



Dirk Eismann | Herrlich & Ramuschkat GmbH

# Spring- und Flex- Integration

# Über mich

---

- Dirk Eismann
- Softwareentwickler und Consultant bei der Herrlich & Ramuschkat GmbH in Hannover
- Schwerpunkt: Projekte und Consulting im RIA Umfeld auf Basis von Flex und Java
- Flexentwicklung seit Flex 1.0 (März 2004)
- Adobe Certified Expert und Adobe Certified Instructor für Flex und AIR
- Mitgründer von [www.flexforum.de](http://www.flexforum.de)

# Agenda

---

- Überblick über Flex
  - Flex SDK und Tools
  - Deployment und Laufzeitumgebung
- Überblick BlazeDS
  - Architektur
- Spring BlazeDS Integration
  - Funktionsweise
  - Demos
- Q&A

# Das Adobe Flex SDK

- Open Source Framework zur Erstellung von Rich Internet Applications („RIA“)
  - Umfangreiche Klassenbibliothek (Controls, Container)
  - Tools (Compiler, Linker, etc.)
  - MPL Lizenz
- Aktuelle Versionen
  - Release Version 3.4.0.9271 („Flex 3“)
  - Beta Version 4.0.0.10485 („Flex 4“)
  - über SVN Zugriff auf aktuellen trunk
  - siehe auch [opensource.adobe.com](http://opensource.adobe.com)



# ActionScript 3 und MXML

- Flex-Anwendungen werden mit ActionScript 3 und MXML entwickelt
- ActionScript 3 (AS3)
  - ECMAScript 262 Edition 4 Implementierung
  - Strikt typisiert, bietet aber auch dynamische Spracheigenschaften
- MXML
  - XML-Namespace
  - Abstraktion für das in AS3 programmierte Flex Framework

# Beispiel AS3

```
import mx.controls.Button;
```

```
...
```

```
var btn:Button = new Button();
```

```
btn.label = „Klick mich!“;
```

```
btn.addEventListener(MouseEvent.CLICK,  
    buttonClickHandler)
```

```
addChild(btn);
```



Klick mich!

# Beispiel MXML

---

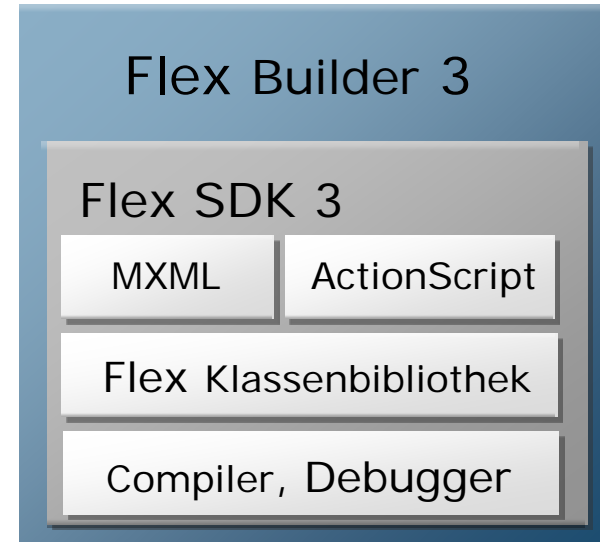
```
<mx:Button  
    label=„Klick mich“  
    click=„buttonClickHandler(event)“ />
```

A rectangular button with rounded corners, a light gray background, and a thin black border. The text "Klick mich!" is centered on the button in a dark blue font.

# Flex und Flex Builder

## Flex SDK 3

- MXML
- ActionScript 3
- Klassenbibliothek
- Compiler, Tools
- Open Source (MPL)
- kostenlos



## Flex Builder 3

- Integrierte Entwicklungsumgebung (IDE)
- Als Eclipse Plugin oder Vollinstallation (Standalone)
- Interaktiver Debugger und Profiler
- kommerziell

# Kompilierung, Deployment, Laufzeit (1/2)

- Kompilierung
  - AS3 und MXML-Klassen werden vom Compiler `mxm1c` in ActionScript-Bytecode (ABC) kompiliert
  - Bytecode liegt in SWF Datei („Flash Film“)
- Deployment
  - SWF wird auf Webserver geladen
  - Über HTML/JS wird SWF-Datei in HTML eingebunden
- Laufzeit
  - Browser fordert SWF an, Flash Player führt Bytecode aus (AVM – ActionScript Virtual Machine)
  - Geladene Flex-Anwendung kann über HTTP kommunizieren
  - SWF kann auch als AIR-Package kompiliert werden und ist als Desktopanwendung lauffähig

