Spring & Remoting

Alef Arendsen & Bram Smeets
Interface21

Spring from the source

INTERFACE21

Topics

- Discussion of remoting and RPC
- Spring's remoting architecture
- Discussion of protocols
- How to deal with security
- Advanced scenarios
- Conclusion

Spring from the source
INTERFACE21

Discussion of remoting

- Some quotes
- When to use RPC, the short version
- Caveats, things to keep in mind
- Architecture versus infrastructure

Fowler

First Law of Distributed Object Design:
Don’t distribute your objects!

Martin Fowler

Spring from the source

INTERFACE21

Waldo

"Objects that interact in a distributed system need to be dealt with in ways that are intrinsically different from objects that interact in a single address space."

Jim Waldo et al, 1994

Spring from the source

When to use RPC, the short version

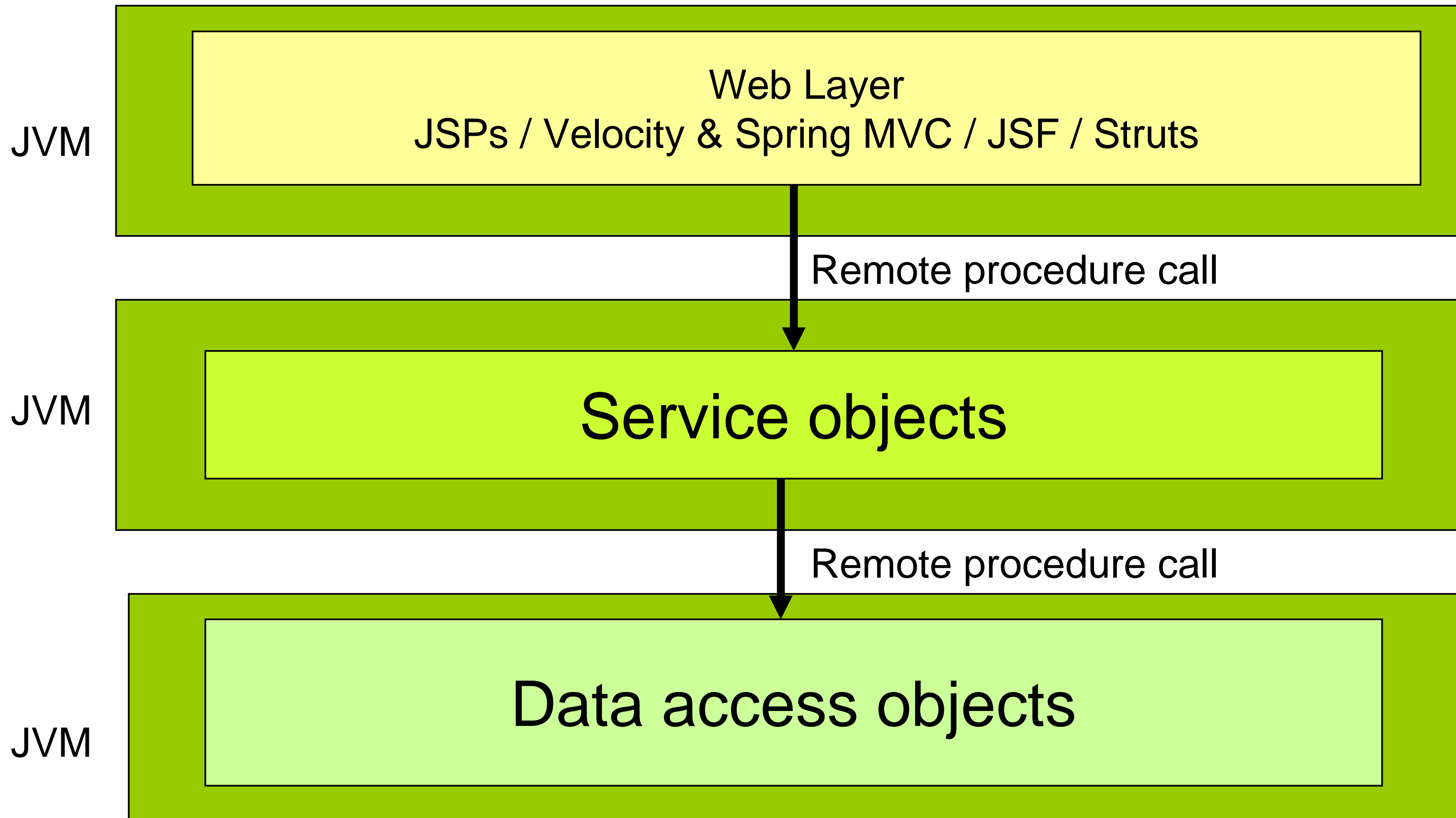
Spring
from the source

1. To support scalability within one app?

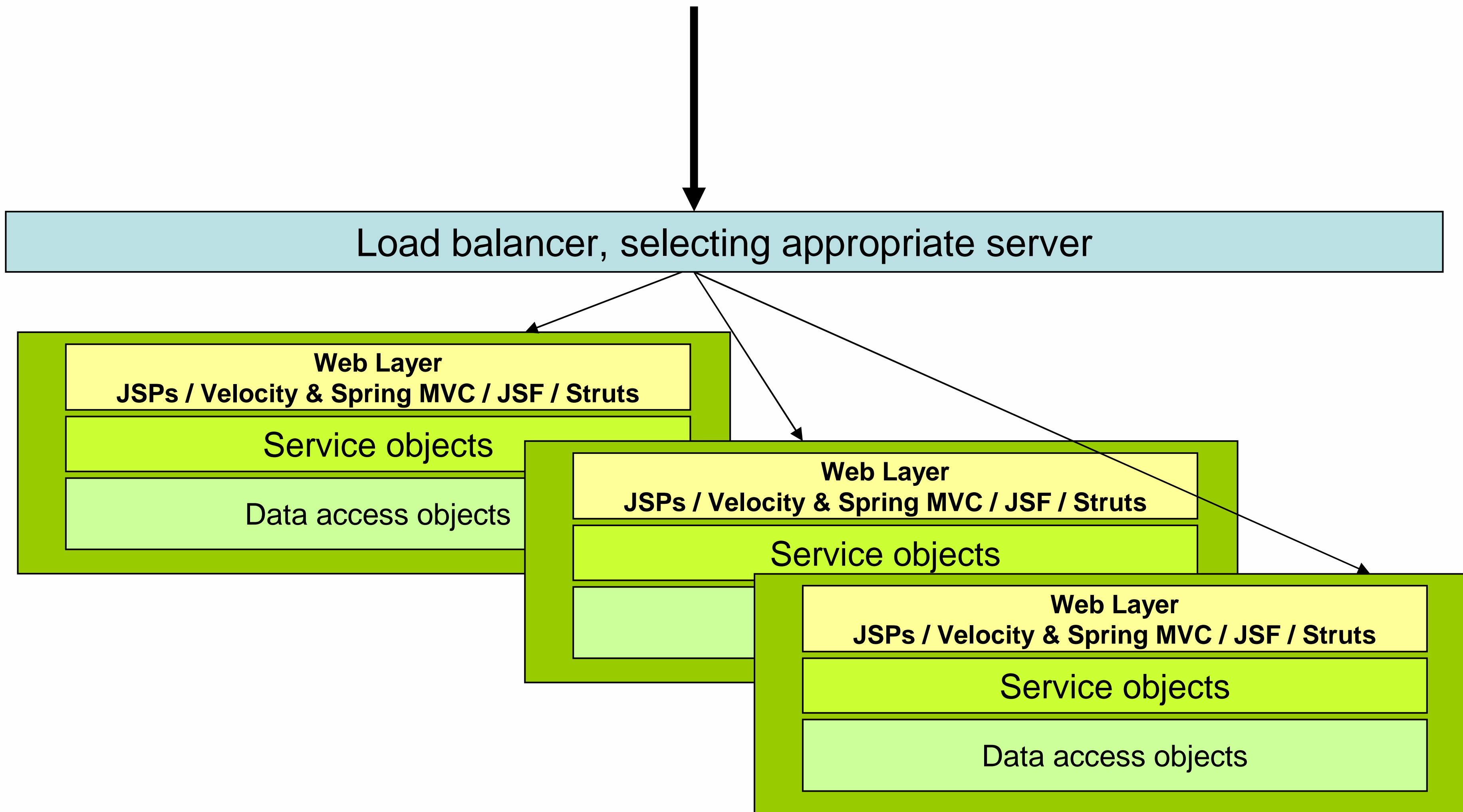
2. To integrate with remote (fat) clients?

