

# Atelier 1

## Récapitulatif des bus de communication

Source : <https://fr.slideshare.net/lmedFrioukh/differents-bus-terrain>

### Caractéristiques physiques

PHYSICAL CHARACTERISTICS					Récapitulatif sur les bus de terrain
Fieldbus Name	Network Topology	Physical Media	Max. Devices (nodes)	Max. Distance	
WorldFIP	Bus	Twisted-pair, fiber	256 nodes	up to 40 Km	
LonWorks	Bus, ring, loop, star	Twisted-pair, fiber, power line	32,000/domain	2000m @ 78 kbps	
SDS	Trunkline/Dropline	Twisted-pair for signal & power	64 nodes, 126 addresses	500m (baudrate dependent)	
ControlNet	Linear, Tree, Star, or Combination Thereof	Coax, fiber	99 nodes	1000m (coax) 2 nodes 250m with 48 nodes 3km fiber; 30km fiber w/ repeaters	
CANopen	Trunkline/Dropline	Twisted Pair + optional Signal & Power	127 Nodes	25-1000m (baudrate dependent)	
Industrial Ethernet	Bus, Star, Daisy-Chain	ThinCoax, Twisted Pair, Fiber; Thick Coax (rare)	1024 nodes, expandable to more via Routers	Thin: 185m 10 Base T (Twisted Pair): Max 100m long (90 metres horizontal cable, 5m drops, 1m match) Max 4 hubs/repeaters between nodes 4Km distances w/o routers Fiber: 100 Base FX 400m 2.5 Km multi mode w/o Switches; 50 Km mono mode w/ Switches	
Modbus Plus	Linear	Twisted Pair	32 nodes per segment, 64 max	500m per segment	
Modbus RTU/ASCII	Line, star, tree Network w/ segments	Twisted Pair	250 nodes per segment	350m	
Remote I/O	Linear/Trunk	Twinaxial	32 nodes/segment	6 km	
DH+	Linear/Trunk	Twinaxial	64 nodes/segment	3 km	

PHYSICAL CHARACTERISTICS				
Fieldbus Name	Network Topology	Physical Media	Max. Devices (nodes)	Max. Distance
PROFIBUS DP/PA	Line, star & ring	Twisted-pair or fiber	127 nodes /124 slaves + 4sec. 3 rptrs) + 3 masters	100m between segments @ 12Mbaud; 24 Km fiber) (baudrate and media dependent)
INTERBUS-S	Segmented with "T" drops	Twisted-pair, fiber, and slip-ring	256 nodes	400 m/segment, 12.8 Km total
DeviceNet	Trunkline/dropline with branching	Twisted-pair for signal & power	64 nodes	500m (baudrate dependent) 6km w/ repeaters
ARCNET	Star, bus, distributed star	Coax, Twisted-pair, Fiber	255 nodes	Coax 2000 feet; Twisted pair 400 feet; Fiber 5000 Feet
AS-I	Bus, ring, tree star, or al	Two wire cable	31 slaves	100 meters, 300 with repeater
Foundation Fieldbus H1	Star or bus	Twisted-pair, fiber	240/segment, 65,000 segments	1900m @ 31.25K wire
Foundation Fieldbus HSE	Star	Twisted-pair, fiber	IP addressing - essentially unlimited	100m @ 100Mbaud twisted-pair 2000m @ 100Mbaud/fiber full duplex
IEC/ISA SP50 Fieldbus	Star or bus	Twisted-pair fiber, and radio	IS 3-7 non IS 128	1700m @ 31.25K 500M @ 5Mbps
Seriplex	Tree, loop, ring, multi-drop, star	4-wire shielded cable	500+ devices	500+ ft

