GlassFish Project Web Services Stack "Metro" : Easy to Use, Robust, and High-Performance

Jitendra Kotamraju
Marek Potociar
Sun Microsystems

TS-6658

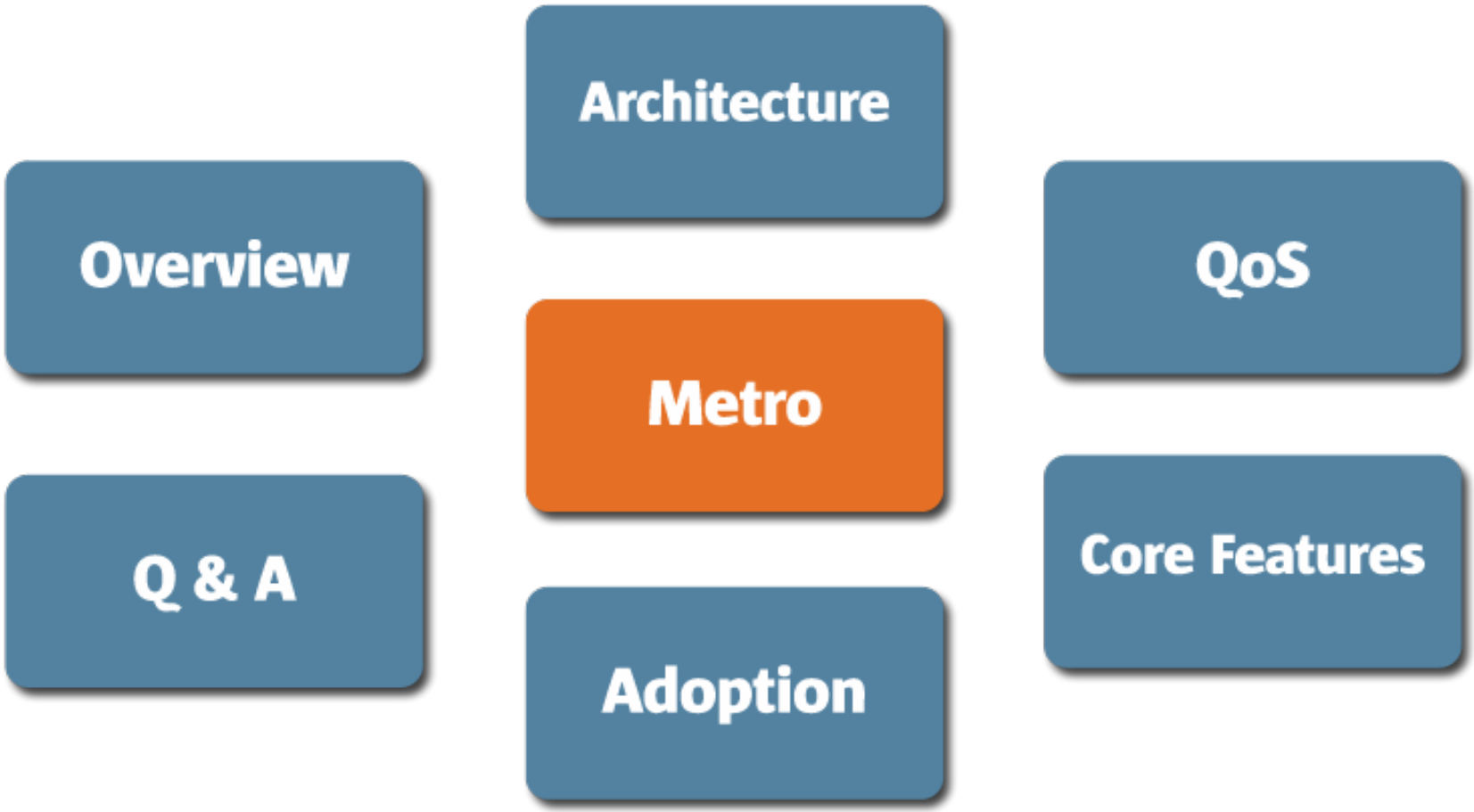
Java

Sun microsystems

Learn how to leverage latest features of the Metro Web Services and achieve interoperability with your Web Services

GOAL

Agenda


Architecture
Overview
QoS
Metro
Core Features
Q & A
Adoption

What is Metro?