3. To open up an application for the world?

Scalability of an application



Scalability of an application (2)



Spring from the source
INTERFACE21

To open up an application for the world

- Other issues here
  - Latency
  - Portability (PHP vs Java vs Ruby vs .NET)
  - Contract-first rather than contract-last

- Web services have more requirements
  - Security
  - Versioning
  - Asynchronous

Spring from the source
INTERFACE21
i21

Question

Spring
from the source

If we look at the typical layering from the previous slide, what defines the behavior of our application?

The service objects!

INTERFACE21

Service objects

Spring from the source

- Are central to our application
- Should be insulated
  - From the data-access technology in use
  - From the way the functionality is exposed

- Should be coded as POJOs
  - While applying infrastructural services without influencing the POJOs

Infrastructure
SHOULD NOT
influence architecture

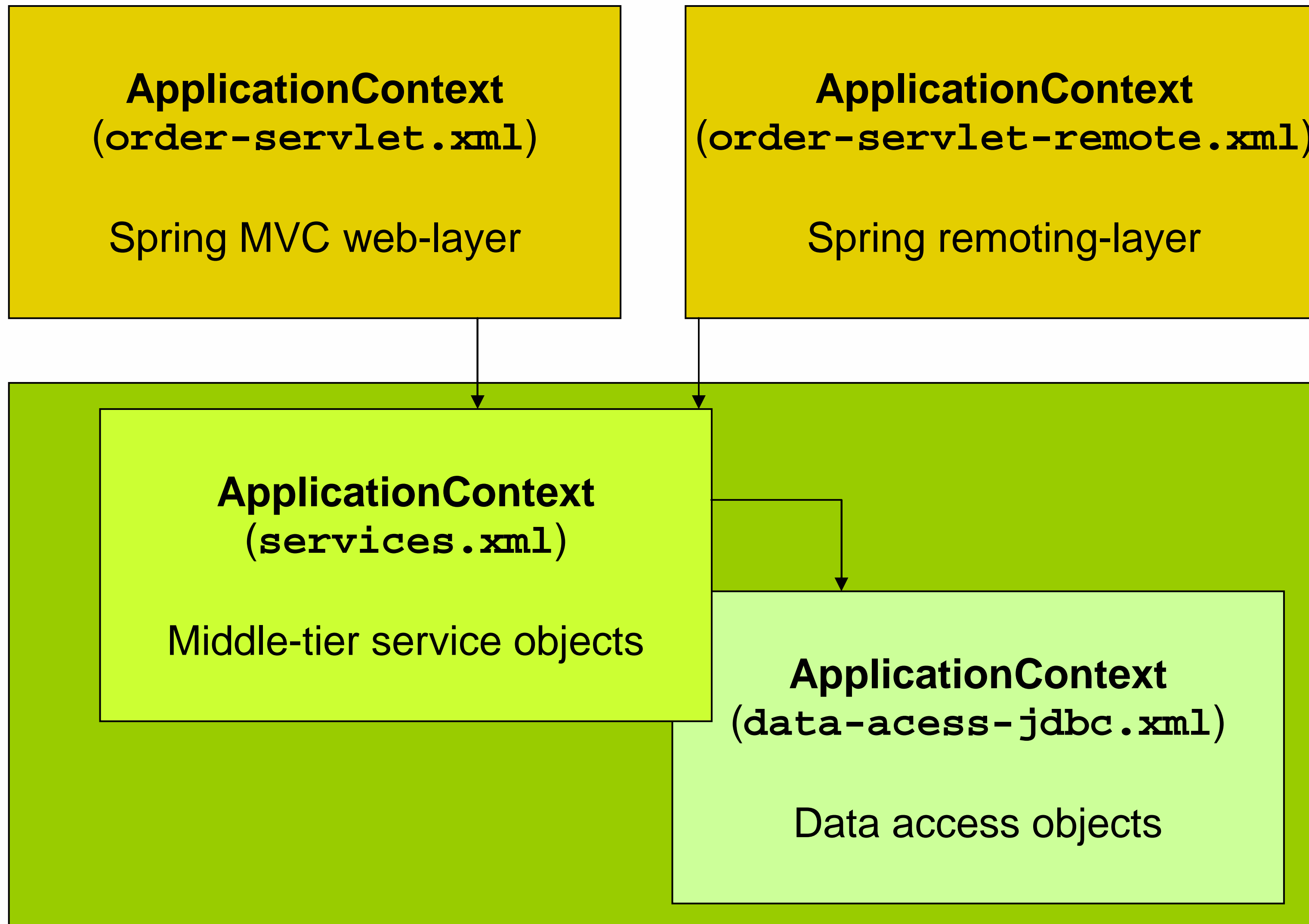
Clients of the service objects

Web MVC layer (Spring MVC, JSF, Struts)	Web services	RPC

Service objects

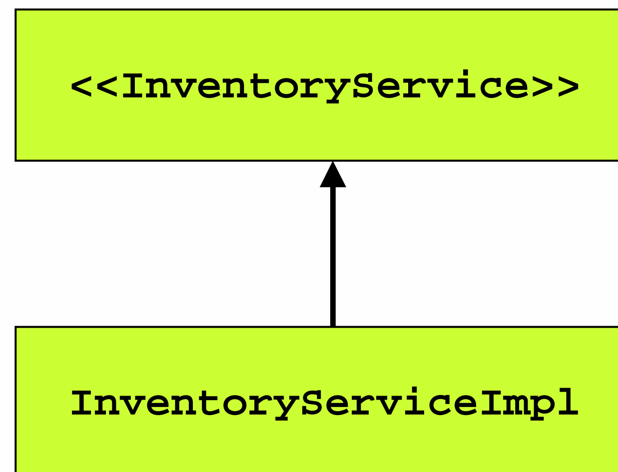
Data access objects

Infrastructure separated from architecture



Sample service

```
public interface InventoryService {
  List<String> getProductNames();
  Product getProduct(String name);
  void updateStock(Product product, int amount);
}
```



Spring Remoting

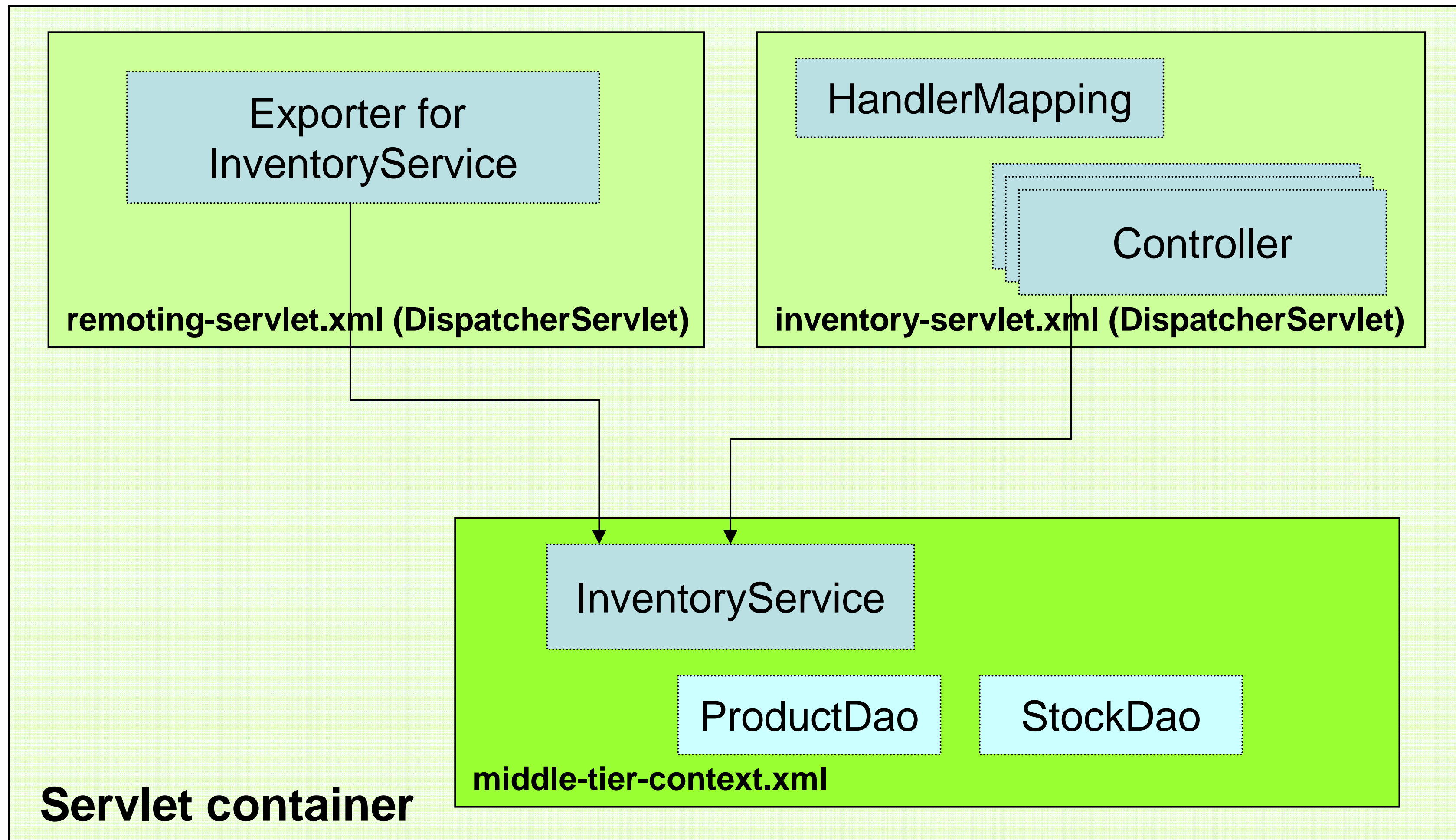
- Seamless remote access to your middle-tier services
- Marshalling and unmarshalling of
  - Method arguments
  - Return values
  - Any exceptions thrown

