



Teaming up NoSQL & Agile NoSQL Data

Agile Data Modeling & Developer Journey with MarkLogic NoSQL Database

Niko Schmuck, Jochen Jörg

Motivation

- Proof Thesis:
 - “NoSQL increases flexibility and agility in the data layer”
 - Show case for Agile Data Management
 - Analytics Application on top of complex data with MarkLogic as NoSQL Technology
 - Developer Journey & Architecture

Application “WM14”

- Football World Cup Brazil 2014
 - Finding out insights about
 - Matches
 - Teams
 - Players
- ... based on heterogeneous sports data



WM2014 Demo Home Matches Players

Maracana matchVenue:"Rio De Janeiro" team:"Germany" 🔍

Date	Group	Team Home	Team Away	Score	Image	Snippet	Venue	Type
04/Jul/2014	Quarter-finals	France	Germany	0-1			Rio De Janeiro	Normal
13/Jul/2014	Final	Germany	Argentina	1-0			Rio De Janeiro	Extra-Time

Rio De Janeiro 2

Argentina 1

Match Date

- 2014-07-04 1
- 2014-07-13 1

Match Type

- Extra-Time 1
- Normal 1

Group

- Final 1
- Quarter-finals 1

Venue

- Rio De Janeiro 2

facet.team

- Argentina 1
- France 1
- Germany 2

Frameworks & Tools

- MarkLogic NoSQL Platform Search & Store



- Spring MVC, Spring Boot



- Thymeleaf (Templating)



- Gradle: as build tool, but also for managing MarkLogic setup/deployment

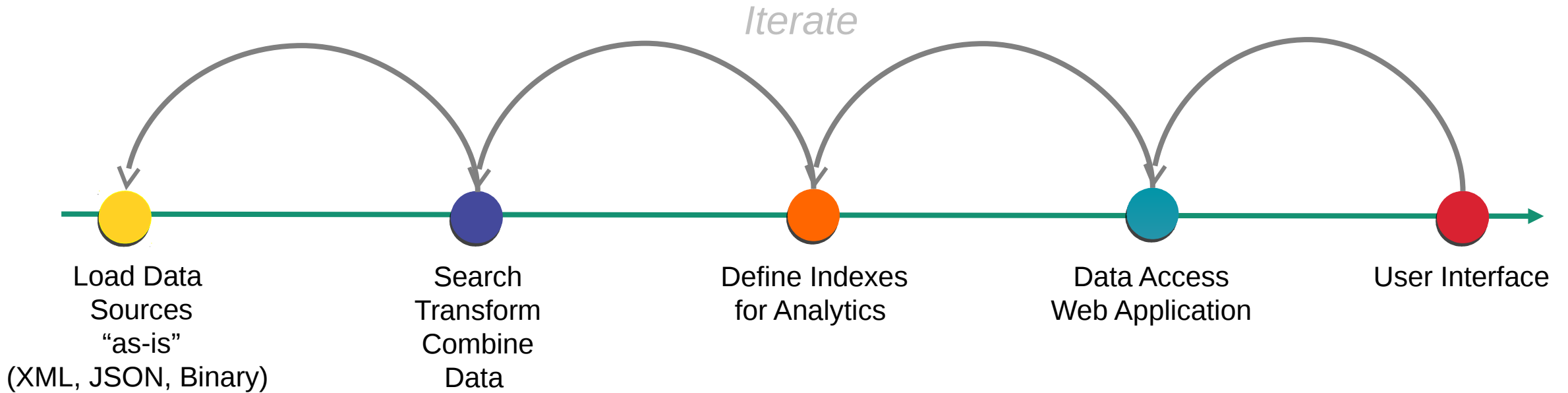


- Apache Camel:
 - Integration Framework
 - Swiss Army Knife of consuming variety of data sources



Demo I: Introduce core entities

Developer Journey == Agile Process



Domain Model in the Middle Tier?

Generic

- Exposes raw JSON to UI as stored in database
- Shifts responsibility of Validation & Resilience rgd. sparse / data variety to UI

Hybrid

- Cover dynamic attributes
- (extend from XML Document)

