What's new in Spring?

Craig Walls
craig@habuma.com
Twitter: @habuma @springsocial
http://github.com/habuma

WebSocket/STOMP

What is WebSocket?

Full duplex communication channel

Enables the server to talk to the browser*

* What you **really** want to do

How Spring supports WebSocket

Low-level API

SockJS support

Higher-level Spring MVC-based API

Messaging template

Enabling WebSocket support

```
@EnableWebSocket
public class WebSocketConfig implements WebSocketConfigurer {

    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(marcoHandler(), "/marco");
    }

    @Bean
    public MarcoHandler marcoHandler() {
        return new MarcoHandler();
    }

}
```

Handling messages

```
public class MarcoHandler extends AbstractWebSocketHandler {

  private static final Logger logger = LoggerFactory
      .getLogger(MarcoHandler.class);

  protected void handleTextMessage(WebSocketSession session, TextMessage message)
      throws Exception {
    logger.info("Received message: " + message.getPayload());
    Thread.sleep(2000);
    session.sendMessage(new TextMessage("Polo!"));
  }

  public void afterConnectionEstablished(WebSocketSession session)
      throws Exception {
    logger.info("Connection established");
  }

  @Override
  public void afterConnectionClosed(
      WebSocketSession session, CloseStatus status) throws Exception {
    logger.info("Connection closed. Status: " + status);
  }
}
```

A simple JavaScript client

```
var url = ‘ws://localhost:8080/websocket/marco’;
var sock = new WebSocket(url);
var counter = 0;

sock.onopen = function() {
  console.log('Opening');
  sayMarco();
};

sock.onmessage = function(e) {
  console.log('Received message: ', e.data);
  $('#output').append("<b>Received: " + e.data + "</b><br/>")
  setTimeout(function(){sayMarco()}, 2000);
};

sock.onclose = function() {
  console.log('Closing');
};

function sayMarco() {
  sock.send(JSON.stringify({ id: counter++, message : 'Marco!' }));
  $('#output').append("<b>Send: Marco!</b><br/>")
}
```

The problem with WebSocket

It's not ubiquitous

Browser support iffy

Server support iffy

Proxy server support iffy

Enabling SockJS

```
@EnableWebSocket
public class WebSocketConfig implements WebSocketConfigurer {

    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(marcoHandler(), "/marco").withSockJS();
    }

    @Bean
    public MarcoHandler marcoHandler() {
        return new MarcoHandler();
    }

}
```

A SockJS-enabled JavaScript client

```
var url = ‘marco’;
var sock = new SockJS(url);
var counter = 0;

sock.onopen = function() {
  console.log('Opening');
  sayMarco();
};

sock.onmessage = function(e) {
  console.log('Received message: ', e.data);
  $('#output').append("<b>Received: " + e.data + "</b><br/>")
  setTimeout(function(){sayMarco()}, 2000);
};

sock.onclose = function() {
  console.log('Closing');
};

function sayMarco() {
  sock.send(JSON.stringify({ id: counter++, message : 'Marco!' }));
  $('#output').append("<b>Send: Marco!</b><br/>")
}
```