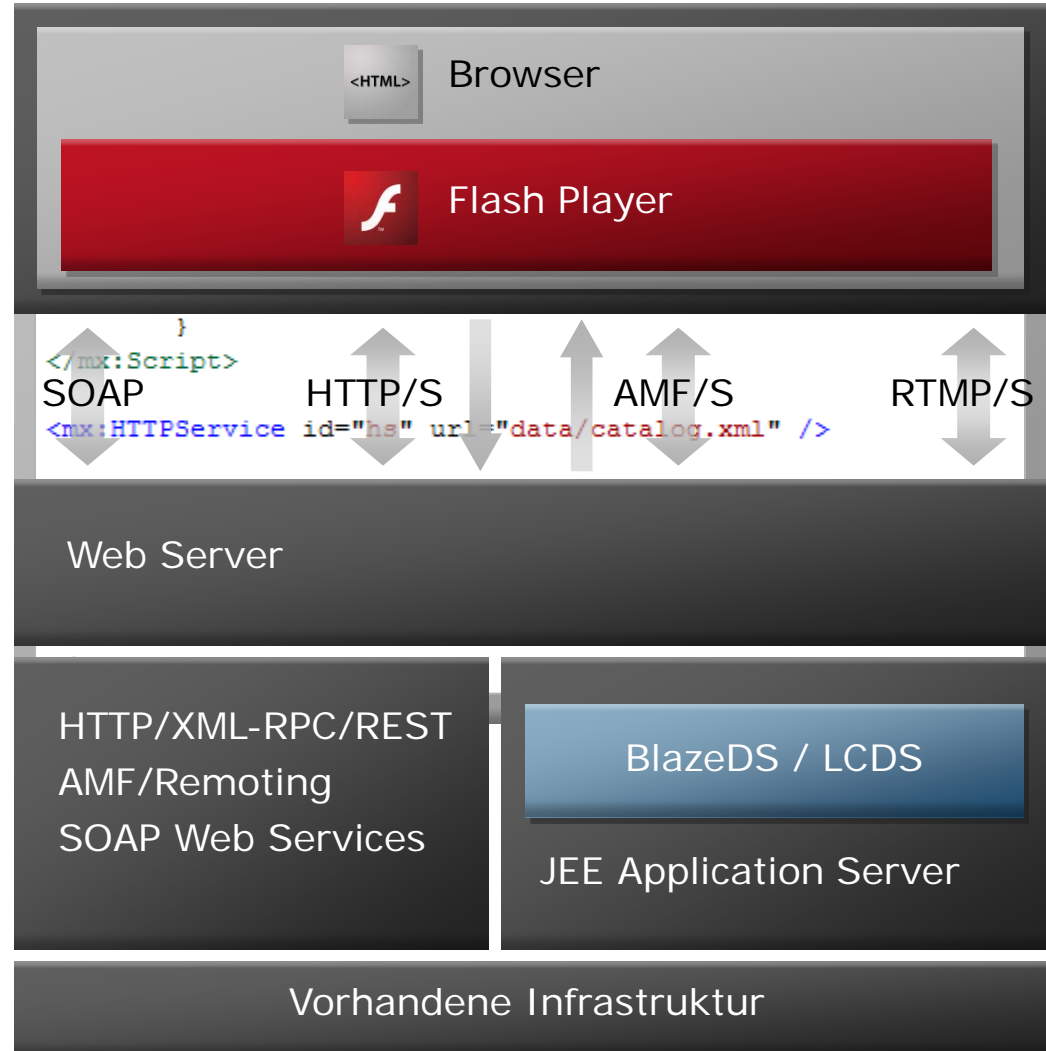


ADOBE AIR™

# Kompilierung, Deployment, Laufzeit (2/2)



Kompilierung



# BlazeDS

- Quelloffene, Java-basierte Remoting und Messaging Lösung für RIAs
- Performante Datenübertragung zwischen Java und Flex
- Leichtgewichtig
  - BlazeDS besteht aus 5 JARs
  - ~ 840 KB
- Einfache Integration über web.xml