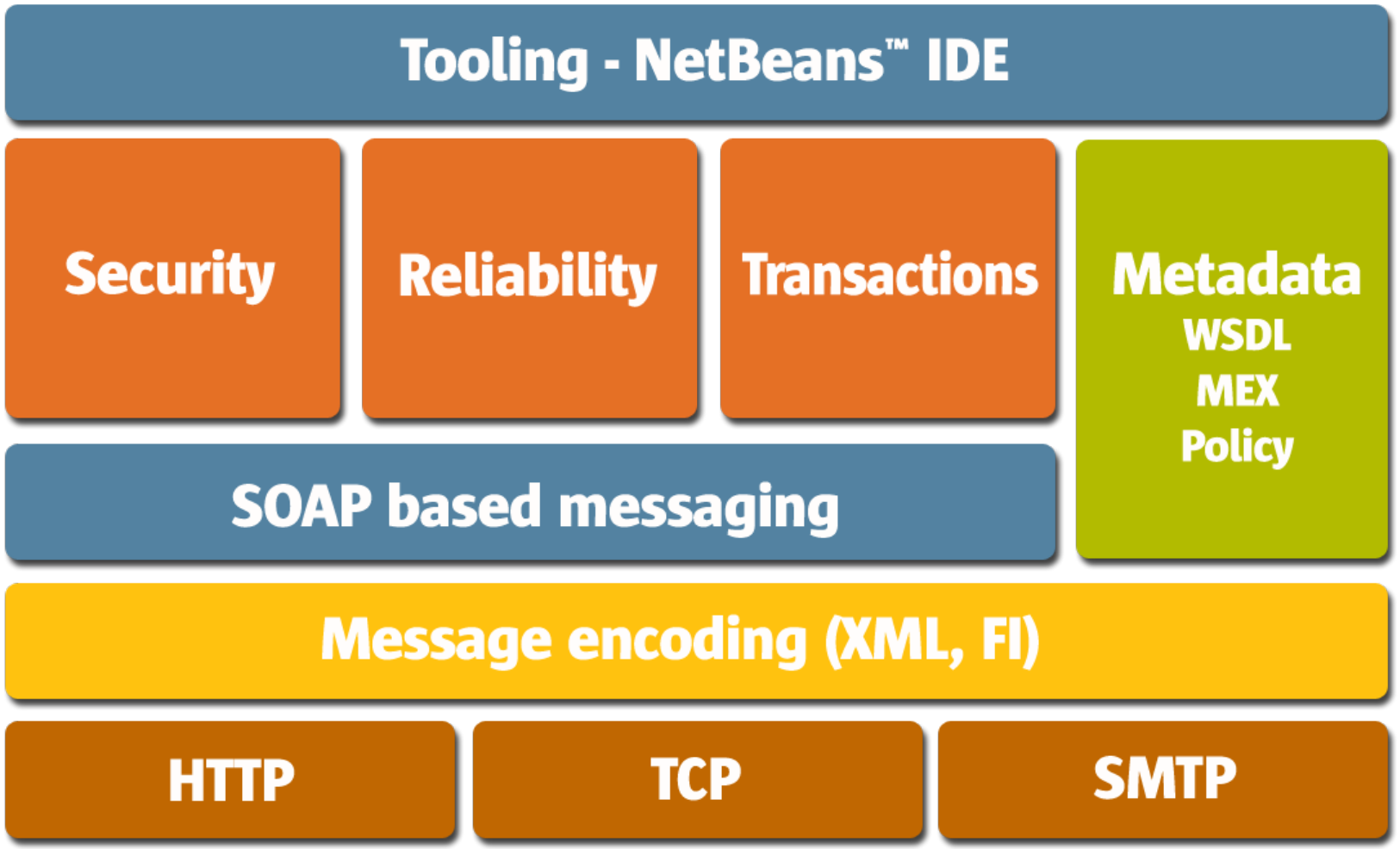
Metro is one stop shop for all your Web services needs.