More problems with WebSocket

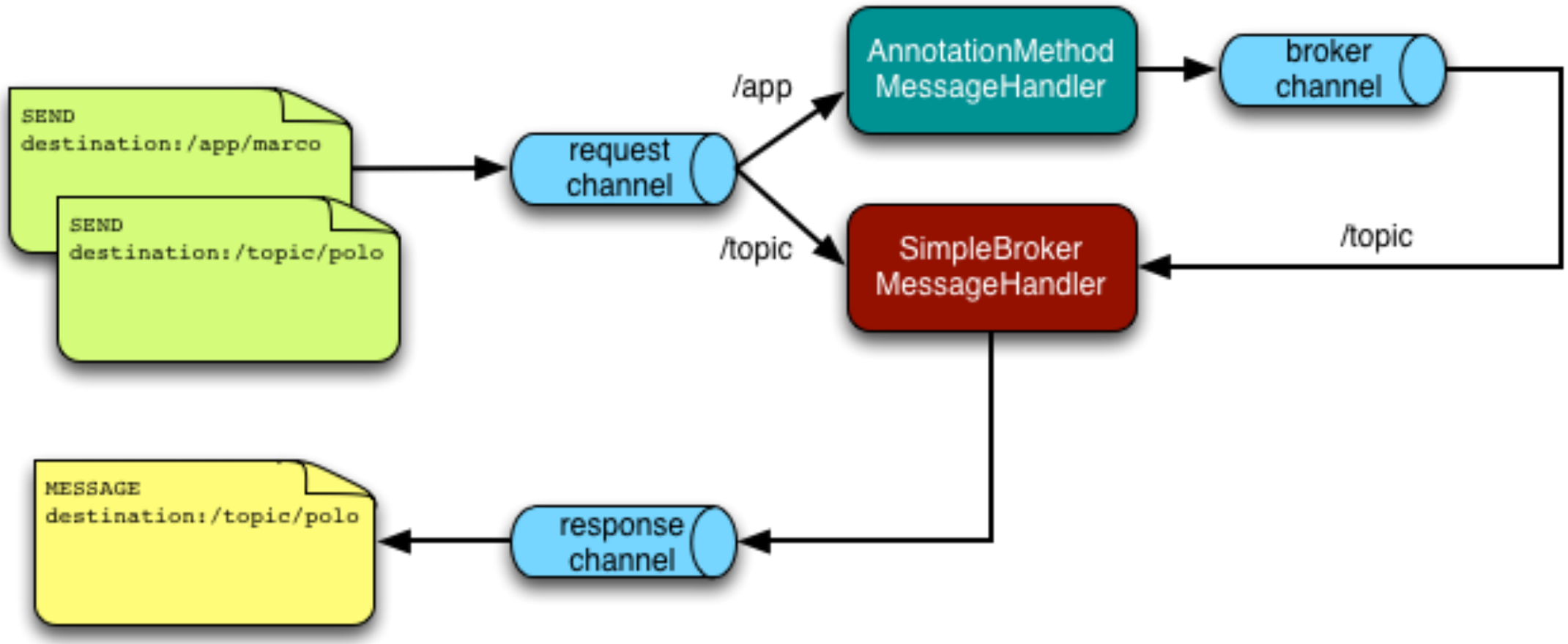
Too low-level

No messaging semantics

Enabling STOMP

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketStompConfig extends AbstractWebSocketMessageBrokerConfigurer {

  @Override
  public void registerStompEndpoints(StompEndpointRegistry registry) {
    registry.addEndpoint("/marcopolo").withSockJS();
  }

  @Override
  public void configureMessageBroker(MessageBrokerRegistry registry) {
    registry.enableSimpleBroker("/queue/", "/topic/");
    registry.setApplicationDestinationPrefixes("/app");
  }