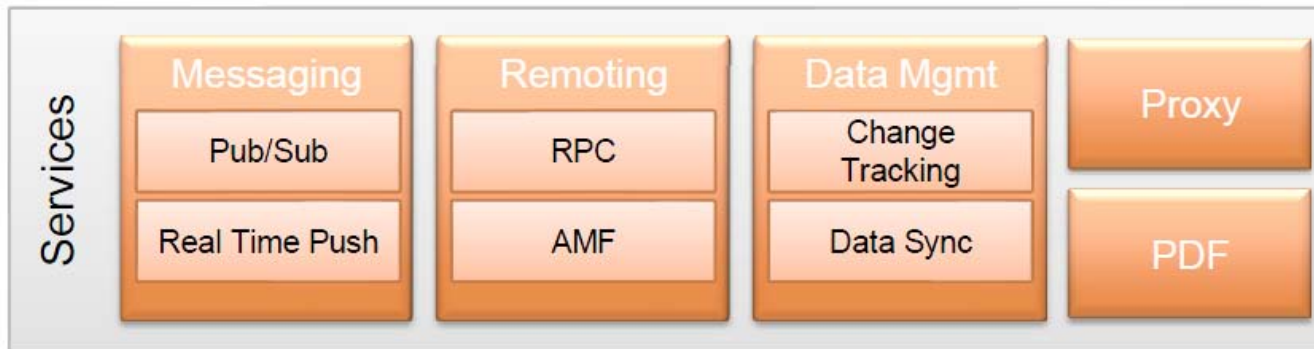
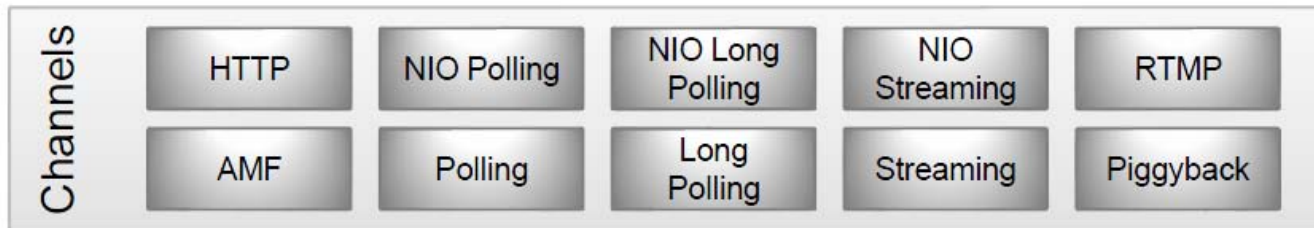
# BlazeDS - Remoting

- Vergleichbar mit RMI, verwendet aber HTTP(S)
- Erlaubt Flex-Clients den Aufruf entfernter Java-Methoden
- Daten werden typischer im AMF-Format („Action Message Format“) übertragen
- Massive Geschwindigkeitsvorteile gegenüber z.B. SOAP

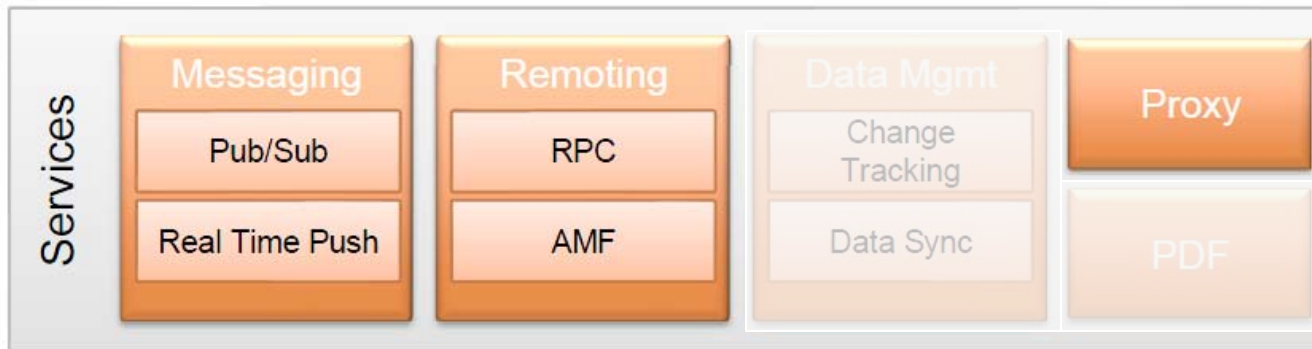
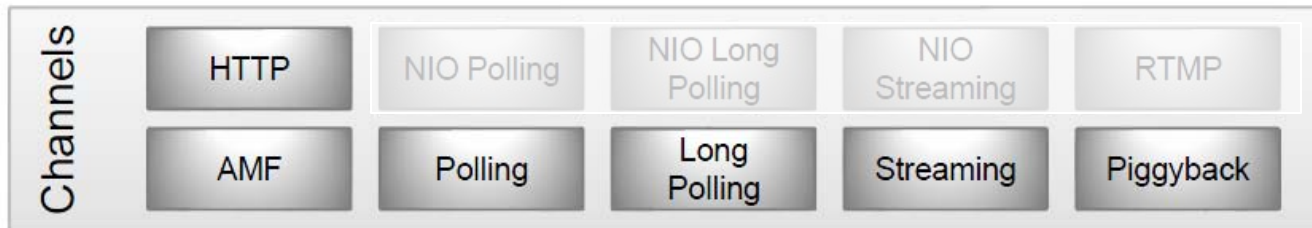
# BlazeDS - Messaging