## Caractéristique transport

TRANSPORT MECHANISMS						
Fieldbus Name	Communication Methods	Transmission Properties	Data Transfer Size	Arbitration Method	Error Checking	Diagnostics
PROFIBUS DP/PA	Master/slave peer to peer	DP: 9.6, 19.2, 93.75, 187.5, 500 Kbps, 1.5, 3, 6, 12 Mbps PA: 31.25 kbps	0-244 bytes	Token passing	HD4 CRC	Station, module & channel diagnostics
INTERBUS-S	Master/slave with total frame transfer	500kBits/s, full duplex	1-64 Bytes data 246 Bytes Parameter 512 bytes h.s., unlimited block	None	16-bit CRC	Segment location of CRC error and cable break
DeviceNet	Master/slave, multi-master, peer to peer	500 kbps, 250 kbps, 125 kbps	8-byte variable message with fragmentation for larger packets	Carrier-Sense Multiple Access w/ Non-Destructive Bitwise Arbitration	CRC check	Bus monitoring
ARCNET	Peer to peer	19.53K to 10M	0 to 507 bytes	Token passing	16-bit CRC	Built in Acknowledgements at Datalink layer
AS-I	Master/slave with cyclic polling	Data and power, EMI resistant	81 slaves with 4 in and 4 out	Master/slave with cyclic polling	Manchester Code, Jamming-2	Slave fault, device fault
Fieldbus Name	Communication Methods	Transmission Properties	Data Transfer Size	Arbitration Method	Error Checking	Diagnostics
Foundation Fieldbus HI	Client/server publisher/subscriber, Event notification	31.25 kbps	128 octets	Scheduler, multiple backup	16-bit CRC	Remote diagnostics, network monitors, parameter status
Foundation Fieldbus HSE	Client/Server, Publisher/Subscriber, Event Notification	100Mbps	Varies, Uses Standard TCP/IP	CSMA/CD	CRC	
IEC/ISA SP50 Fieldbus	Client/server Publisher/subscriber	31.25 kbps IS+1, 2.6, 5 Mbps	64 octets high & 256 low priority	Scheduler, okens, or master	16-bit CRC	Configurable on network management
Seriplex	Master/slave peer to peer	200 Mbps	7680/transfer	Spiral multiplexing	End of frame & echo check	Cabling problems
WorldFIP	Peer to peer	31.25 kbps, 1 & 2.5Mbps, 6 Mbps fiber	No limit, variables 128 bytes	Central arbitration	16-bit CRC, data "freshness" indicator	Device message time-out, redundant cabling
LonWorks	Master/slave peer to peer	1.25 Mbs full duplex	228 bytes	Carrier Sense, Multiple Access	16-bit CRC	Database of CRC errors and device errors
SDS	Master/slave, peer to peer, multi-cast, multi-master	1Mbps, 500 kbps, 250 kbps, 125 kbps	8-byte variable message	Carrier-Sense Multiple Access w/ Non-Destructive Bitwise Arbitration	CRC check	Bus monitoring
source : Synergistic Micro Systems, Inc.						
Fieldbus Name	Communication Methods	Transmission Properties	Data Transfer Size	Arbitration Method	Error Checking	Diagnostics
ControlNet	Producer/Consumer, Device Object Model	6 Mbps	0-510 bytes variable	CTDMA Time Slice Multiple Access	Modified CCITT with 16-bit Polynomial	Duplicate Node ID, Device, Slave Faults
CANopen	Master/slave, peer to peer, multi-cast, multi-master	10K, 20K, 50K, 125K, 250K, 500K, 800K, 1Mbps	8-byte variable message	Carrier-Sense Multiple Access w/ Non-Destructive Bitwise Arbitration	15 Bit CRC	Error Control & Emergency Messages
Industrial Ethernet	Peer to Peer	10, 100Mbps	46-1500 Bytes	CSMA/CD	CRC 32	
Modbus Plus	Peer to Peer	1Mbps	variable			
Modbus RTU/ASCII	Master/Slave	500 bps - 38.4Kbps	0-254 Bytes			
Remote I/O	Master/Slave	57.6 - 230 kbps	128 Bytes		CRC 16	none
DH+	Multi-Master, Peer<Peer	57.6 kbps	180 Bytes			none

# Questions

Sources globales : [www.wikipedia.fr](http://www.wikipedia.fr)

## Cross origin

Le Cross-Origin Resource Sharing ou CORS est un mécanisme qui permet à des ressources restreintes d'une page web d'être récupérées par un autre domaine extérieur au domaine à partir duquel la première ressource a été servie.