}
```

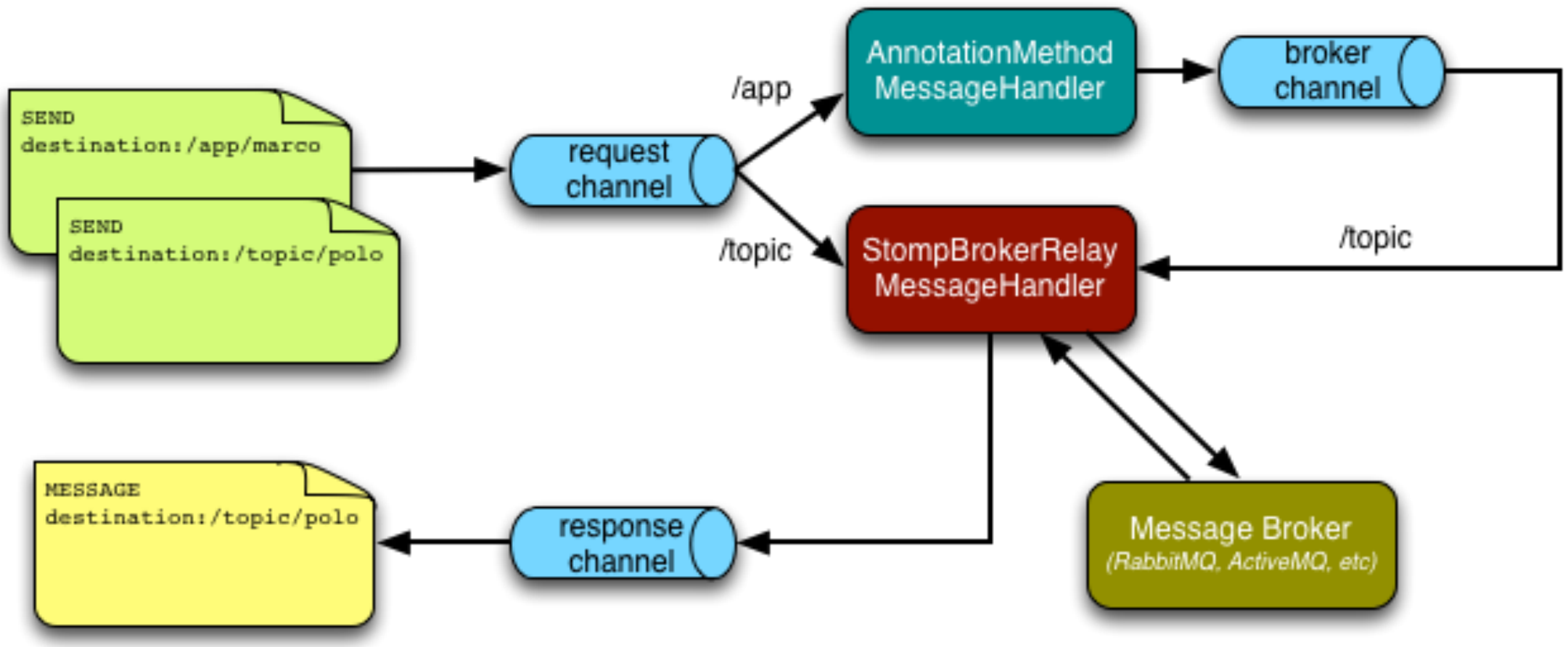
Simple broker message flow



Enabling STOMP over a broker relay

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketStompConfig extends AbstractWebSocketMessageBrokerConfigurer {

  @Override
  public void registerStompEndpoints(StompEndpointRegistry registry) {
    registry.addEndpoint("/marcopolo").withSockJS();
  }

  @Override
  public void configureMessageBroker(MessageBrokerRegistry registry) {
    registry.enableStompBrokerRelay("/queue/", "/topic/");
    registry.setApplicationDestinationPrefixes("/app");
  }

}
```

Broker relay message flow



Handling messages

```
@Controller
public class MarcoController {

  private static final Logger logger = LoggerFactory
      .getLogger(MarcoController.class);

  @MessageMapping("/marco")
  public Shout handleShout(Shout incoming) {
    logger.info("Received message: " + incoming.getMessage());

    try { Thread.sleep(2000); } catch (InterruptedException e) {}

    Shout outgoing = new Shout();
    outgoing.setMessage("Polo!");

    return outgoing;
  }

}
```

Handling subscriptions

```
@Controller
public class MarcoController {

  private static final Logger logger = LoggerFactory
      .getLogger(MarcoController.class);

  @SubscribeMapping({"/marco"})
  public Shout handleSubscription() {
    Shout outgoing = new Shout();
    outgoing.setMessage("Polo!");
    return outgoing;
  }

}
```

Handling exceptions

```
@Controller
public class MarcoController {

  private static final Logger logger = LoggerFactory
      .getLogger(MarcoController.class);

  @MessageExceptionHandler(SomeException.class)
  @SendTo("/topic/marco")
  public Shout handleException(Throwable t) {
    Shout s = new Shout();
    s.setMessage("EXCEPTION:  " + t.getMessage());
    return s;
  }

}
```