General concepts

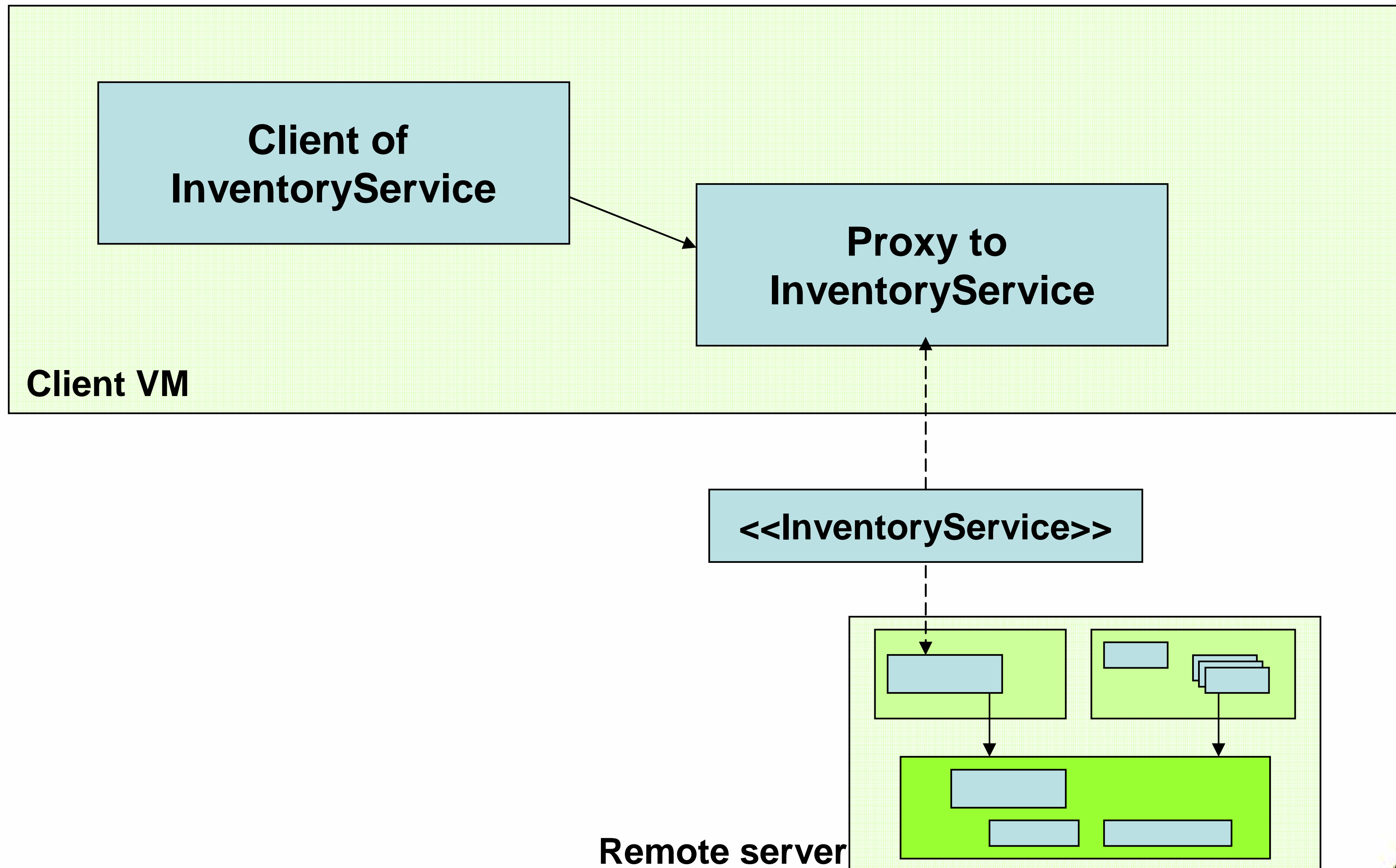
- (Remote) Exporters
  - Create endpoints for remote clients
  - Manage registries
- (Remote) ProxyFactoryBeans
  - Create proxies for remote clients to use
  - Using service interface (InventoryService)