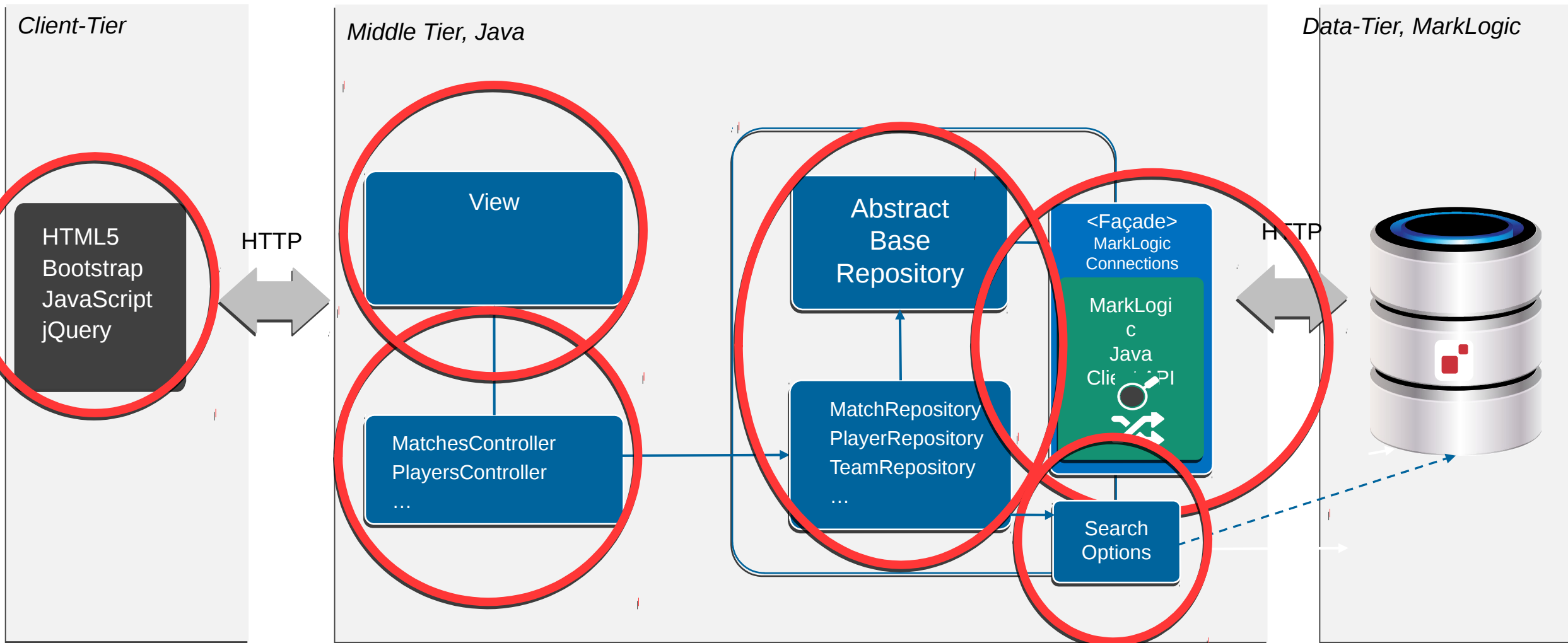
Typed

- Transform from/to JSON (semi / automatic mapping)
- Validation on ingestion
- Further access on domain objects (i.e. on templates)