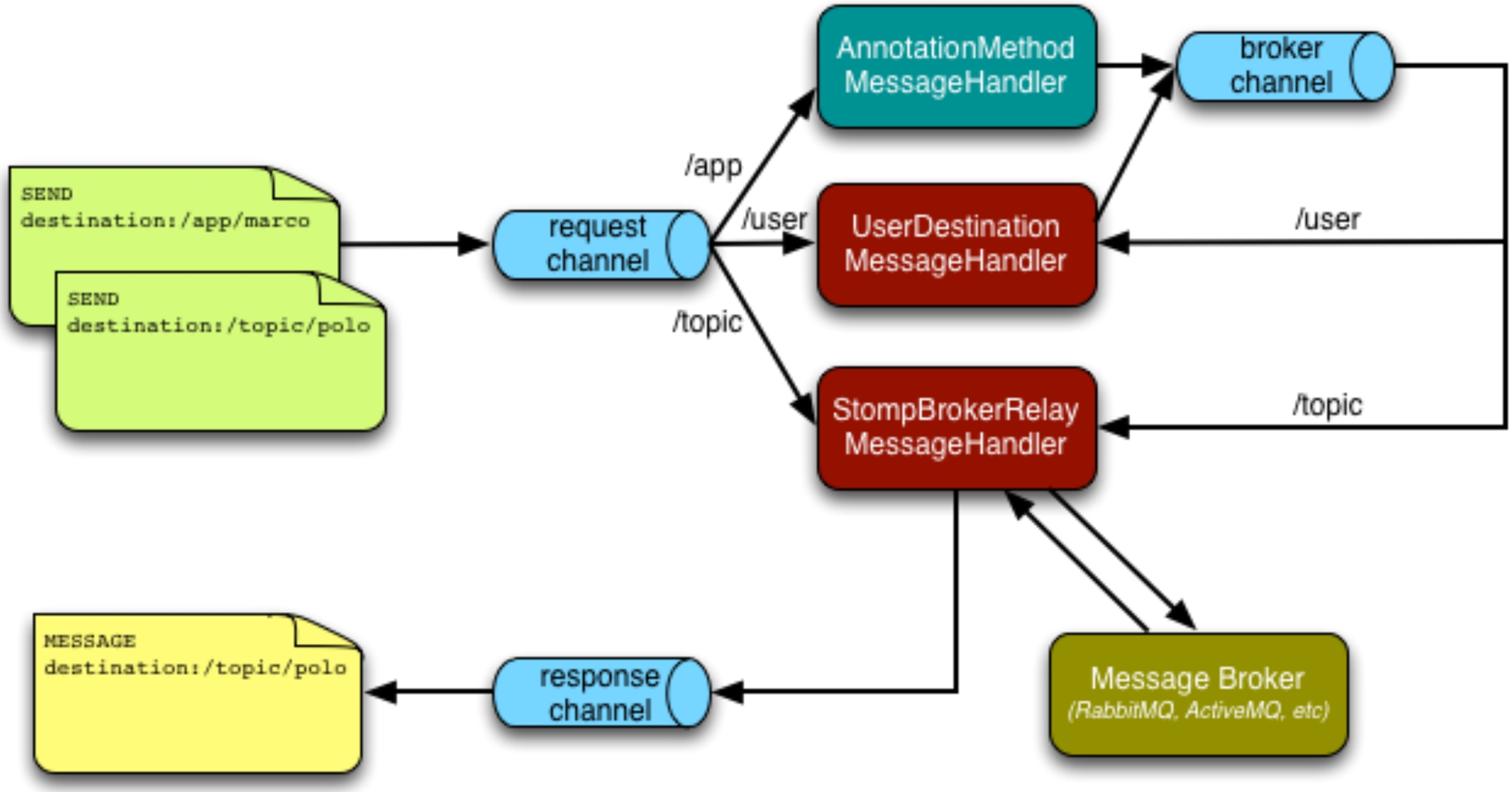
Sending messages

```
@Component
public class RandomNumberMessageSender {

  private SimpMessagingTemplate messaging;

  @Autowired
  public RandomNumberMessageSender(SimpMessagingTemplate messaging) {
    this.messaging = messaging;
  }

  @Scheduled(fixedRate=10000)
  public void sendRandomNumber() {
    Shout random = new Shout();
    random.setMessage("Random # : " + (Math.random() * 100));
    messaging.convertAndSend("/topic/marco", random);
  }

}
```

Handling user messages

```
@Controller
public class MarcoController {

  private static final Logger logger = LoggerFactory
      .getLogger(MarcoController.class);

  @MessageMapping("/marco")
  @SendToUser
  public Shout handleShout(Principal principal, Shout incoming) {
    logger.info("Received message: " + incoming.getMessage());

    try { Thread.sleep(2000); } catch (InterruptedException e) {}

    Shout outgoing = new Shout();
    outgoing.setMessage("Polo!");

    return outgoing;
  }

}
```

