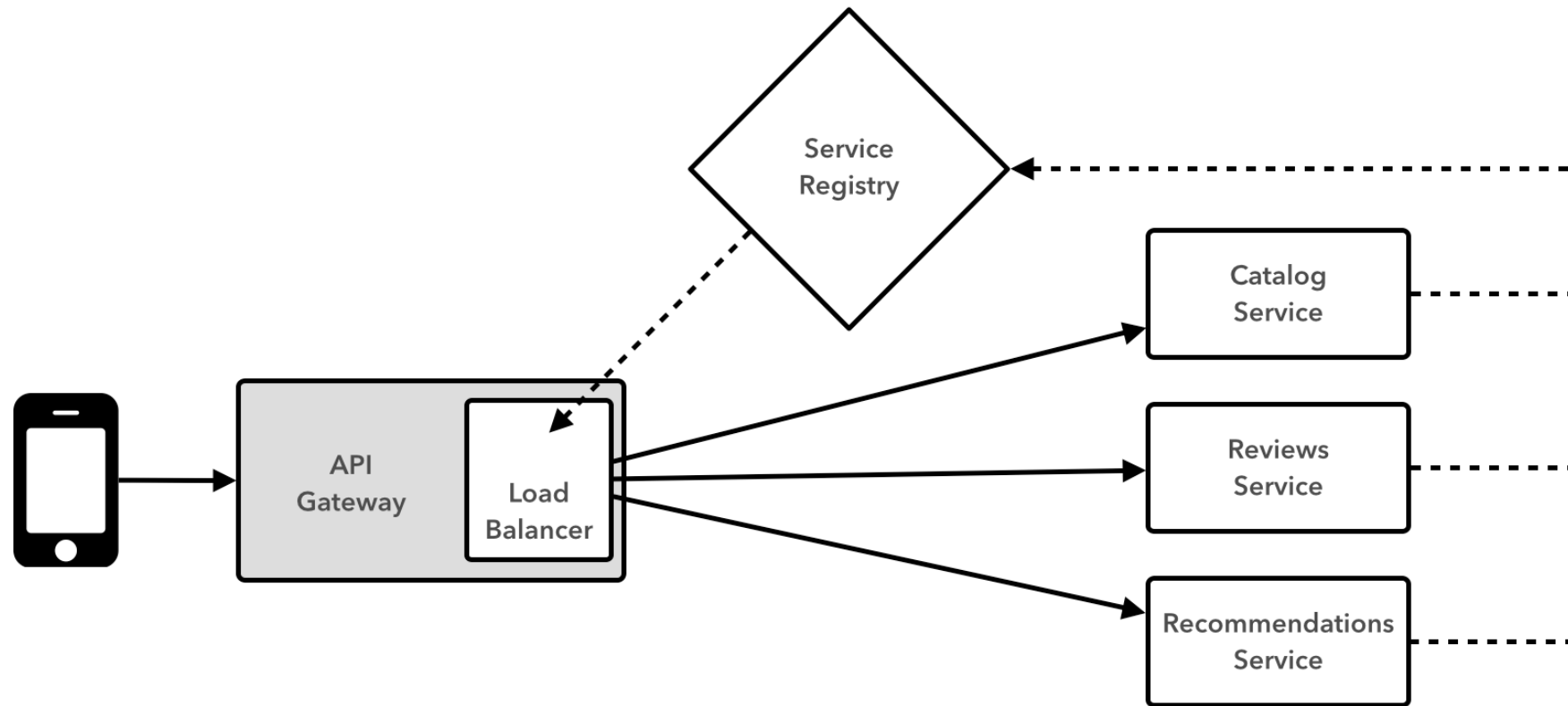


The API Gateway Pattern

Session 09

API Gateway Pattern

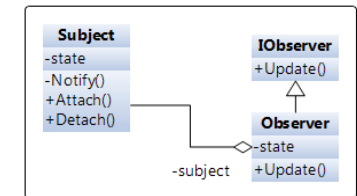
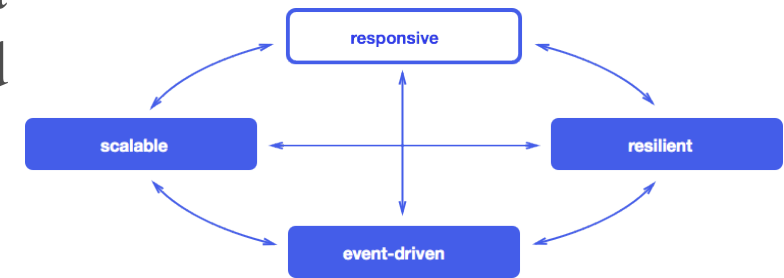