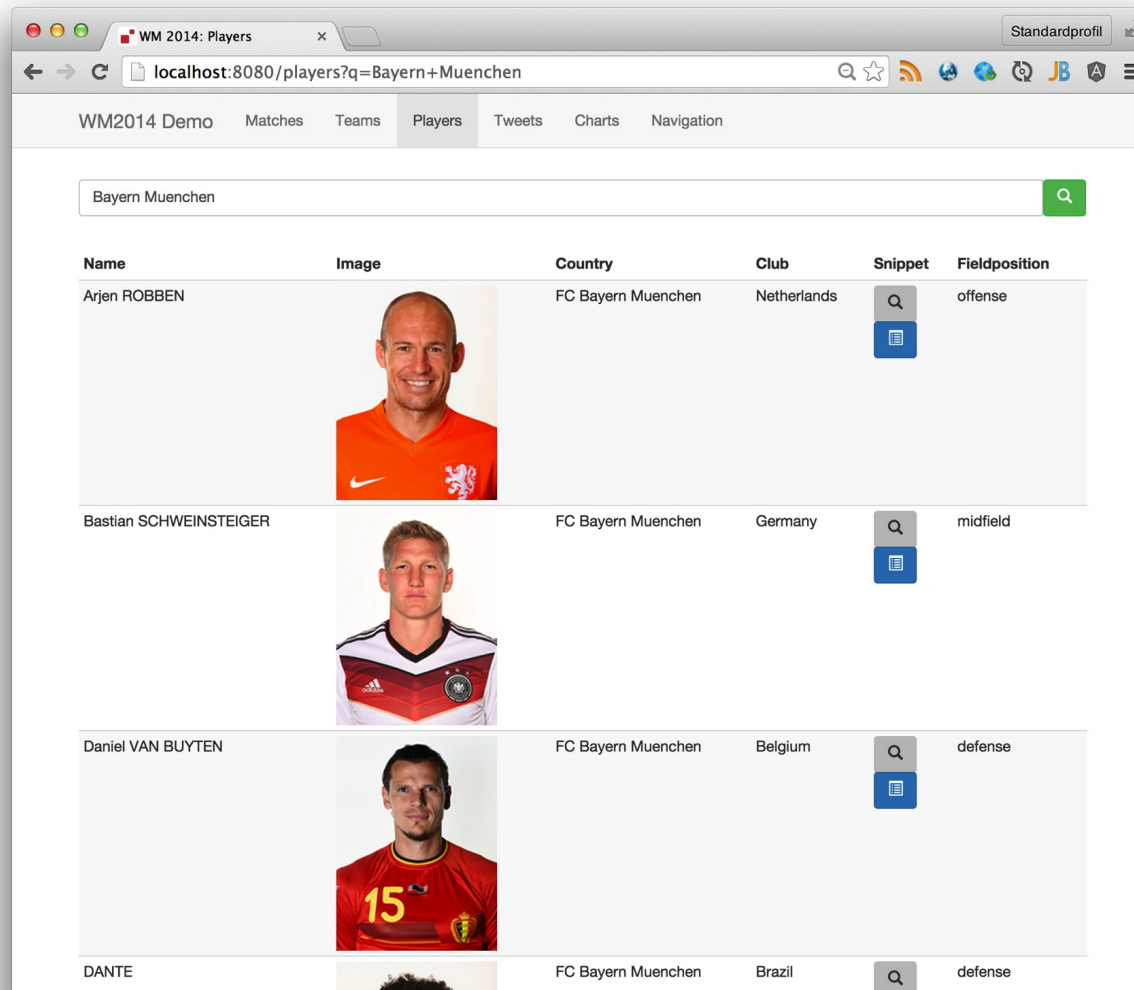


Demo II: Search scenarios












Application Architecture – Main components



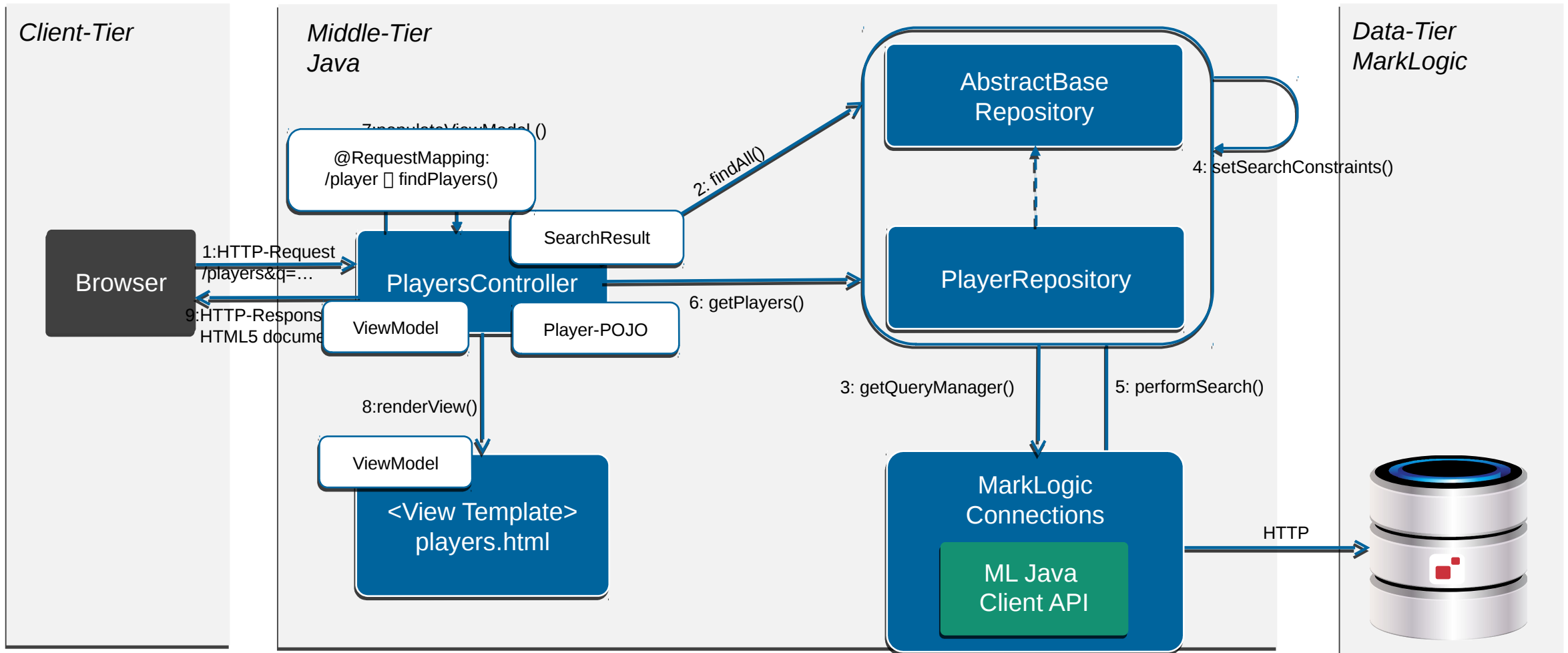
Request Lifecycle: Search for Players



The screenshot shows a web browser window with the URL `localhost:8080/players?q=Bayern+Muenchen`. The page displays a search bar with the text "Bayern Muenchen" and a search button. Below the search bar is a table of search results for players from FC Bayern Muenchen.

Name	Image	Country	Club	Snippet	Fieldposition
Arjen ROBBEN		FC Bayern Muenchen	Netherlands	 	offense
Bastian SCHWEINSTEIGER		FC Bayern Muenchen	Germany	 	midfield
Daniel VAN BUYTEN		FC Bayern Muenchen	Belgium	 	defense
DANTE		FC Bayern Muenchen	Brazil		defense

Request Lifecycle: Search for Players



Controller

```
@Controller
public class PlayersController extends AbstractBaseController {

    @Autowired
    private PlayerRepository playerRepository;

    @RequestMapping(value = "/players", method = RequestMethod.GET)
    public String findPlayers(@RequestParam(required = false, defaultValue = "") String q,
                             @RequestParam(required = false, defaultValue = "1") int page,
