grained</td>
</tr>
<tr>
<td>Component Sharing</td>
<td>Share-as-little-as-possible</td>
<td>Share-as-much-as-possible</td>
</tr>
<tr>
<td>Middleware vs API Layer</td>
<td>API layer</td>
<td>Messaging Middleware</td>
</tr>
<tr>
<td>Orchestration and Choreography</td>
<td>Choreography</td>
<td>Orchestration and Choreography</td>
</tr>
<tr>
<td>Technology</td>
<td>REST</td>
<td>SOAP, REST,AQMP, WS*, JMS…</td>
</tr>
</tbody>
</table>
Benefits of Using Microservices

- Ease of Deployment
- Resilience
- Scaling
- Technology Heterogeneity
- Composability
- Optimized for Replace-ability
- Organizational Alignment
Reactive Manifesto

- Message Driven
- Responsive
- Elastic
- Resilient
## Reactive Programming Tools

### Functional Programming
- First class citizen
- Immutable
- Referential Transparency
- Pass value as parameter
- Allows Parallel execution

### Communicating Sequential Processes
- Mathematical theory of concurrency
- Free from common pitfalls
- JCSP
- Support Asyn, Non-Blocking, Message passing
- No scale outwards or failure

### Event Loops
- Event Driven Programming
- Callbacks
- Asyn and Non-Blocking
- No message passing, scale outwards
- Javascript in Node.js
- Vert.x supports distributed event bus

### Future and Promises
- Read-only handle - Future
- Write-Handle – Promise
- Completion Callback
- Supports Asyn, Non-Blocking

### Reactive Extensions
- Originated .Net
- Observer and Iterable Pattern
- Standard operations for transformation
- Supports Asyn and Non-Blocking
- No Outward scaling

### Actor Model
- Computational Entity
- Single Responsibility
- Share nothing
- Supports Asyn, Non-Blocking
- Supports Message Passing
- Scale Up and Outwards
- Built in failure mechanism
Reactive Programming (Application Services) APIs and Frameworks

JAVA 8
• LambdaExpressions
• Futures and CompletableFuture
• Stream APIs
• ExecutorService

RxJava
• Observer pattern
• Asynchronous
• Operators and Transformation
• Backpressure

Akka
• Actor Model
• Location Transparency
• Asynchronous
• Supervision
• Remoting
• BackPressure

Others
• Vert.x
• Spring 5
• Play
• Lagom
Reactive System Design

API Gateway

Location Transparency and Service Discovery

Service Communication Pattern
- Message Based
- Asynchronous and Non-Blocking
Reactive Patterns

- Fault Tolerance Pattern
  - Circuit Breaker pattern

- Message Flow Patterns
  - Saga pattern

- Flow control patterns
  - Pull-Push pattern (Back Pressure)

- State management and persistence patterns
  - Event Sourcing and CQRS pattern

- Replication
  - Active-Active
Reactive System (Application Infrastructure and Persistence) APIs and Frameworks

- **Distributed Streaming**
  - STORM
  - Others
    - SPARK
    - KAFKA
    - FLINK

- **Distributed Messaging**
  - KAFKA
  - Others
    - zeroMQ

- **Distributed Database Management**
  - Cassandra
  - Others
    - MongoDB
    - HBase
Typical Reactive Architecture

- **Application Service Layer**
  - Service A
  - Service B
  - Service C
  - Service D

- **Application Infrastructure Layer**
  - Distributed Messaging
  - Distributed Streaming

- **Application Persistence Layer**
  - Data
  - Event Store

**Other Systems**
- Mobile
- IOT
- Third Party Applications
- Internal Applications

**API Gateway**
- Workstation
- Circuit Breaker
- In Memory Cache

**Application Service Layer Frameworks and APIs**
- Java 8
- RxJava
- Reactor
- Lagom
- AKKA
- Vert.x

**Distributed Messaging**
- Kafka
- ZeroMQ

**Distributed Streaming**
- Storm
- Spark

**Application Persistence**
- Cassandra
- MongoDB

**In-Memory Cache**
- Redis
- Memcache
Thank You
MOBILE & DISRUPTIVE TECHNOLOGY SUMMIT

October 5-6, 2017
IISc, Bangalore
www.modsummit.com

Register early and get the best discounts

April 23-28, 2018
IISc, Bangalore
www.developersummit.com

GREAT INDIAN DEVELOPER SUMMIT
2018