User message flow



Sending messages to a user

```
Notification notification = new Notification("You just got mentioned!");
messaging.convertAndSendToUser("habuma", "/queue/notifications", notification);
```

REST

Simpler REST controllers

```
@Controller
@RequestMapping("/books")
public class BookController {

  @RequestMapping(method=RequestMethod.GET)
  @ResponseBody
  public List<Book> allBooks() { ... }

  @RequestMapping(value="/{isbn}", method=RequestMethod.GET)
  @ResponseBody
  public Book bookByIsbn(@PathVariable String isbn) { ... }

  @RequestMapping(value="/search", method=RequestMethod.GET)
  @ResponseBody
  public List<Book> search(@RequestParam("q") String query) { ... }

}
```

Simpler REST controllers

```
@RestController
@RequestMapping("/books")
public class BookController {

  @RequestMapping(method=RequestMethod.GET)
  public List<Book> allBooks() { ... }

  @RequestMapping(value="/{isbn}", method=RequestMethod.GET)
  public Book bookByIsbn(@PathVariable String isbn) { ... }

  @RequestMapping(value="/search", method=RequestMethod.GET)
  public List<Book> search(@RequestParam("q") String query) { ... }

}
```

Asynchronous REST clients

```
RestTemplate rest = new RestTemplate();
ResponseEntity<String> response =
        rest.getForEntity("http://graph.facebook.com/4", String.class);
```

Asynchronous REST clients

```
AsyncRestTemplate rest = new AsyncRestTemplate();

ListenableFuture<ResponseEntity<Profile>> future =
        rest.getForEntity("http://graph.facebook.com/4", Profile.class);

future.addCallback(new ListenableFutureCallback<ResponseEntity<Profile>>() {
  public void onSuccess(ResponseEntity<Profile> response) {};
  public void onFailure(Throwable throwable) {}
});
```

Configuration

Conditional configuration

```
@Configuration
@Conditional(ThymeleafCondition.class)
public class ThymeleafConfig {
    @Bean
    public SpringTemplateEngine templateEngine(TemplateResolver templateResolver) {
        SpringTemplateEngine templateEngine = new SpringTemplateEngine();
        templateEngine.setTemplateResolver(templateResolver);
        return templateEngine;
    }

    @Bean
    public ThymeleafViewResolver viewResolver(
            SpringTemplateEngine templateEngine) {
        ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();
        viewResolver.setTemplateEngine(templateEngine);
        return viewResolver;
    }

    public TemplateResolver templateResolver() {
        TemplateResolver templateResolver = new ServletContextTemplateResolver();
        templateResolver.setPrefix("/WEB-INF/views/");
        templateResolver.setSuffix(".html");
        templateResolver.setTemplateMode("HTML5");
        return templateResolver;
    }
}
```

Conditional configuration

```
public class ThymeleafCondition implements Condition {
    public boolean matches(ConditionContext context, AnnotatedTypeMetadata metadata) {
        try {
            context.getClassLoader().loadClass(
                    "org.thymeleaf.spring4.SpringTemplateEngine");
            return true;
        } catch (ClassNotFoundException e) {
            return false;
        }
    }
}
```

Generics as autowire qualifiers

```
@Configuration
@ComponentScan
public class GenericQualifierConfig {

    @Bean
    public Thing<String> stringThing() {
        return new Thing<String>("Hello");
    }

    @Bean
    public Thing<Integer> intThing() {
        return new Thing<Integer>(42);
    }

}
```

```
@Autowired
public ThingKeeper(Thing<String> stringThing, Thing<Integer> intThing) {
    this.stringThing = stringThing;
    this.intThing = intThing;
}
```

Ordered list injection

```
@Configuration
@ComponentScan
public class ShooterConfig {

    @Bean
    @Order(1)
    public Shooter greedo() {
        return new Shooter("Greedo");
    }

    @Bean
    @Order(0)
    public Shooter han() {
        return new Shooter("Han Solo");
    }

}
```

```
@Component
public class Shooters {

    private List<Shooter> shooters;

    @Autowired
    public Shooters(List<Shooter> shooters) {
        this.shooters = shooters;
    }

    public List<Shooter> getShooters() {
        return shooters;
    }

}
```