## ReactJS

Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page HTML à chaque changement d'état.

## Flux

Source : <https://putaindecode.io/articles/flux-qu-est-ce-que-c-est/>

Flux est unidirectionnel et comporte 4 concepts :

- les **actions**, qu'elles proviennent du serveur ou d'une interaction utilisateur ;
- le **dispatcher** dans lequel sont envoyées les actions que ce dernier transmet à *qui veut*, un peu comme un EventEmitter global ;
- les **stores**, qui sont l'équivalent du model de l'architecture MVC, ils contiennent les données, et réagissent aux actions que le dispatcher leur transmet ;
- les **views**, qui s'occupent du rendu des données dans le DOM, et de lancer des actions lorsque l'utilisateur effectue certaines actions.

## Redux

Redux est une bibliothèque open-source JavaScript de gestion d'état pour applications web. Elle est plus couramment utilisée avec des bibliothèques comme React ou Angular pour la construction d'interfaces utilisateur. C'est semblable à l'architecture Flux.

## JMS

L'interface de programmation **Java Message Service (JMS)** permet d'envoyer et de recevoir des messages de manière asynchrone entre applications ou composants Java. JMS permet d'implémenter une architecture de type MOM (message oriented middleware). Un client peut également recevoir des messages de façon synchrone dans le mode de communication point à point.

L'API JMS permet aux applications Java de s'interfacer avec des intergiciels (middleware) à messages ou MOM. Les MOM permettent des interactions entre composants applicatifs dans un cadre faiblement couplé, asynchrone et fiable.

JMS n'est pas spécifique à Springboot.

## ActiveMQ

Apache ActiveMQ est un courtier de messages open source écrit en Java avec un client Java Message Service (JMS) complet. Il fournit des "fonctionnalités d'entreprise" qui, dans ce cas, signifie favoriser la communication à partir de plusieurs clients ou serveurs. Les clients pris en charge incluent Java via JMS 1.1 ainsi que plusieurs autres clients "interlinguistiques". La communication est gérée avec des fonctionnalités telles que le clustering d'ordinateurs et la possibilité d'utiliser n'importe quelle base de données comme fournisseur de persistance JMS en plus de la mémoire virtuelle, du cache et de la persistance du journal.

Il existe un autre courtier sous l'égide d'ActiveMQ, nommé Artemis. Il est basé sur la base de code HornetQ qui a été donnée par la communauté JBoss à la communauté Apache ActiveMQ en 2015. Artemis est le courtier "nouvelle génération" d'ActiveMQ et deviendra finalement la prochaine version majeure d'ActiveMQ.

## Quels avantages proposent un bus de communication vis-à-vis de requêtes http classiques ?

Ils permettent un échange de données plus gros et plus rapides.

## Export de composants React

```
class MyClass extends Component {  
  ...  
}  
  
export default MyClass;
```

Et maintenant, vous utilisez la syntaxe suivante pour importer ce module:

```
import MyClass from './MyClass.react'
```

Si vous êtes à la recherche d'exporter plusieurs composants à partir d'un seul fichier de la déclaration ressemblerait à quelque chose comme ceci:

```
export class MyClass1 extends Component {  
  ...  
}  
  
export class MyClass2 extends Component {  
  ...  
}
```

Et maintenant, vous pouvez utiliser la syntaxe suivante pour importer ces fichiers:

```
import {MyClass1, MyClass2} from './MyClass.react'
```

# Quel est le pré-requis pour Springboot afin de pouvoir convertir automatiquement le message reçu dans un bus de communication en objet ?

Il faut créer une classe de mapping qui peut convertir cela.

## Springboot :

Source : <https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/domain/EntityScan.html>

- **@EnableJpaRepositories**  
Analyse le paquet de la classe de configuration annotée pour les référentiels Spring Data par défaut.
- **@EntityScan**  
Configure les paquets de base utilisés par la configuration automatique lors de la recherche de classes d'entités.
- **@ComponentScan**  
Configure les directives d'analyse des composants à utiliser avec les classes @Configuration. Fournit une prise en charge parallèle à l'élément <context:component-scan> de Spring XML.