- Web services stack
- Part of GlassFish™ project community
- Production-quality
- High-performance

Metro = JAX-WS RI + WSIT/Tango

- JAX-WS RI
  - Provides core Web services support
  - Extensible and pluggable architecture
- WSIT/Tango
  - Provides support for QoS
    - Security, Reliability, Transactions
  - Implementation of WS-* specifications
    - Interoperability with .NET 3.x

Metro Architecture



Metro Features - JAX-WS

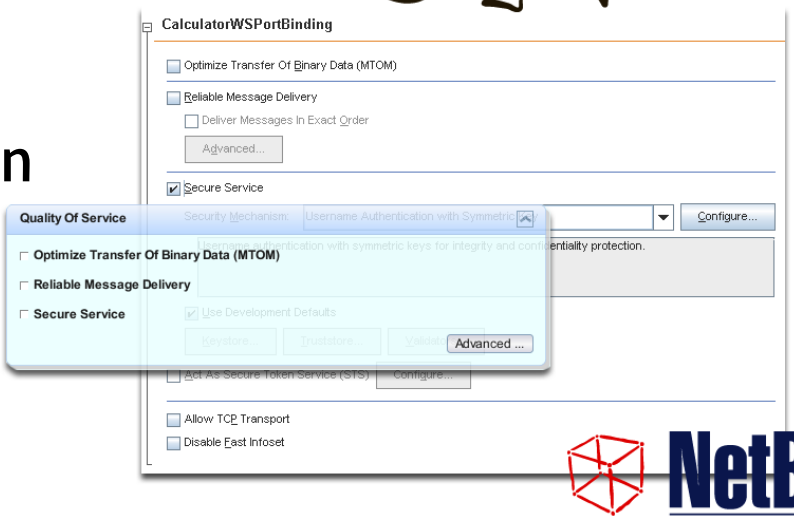
- JAX-WS 2.1: Easy to use Web services API
  - New addition : Web Services Addressing 1.0 support
- Embrace POJO concepts via annotations
  - Descriptor-free programming
- Encoding, Protocol and Transport Independence
- Integrated Java Architecture for XML Binding (JAXB)
  - Java™ platform API
  - 100% XML Schema Support
- Optimizing Communication
  - MTOM/XOP (W3C), FastInfoset (ITU-T/ISO)

Metro Features – WSIT (Project Tango)

- Bootstrapping Communication
  - WS-MetadataExchange, WS-Policy
- Providing Reliability
- Enabling Atomic Transactions
- Securing Communication
- Transparent to application

WSIT Programming model

- No runtime APIs
- Components developed using JAX-WS and EJB APIs
- Quality of service specified in configuration file
- Configuration file produced by NetBeans™ module
  - optionally can be written by hand



NetBeans

Configuring QoS

DEMO

Metro Core

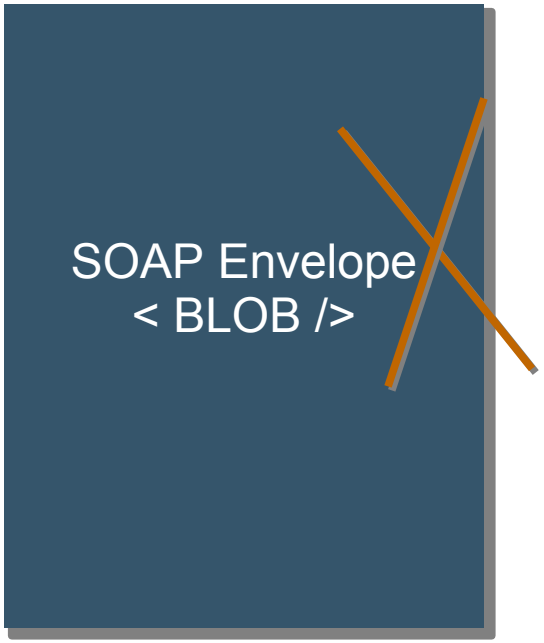
- Stable Code base with years of engineering
  - JAX-RPC RI/JWSDP --> JAX-WS RI --> Metro Core
- Built for compatibility, extensibility and performance
- Is **much** more than JAX-WS API
- Battle-tested
  - Java EE 5, Glassfish, BEA WebLogic 10, TmaxSoft JEUS6, OpenESB, ...
  - Sun Java SE 6, IBM JDK 6, ..
- Good tools support
  - IDEs, ant, maven