                             Map<String, Object> model) {

        SearchHandle searchResult = playerRepository.findAll(defaultUser, q, page);

        ...
        model.put("facets", searchResult.getFacetResults());
        // map generic search result to domain objects
        model.put("resultItems", playerRepository.getResultObjects(defaultUser, searchResult));

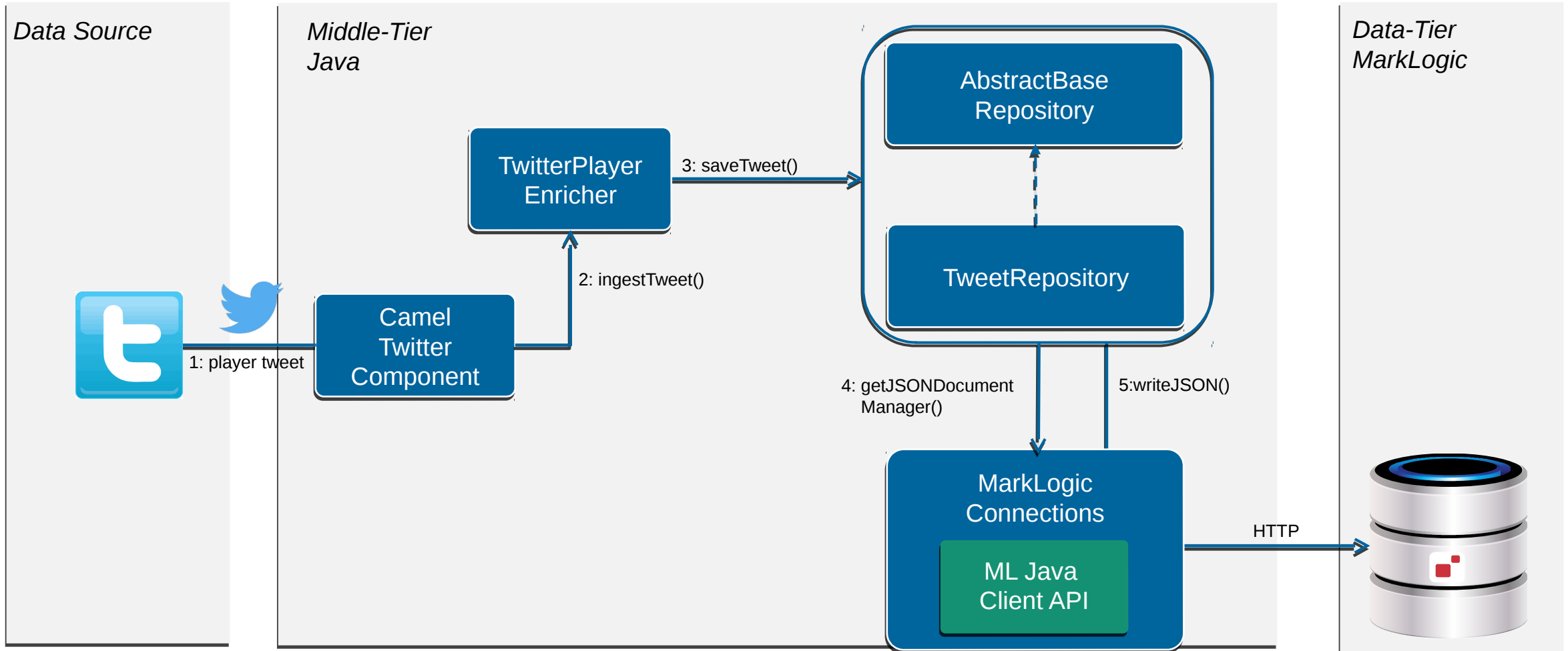
        return "search/players";
    }

    ...
}
```

```
@Service
public abstract class AbstractBaseRepository<T> {
    @Autowired
    protected MarkLogicConnections connections;

    public SearchHandle findAll(String username, String query, int page) {
        long start = (page - 1) * defaultPageLength + 1;
        QueryManager queryManager = connections.getQueryManager(username);
        StringQueryDefinition qdef = queryManager.newStringDefinition(getOptionsName());
        qdef.setCriteria(query);
        qdef.setDirectory(getDocumentDirectory());
        SearchHandle resultsHandle = new SearchHandle();
        queryManager.search(qdef, resultsHandle, start);
        return resultsHandle;
    }
}
```