Meta-annotations and attributes

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD, ElementType.TYPE})
@Service
@Cacheable(value="myCache", key="#id")
@Transactional(propagation=Propagation.REQUIRES_NEW,
               isolation=Isolation.REPEATABLE_READ)
public @interface SlowTransactionalService {
}
```

Meta-annotations and attributes

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD, ElementType.TYPE})
@Service
@Cacheable(value="myCache", key="#id")
@Transactional(propagation=Propagation.REQUIRES_NEW,
               isolation=Isolation.REPEATABLE_READ)
public @interface SlowTransactionalService {

   Propagation propagation();
   Isolation isolation();
   String key();

}
```

Groovy configuration

```
import com.habuma.spring4fun.Foo;
import com.habuma.spring4fun.Bar;

beans {

    bar(Bar) {
        name = "Cheers"
    }

    foo(Foo) {
        bar = bar
    }

}
```

```
GenericGroovyApplicationContext appContext =
    new GenericGroovyApplicationContext("com/habuma/spring4fun/beans.groovy");
```

Also…

@Description

```
@Component
@Description("The Thing bean")
public class Thing { … }
```

```
@Bean
@Description("The production data source")
public DataSource dataSource() { … }
```

@Lazy at injection point

```
@Autowired
@Lazy
public void setDataSource(DataSource ds) { … }
```

CGLib proxies no longer require a default constructor

Compatibility

Java 8 support

Lambdas and method references for Spring callback interfaces

java.time support

Parameter name discovery

Annotations retrofitted with @Repeatable

Compatible with Java 6 and 7

Java 8 lambda support

```
public List<Book> findAllBooks() {
    return jdbc.query(
        "select isbn, title, author from books",
        new RowMapper<Book>() {
            public Book mapRow(ResultSet rs, int rowNum) throws java.sql.SQLException {
                return new Book(
                        rs.getString("isbn"),
                        rs.getString("title"),
                        rs.getString("author"));
            };
        });
}
```

Java 8 lambda support

```
public List<Book> findAllBooks() {
    return jdbc.query(
        "select isbn, title, author from books",
        (rs, rowNum) -> {
            return new Book(
                    rs.getString("isbn"),
                    rs.getString("title"),
                    rs.getString("author"));
        });
}
```

JavaEE 6 and 7

Java EE 6 is now the baseline

JPA 2.0 / Servlet 3.0

Servlet 2.5 supported (for the sake of GAE)

Servlet 3.0+ recommended

The Spring Ecosystem

New website!

http://spring.io

Spring Boot

Auto-configuration

Dependency “starters”

Groovy CLI

Actuator

This is a **__**complete**__** Spring application

```
@RestController
class App {
  @RequestMapping("/")
  String home() {
    "Hi!"
  }
}
```