Spring from the source
INTERFACE21

Example: remoting on the server



Example: remoting on the client



Example configuration on the server

```
<bean name="/inventoryService.html"
  class="org.sprfr.remoting.httpinvoker.HttpInvokerServiceExporter">
    <property name="serviceInterface" value="example.InventoryService"/>
    <property name="service" ref="inventoryService"/>
</bean>
```

inventory-remote-servlet.xml

```
<bean id="inventoryService" class="example.InventoryService">
  <property name="productDao" ref="productDao"/>
  <property name="stockDao" ref="stockDao"/>
</bean>


<bean id="stockDao" class="example.JdbcStockDao">
  <property name="dataSource" ref="dataSource"/>
</bean>

<bean id="productDao" class="example.JdbcProductDao">
  <property name="dataSource" ref="dataSource"/>
</bean>
```

middle-tier-context.xml

Example configuration on the client

```
<bean id="inventoryService"
  class="org.sprfr.remoting.httpinvoker.HttpInvokerProxyFactoryBean">
    <property name="serviceInterface" value="example.InventoryService"/>
    <property name="serviceUrl"
      ref="http://remotehost.com/inventoryService.html"/>
</bean>

<bean id="client" class="com.mycompany.MyClient">
  <property name="inventoryService" ref="inventoryService"/>
</bean>
```

client-context.xml

```
public class MyClient {

  private InventoryService service;

  public void setInventoryService(InventoryService service) {
    this.service = service;
  }

  // business methods using the inventory service

}
```

inventory-remote-servlet.xml

Spring from the source
INTERFACE21

Spring & Remoting

Protocols

Spring
from the source

INTERFACE21

Hessian

Spring
from the source

- Lightweight binary protocol by Caucho
- HTTP-based
- Uses custom serialization mechanism
- Support for several platforms
  - PHP / Python / C++ / C# / Ruby / Java
- Problematic when using for example Hibernate lazy loading without OSIV
- HessianServiceExporter
- HessianProxyFactoryBean

Burlap

- XML-based lightweight protocol (Caucho)
- HTTP-based
- Uses custom serialization mechanism
- Known support only for Java
- Problematic when using for example Hibernate lazy loading without OSIV

Spring from the source
INTERFACE21 i21

Spring HttpInvoker

Spring from the source

- Spring-based Java-to-Java remoting
- HTTP-based
- Uses Java serialization just like RMI
- Easy to setup
  (no HTTP tunneling as with RMI)
- HttpInvokerServiceExporter
- HttpInvokerProxyFactoryBean

RMI

- Plain-and-simple RMI
- With some additions:
  - Converts *checked* `RemoteException`s to *unchecked* `RemoteAccessException`s
  - Supports exposing plain Java interfaces
- Client support for both Spring-exposed as well as convential RMI endpoints
- RmiServiceExporter
- RmiProxyFactoryBean

Spring from the source
INTERFACE21
i21

RMI-IIOP / CORBA

- Ability to use RMI-IIOP implementation from vendor (see demo)
- Of course with the need for stubs and skeletons
- With a Spring-based client, still no need for `java.rmi.Remote`
- JndiRmiServiceExporter
- JndiRmiProxyFactoryBean

When to use what

- Both client and server under control
  - Both Spring: HttpInvoker, Hessian, Burlap
  - Low bandwidth: Hessian, Burlap (beware of complex object graphs)
  - No servlet container: RMI(-IIOP)
- Only server under control: RMI
- Multiple client platforms: Hessian (or web services)
- Need for security: HttpInvoker or RMI

Security

- Basic security for HTTP-based protocols using Servlet specification security
- Method-level security w/ Spring Security
  - Authentication
  - Method-level authorization
  - Integrates with HttpInvoker and RMI

Spring from the source

INTERFACE21

Spring from the source

Spring & Remoting

Demo

INTERFACE21