Metro Core Newer Features

- Efficient Attachment Handling
- Schema Validation
- Simplified “Starting from Java”
- Efficient Handlers
- Additional Headers Support

Efficient Attachment Handling

- BLOB in XML is expensive

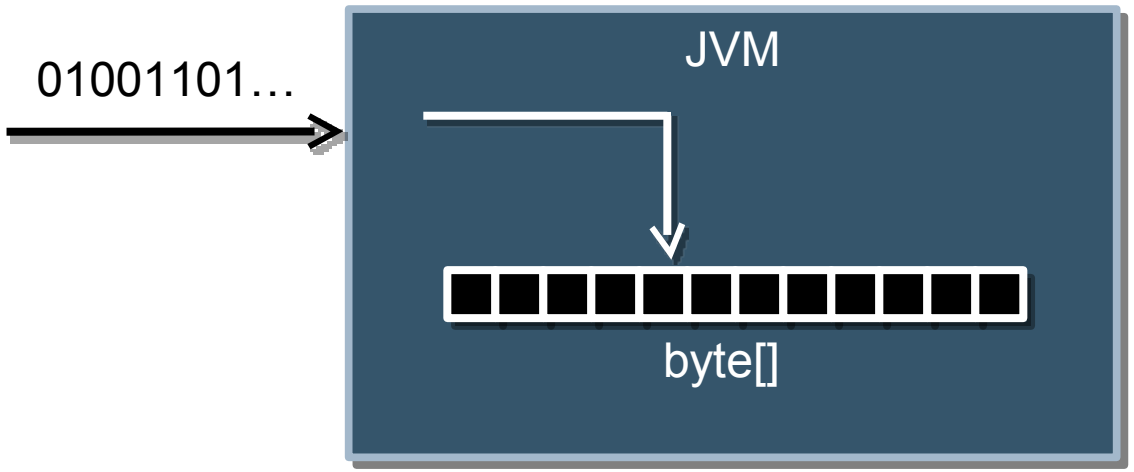


Efficient Attachment Handling

- BLOB in XML is expensive
- You want to send the BLOB as SOAP attachment

Efficient Attachment Handling

- Default binding for binary data is byte[]
  - SEI would be : echo(String name, byte[] blob)
  - Large heap to store the BLOB