Spring Security and Java configuration

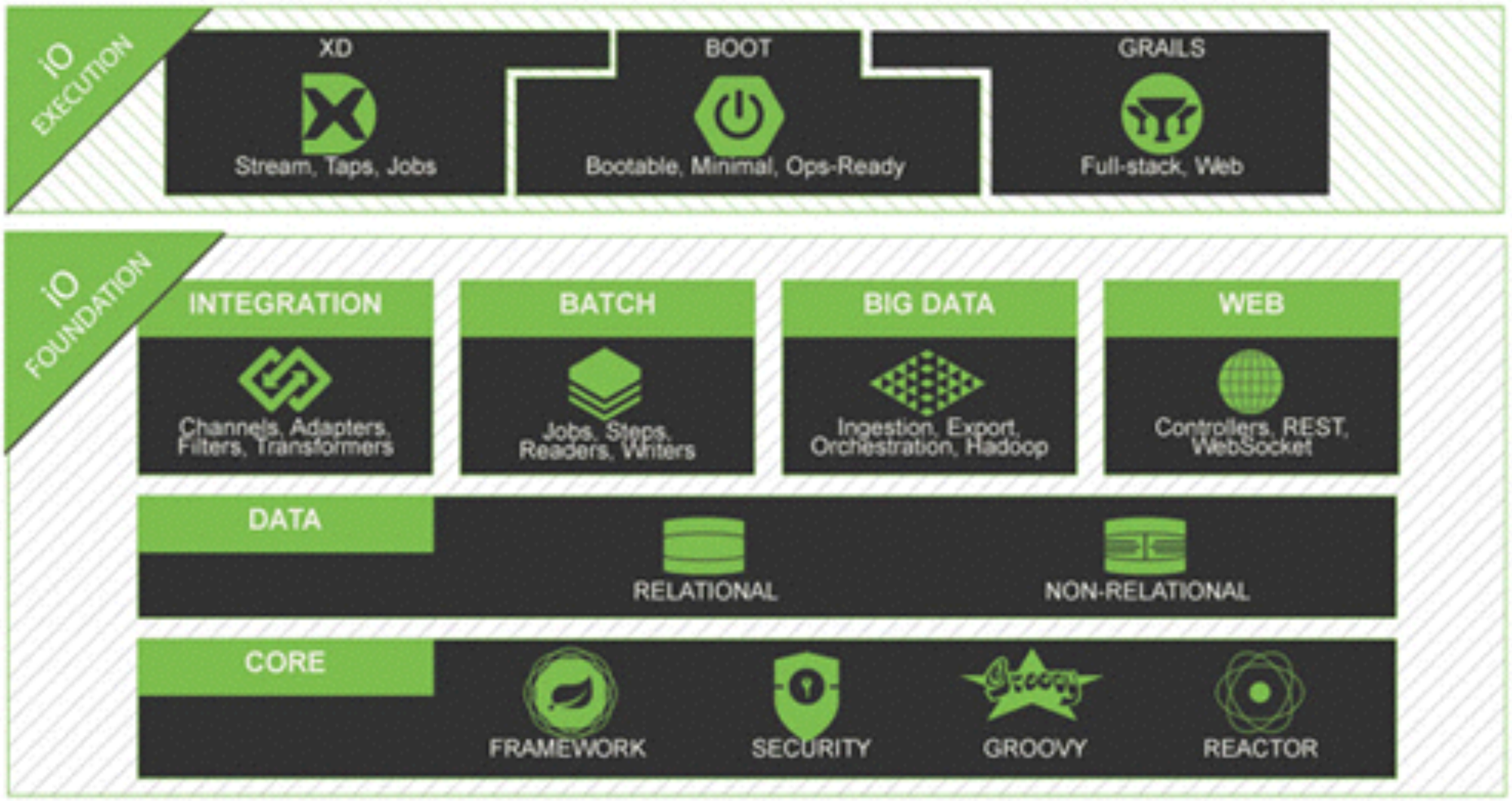
```
@Configuration
@EnableWebMvcSecurity
public class WebSecurityConfiguration extends WebSecurityConfigurerAdapter {

  @Autowired
  public void registerAuthentication(AuthenticationManagerBuilder auth) throws Exception {
    auth
      .inMemoryAuthentication()
        .withUser("habuma").password("password").roles("USER");
  }

}
```

Spring XD

```
% xd-singlenode
```

```
% xd-shell
 _____                                __   _______
/  ___|           (-)                \ \ / /  _  \
\ `--. _ __  _ __ _ _ __   __ _       \ V /| | | |
 `--. \ '_ \| '__| | '_ \ / _` |      / ^ \| | | |
/\__/ / |_) | |  | | | | | (_| |  / / \ \ |/ /
\____/| .__/|_|  |_|_| |_|\__, | \/   \/___/
      | |                   __/ |
      |_|                  |___/
eXtreme Data
1.0.0.M4 | Admin Server Target: http://localhost:8080
Welcome to the Spring XD shell. For assistance hit TAB or type "help".
xd:> stream create --name twittersearchjava --definition "twittersearch --
outputType=application/json --fixedDelay=1000 --consumerKey=afes2uqo6JAuFljdJFhqA
--consumerSecret=0top8crpmd1MXGEbbgzAwVJSAODMcbeAbhwHXLnsg --query='java' | file"
```

Spring IO



Email: craig@habuma.com Twitter: @habuma Blog: http://www.springinaction.com Sample Code: http://github.com/habuma

Thank you!