- Ermöglicht HTTP-basierten „Server Push“
- Bietet verschiedene Kommunikationswege
  - Polling
  - Long-polling
  - Streaming
  - Piggyback
- Daten werden typischer im AMF-Format („Action Message Format“) übertragen
- Ganz nett, skaliert aber wegen Verwendung der Servlet API schlecht

# LiveCycle DS vs BlazeDS



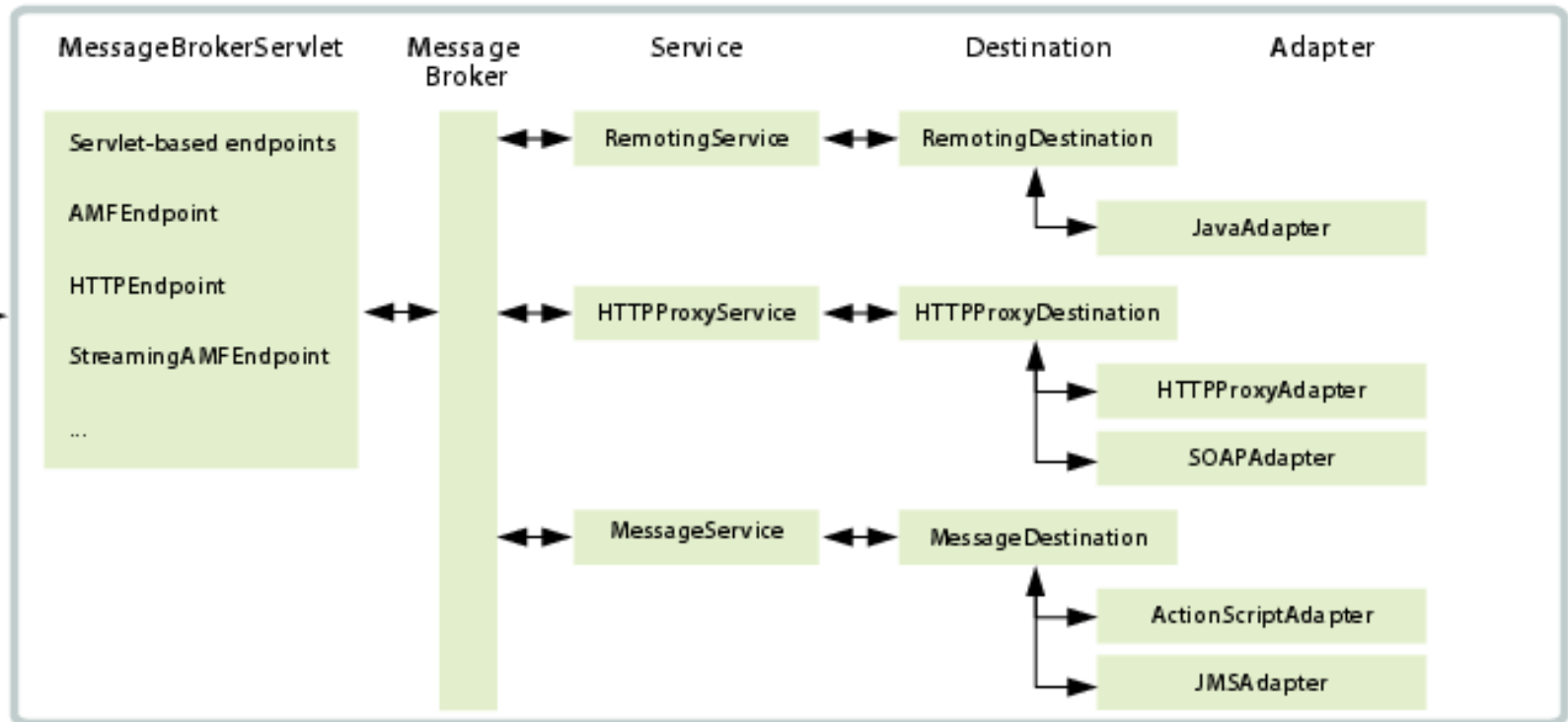
# LiveCycle DS vs BlazeDS



# BlazeDS Architektur (1/2)

- Channel
  - Definiert URL, Port und AMF-Format
- MessageBroker
  - Delegiert AMF-Anfragen
- Service
  - Stellt Dienst-API bereit (z.B. für Remoting)
- Destination
  - Verbindet Channel mit Endpunkt (Java-Klasse)
- Adapter
  - Ruft Methoden auf Endpunkten auf