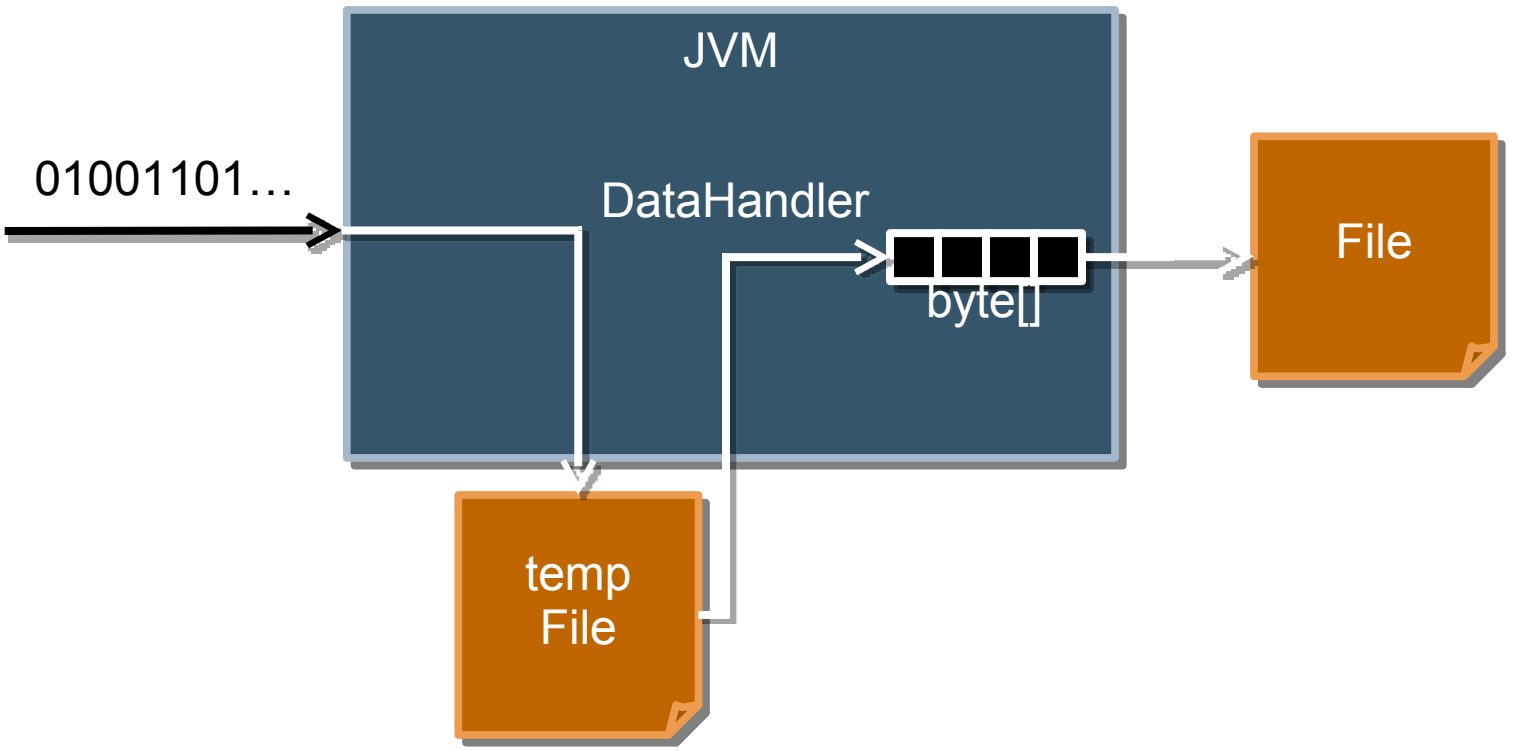


Efficient Attachment Handling

- Use “DataHandler”
  - No longer necessary to hold data in memory
  - Loading can be delayed
  - Plus some convenience methods

```
class DataHandler {
    InputStream getInputStream()
    String getContentType()
    Object getContent()
    ...
}
```

Efficient Attachment Handling



Efficient Attachment Handling

- Metro needs to keep temp file around
  - Because DataHandler allows multiple reads
- Use Metro extension “StreamingDataHandler”
  - application can grab temporary file directly
  - application can say read is done only once
  - Allows streaming without storing the data in a temp file in some cases

```
class StreamingDataHandler extends DataHandler {
    void moveTo(File)
    InputStream readOnce()
    ...
}
```

Streaming Attachments

DEMO

Schema Validation

```
@WebService
class AddService {
  int add(int num1, int num2) {...}
}
```

```
<simpleType name='myint'>
  <restriction base='int'>
    <totalDigits value='4'/>
  </restriction>
</simpleType>
```

- add(12345, 20) works fine !!

Server-Side Schema Validation

```
@SchemaValidation
@WebService
class AddService {
  int add(int num1, int num2) {...}
}

<simpleType name='myint'>
  <restriction base='int'>
    <totalDigits value='4'/>
  </restriction>
</simpleType>
```

Client-Side Schema Validation

```
proxy = service.getAddPort(new SchemaValidationFeature());
```

Client-Side Schema Validation

```
proxy = service.getAddPort(new SchemaValidationFeature());

try {
  proxy.add(12345, 20);
} catch(WebServiceException we) {
  // fails client-side validation
  // request doesn't go to server
}
```

Client-Side Schema Validation

```
proxy = service.getAddPort(new SchemaValidationFeature());

try {
  proxy.add(12345, 20);
} catch(WebServiceException we) {
  // fails client-side validation
  // request doesn't go to server
}

// passes client-side validation.
// request is sent to the server
proxy.add(1234, 20);
```