Real-time Integration of Twitter Streaming API



```
@Bean
public RouteBuilder routeBuilder() {
    return new TwitterRouteBuilder();
}
private class TwitterRouteBuilder extends RouteBuilder {
    @Override
    public void configure() throws Exception {
        // setup Twitter component
        TwitterComponent tc = getContext().getComponent("twitter", TwitterComponent.class);
        tc.setAccessToken(twitter.getConfiguration().getOAuthAccessToken());
        tc.setAccessTokenSecret(twitter.getConfiguration().getOAuthAccessTokenSecret());
        tc.setConsumerKey(twitter.getConfiguration().getOAuthConsumerKey());
        tc.setConsumerSecret(twitter.getConfiguration().getOAuthConsumerSecret());

        // Follow real-time tweet stream for all players ...
        String userIds = StringUtils.arrayToDelimitedString(
            TwitterPlayerEnricher.TWITTER_USER_IDS.toArray(), ",");
        from("twitter://streaming/filter?type=event&userIds=" + userIds)
            .beanRef("twitterPlayerEnricher", "ingestTweet")
            .log("~~~~ Tweet ${body}");
    }
}
```

Wrap-up

WM14 – Role of MarkLogic

Powerful

Deliver more value, build more powerful

Semantics: RDF & SPARQL	Flexible Indexes
Triple Index	JSON/XML Storage
Full Text Search	Alerting & Event Processing
Scalable	Geospatial Query
REST & Java APIs	In-database MapReduce
Analytic Functions	Visualization Widgets

Agile

Prepare for and respond to change quickly

Schema-Agnostic	Elastic
Tiered Storage	Cloud Ready
mlcp Content Pump	Programmatic Controls & Metering
HDFS & Amazon S3 Storage	Application Builder
Hadoop Connector	SQL Support
Information Studio	BI Integration

Trusted

Enterprise-ready and secure for mission-

ACID Transactions	XA Distributed Transaction
Common Criteria Security Certification	Role-based Security & LDAP
Configuration Management	Support Monitoring & Management
Automated Failover	Backup/Restore
Point-in-time Recovery	Replication
Journal Archiving	Database Rollback

Conclusion and Take away

- Agile data modeling with NoSQL
 - Adapt to changes
 - Tackle moving targets
- Separation of concerns
 - Maintainable code base
 - Supports cross functional teams
 - Architecture evolves with your business



Join the

WM14 Demo project:

<https://github.com/jojrg/wm14>

incl. Dockerfile, sample data and instructions

Spring Boot Starter:

<https://github.com/nikos/spring-boot-starter-marklogic>



Danke!

Niko Schmuck

n.schmuck@nava.de

@niko_nava

de.linkedin.com/in/nikoschmuck

Jochen Jörg

jochen.joerg@marklogic.com

@jochenjoerg

de.linkedin.com/in/jochenjoerg