# BlazeDS Architektur (2/2)



Quelle: [http://livedocs.adobe.com/blazeds/1/blazeds\\_devguide/images/blazeds\\_server\\_arch.png](http://livedocs.adobe.com/blazeds/1/blazeds_devguide/images/blazeds_server_arch.png)

---

# BlazeDS Demo

# Konfiguration - Nachteile

- Umständliche Konfiguration von BlazeDS über mehrere XML-Dateien
  - WEB-INF/flex/services-config.xml
  - WEB-INF/flex/messaging-config.xml
  - WEB-INF/flex/proxy-config.xml
  - WEB-INF/flex/remoting-config.xml
- Integration mit Spring bisher nur über Workarounds
  - Jeff Vrooms SpringFactory

# Spring BlazeDS - ganz einfach

- Spring BlazeDS Integration vereinfacht die Integration von Flex und Spring
  - Konfiguration von BlazeDS über ApplicationContext oder Annotations
  - Keine zusätzlichen Config-Dateien mehr!
  - Unterstützt Remoting und Messaging
  - Endpunkte lassen sich über Spring Security absichern
- Aktuelle Version 1.0.1

# Konfiguration

- MessageBroker wird von Spring verwaltet
- Standardvorgehen
  - DispatcherServlet / Spring Web MVC lädt Spring Context
  - /messagebroker/\* Aufrufe werden über URL Mapping an den MessageBroker gereicht
  - Beans können als „Remote Objects“ für Flex-Clients exportiert werden
- Vorteil
  - Springkonforme Konfiguration

# Auszug aus web.xml

```
<servlet>
  <servlet-name>SpringDispatcherServlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/web-application-context.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>SpringDispatcherServlet</servlet-name>
  <url-pattern>/messagebroker/*</url-pattern>
</servlet-mapping>
```

## Konfiguration DispatcherServlet

# Auszug aus web-application-context.xml

```
<!-- Spring-managed BlazeDS MessageBroker -->
<flex:message-broker/>

<!-- Exportieren einer Bean -->
<flex:remoting-destination ref="employeeService"/>

<!-- Bean -->
<bean id="employeeService" class="de.flexperten.demo.service.EmployeeService">
    <property name="employeeDAO" ref="employeeDAO"/>
</bean>

<bean id="employeeDAO" class="de.flexperten.demo.dao.EmployeeDAO"/>
```

Konfiguration Spring Context

# Standardkonfiguration

- Name der Destination = Name der exportierten Bean
  - Übersteuerbar über destination-id property
- Verwendeter Adapter = JavaAdapter
  - Übersteuerbar über service-adapter property
  - So lassen sich 3rd-Party Adapter verwenden
- Alle public Methoden der Bean sind remote verfügbar
  - Über exclude-methods bzw. @RemotingExclude steuerbar

---

# Spring BlazeDS Demos

# Spring BlazeDS - weitere Features

- AMF Message Interceptors
  - Pre- und Postprocessing von AMF Nachrichten
- Eigene Exception Translators
  - Backendexceptions auf Clientfehlermeldungen abbilden
- Spring Security Unterstützung
  - Endpunkte und Services absichern
- Anbindung an JMS und Spring Integration Message Destinations

# Spring BlazeDS - Gotchas

- LazyInitializingException
  - Der Serializer von BlazeDS kann mit Hibernate Proxies nicht umgehen
  - Spring BlazeDS 1.5 (Q4 2009) soll das können...
  - Workaround: Gilead verwenden (s.u.)
- Datentypen AS3 != Java
  - Nicht initialisierte Number in AS3 hat den Wert NaN, wird aber als Integer(0) deserialisiert
  - Abhilfe: eigenen BlazeDS BeanProxy verwenden

# /etc

- Spring BlazeDS Integration Home  
<http://www.springsource.org/spring-flex>
- Gilead  
<http://noon.gilead.free.fr/gilead>
- Flexforum.de  
<http://www.flexforum.de>
- Java CODE CAMP-Flex
  - 07.-09.12. in Köln
  - Mit Florian Müller und Lars Röwekamp

# Q&A

---

- Noch Fragen?

---

Vielen Dank für Eure  
Aufmerksamkeit!