Simplified "Starting from Java"

- Building a sample service starting from Java platform

```
@WebService
class AddService {
  int add(int num1, int num2){...}

  static void main(String ... args) {
    Endpoint.publish(
      "http://localhost:8181/add",new AddService());
    }
}

$ javac pkg/AddService.java
$ wsgen -d gen pkg.AddService
$ ls gen/pkg/jaxws
  Add.class AddResponse.class
$ java pkg.AddService
```

Simplified "Starting from Java"

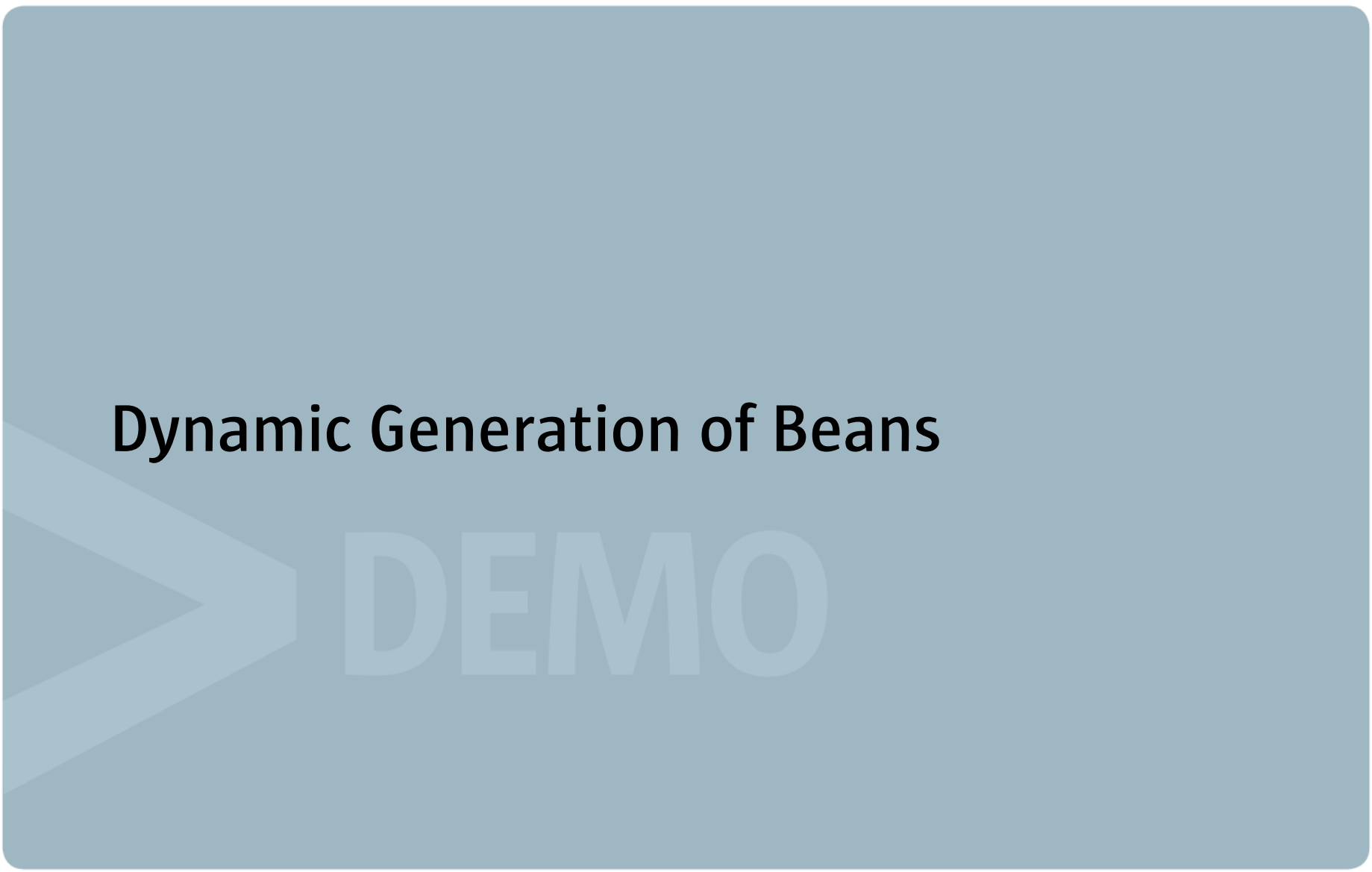
- Now building services starting from Java is easier

```
@WebService
class AddService {
  int add(int num1, int num2){...}

  static void main(String ... args) {
    Endpoint.publish(
      "http://localhost:8181/add",new AddService());
    }
}

$ javac pkg/AddService.java
$ java pkg.AddService
```

Dynamic Generation of Beans



Efficient Handlers

- "Handler" for WS is like filter for Servlets
- JAX-WS spec defined SOAPHandler, LogicalHandler
  - Fixed data models
  - can be slow
- In Metro, we have our own handler type

Efficient Handlers

```
class AuthHandler implements MessageHandler {
```

Efficient Handlers

```
class AuthHandler implements MessageHandler {

  boolean handleMessage(MessageHandlerContext mhc) {
    Message m = mhc.getMessage();
```

Efficient Handlers

```
class AuthHandler implements MessageHandler {

  boolean handleMessage(MessageHandlerContext mhc) {
    Message m = mhc.getMessage();
    // Add SOAP headers for authentication
    Header h1 = Headers.create("...", "user");
    Header h2 = Headers.create("...", "pass");
    m.getHeadersList().add(h1);
    m.getHeadersList().add(h2);
    return true;
  }

  ...
}
```

Efficient Handlers - Benefits

- Much reuse from the spec
  - Configuration, programming model etc
  - Less to learn
- Fast
  - ... provided that there is no data conversion
- Easier to program
  - StAX API, JAXB support
  - Conveninent methods with SOAP semantics

Additional Headers Support

- Spec defines SEI to be generated from porttype
- Binding could add additional headers
- wsimport -XadditionalHeaders

Metro Quality Assurance

- Interoperability testing with .NET endpoints
- Using different platforms
- Using different StAX parsers
- Using Different containers
- Extensive test case coverage

		JDK6.0	JDK5.0_12
jaxws-ri-2.1.3	linux		
	windows		
	solaris		
jaxws-ri-2.1.4	linux		
	windows		
	solaris		

Metro Community

- Lot of Adoption
  - Java EE 5 SDK, GlassFish, BEA WebLogic 10, TmaxSoft JEUS6, JBOSS AS, ...
  - Sun Java SE 6, IBM JDK 6, ..
  - OpenSSO, OpenESB, Wiseman, ...
- Very active mailing lists and forum
  - Even our Microsoft colleagues hang out there !
- Hosted extensions
  - Spring, JSON, SMTP transport etc.
  - Recent additions - DIME plugin, Grails plugin
- Be active!
  - Vote on issues
  - Provide patches
  - Review documentation
  - Participate

Roadmap

- Metro 1.2 – May '08
  - Simplified “Starting from Java” web services
  - Interoperability with .NET 3.5 (EA)
  - Unified 196-security (GF only)
- Metro 1.3 – June '08
  - Interoperability with .NET 3.5 (FCS)
  - Kerberos support & other security features
  - SOAP/TCP on Tomcat
  - Code consolidation
- Metro 2.0 – Q4 '08
  - Jersey (REST stack) integration
  - New features (High availability, Dynamic policies, Profiles, IDE APIs…)
  - Under planning
- More Info
  - https://metro.dev.java.net/discover/roadmap.html

Metro Summary

- Full-featured WS stack
- Many core advanced features
- Interoperability (with Microsoft .NET 3.x)
- Quality of Service
- Running on GlassFish and Tomcat containers

For More Information

- http://metro.dev.java.net
- http://glassfish.dev.java.net/
- http://jax-ws.dev.java.net
- http://forums.java.net/jive/forum.jspa?forumID=46

- users@metro.dev.java.net

THANK YOU

Jitendra Kotamraju
Marek Potociar
Sun Microsystems

TS-6658