Reverse Proxy – ZUUL

```
zuul:  
  routes:  
    springbox-catalog: /catalog/**  
    springbox-reviews: /reviews/**  
    springbox-recommendations: /recommendations/**
```

Concurrent API - RxJava

- A Java VM implementation of Reactive Extensions: a library for composing asynchronous and event-based programs by using observable sequences.
 - Responsiveness - system's ability to respond in a timely manner, if at all, possible
 - Resilient - system's ability to remain responsive in the face of failure.
 - Elastic - system's ability to remain responsive under varying workload
 - Message-driven - system's reliance on asynchronous message passing between components



Aggregation & Transformation

Movie Catalog Service

```
@RequestMapping(value = "/catalog/movies/{mlId}", method = RequestMethod.GET)
public Movie movie(@PathVariable String mlId) {
    return movieRepository.findByMlId(mlId);
}
```

```
{
  id: 1001,
  title: "GoldenEye (1995)",
  mlId: "2",
  genres: [
    {
      id: 1001,
      mlId: "1",
      name: "Action"
    },
    {
      id: 1002,
      mlId: "2",
      name: "Adventure"
    },
    {
      id: 1016,
      mlId: "16",
      name: "Thriller"
    }
  ]
}
```

Movie Review Service

```
@RequestMapping(value = "/reviews/reviews/{mId}", method = RequestMethod.GET)
public Iterable<Review> reviews(@PathVariable String mId) {
    return reviewRepository.findByMId(mId);
}
```

```
[
  (
    id: "54b85cbe004e0464177e90e4",
    mId: "2",
    userName: "mstine",
    title: "GoldenEye (1995)",
    review: "Pretty good...",
    rating: 3
  ),
  (
    id: "54b85cbe004e0464177e90e5",
    mId: "2",
    userName: "starbuxman",
    title: "GoldenEye (1995)",
    review: "BOND BOND BOND!",
    rating: 5
  ),
  (
    id: "54b85cbf004e0464177e90e8",
    mId: "2",
    userName: "littleidea",
    title: "GoldenEye (1995)",
    review: "Good show!",
    rating: 4
  )
]
```

Movie Recommendation Service

```
public interface MovieRepository extends GraphRepository<Movie> {  
    Movie findByMlId(String mlId);  
  
    @Query("MATCH (movie:Movie) WHERE movie.mlId = {0} MATCH movie<-[:LIKES]-slm-[:LIKES]->recommendations " +  
        "RETURN distinct recommendations")  
    Iterable<Movie> moviesLikedByPeopleWhoLiked(String mlId);  
}
```

```
@RequestMapping(value = "/recommendations/forMovie/{mlId}", method = RequestMethod.GET)  
public Iterable<Movie> recommendedMoviesForMovie(@PathVariable String mlId) {  
    return movieRepository.moviesLikedByPeopleWhoLiked(mlId);  
}
```

```
@RequestMapping(value = "/recommendations/forMovie/{mlId}", method = RequestMethod.GET)  
public Iterable<Movie> recommendedMoviesForMovie(@PathVariable String mlId) {  
    return movieRepository.moviesLikedByPeopleWhoLiked(mlId);  
}
```


Movie Recommendation Service

```
[  
  (  
    id: 6,  
    mlId: "1",  
    title: "Toy Story (1995)"  
  ),  
  (  
    id: 1,  
    mlId: "4",  
    title: "Get Shorty (1995)"  
  ),  
  (  
    id: 2,  
    mlId: "5",  
    title: "Copycat (1995)"  
  ),  
  (  
    id: 0,  
    mlId: "3",  
    title: "Four Rooms (1995)"  
  )  
]
```

API Gateway

Catalog Integration Service

```
@Service
public class CatalogIntegrationService {

    @Autowired
    RestTemplate restTemplate;

    @HystrixCommand(fallbackMethod = "stubMovie")
    public Observable<Movie> getMovie(final String mlId) {
        return new ObservableResult<Movie>() {
            @Override
            public Movie invoke() {
                return restTemplate.getForObject("http://catalog-service/catalog/movies/{mlId}", Movie.class, mlId);
            }
        };
    }

    private Movie stubMovie(final String mlId) {
        Movie stub = new Movie();
        stub.setMlId(mlId);
        stub.setTitle("Interesting...the wrong title. Sssshhhh!");
        return stub;
    }
}
```

Review Integration Service

```
@Service
public class ReviewsIntegrationService {

    @Autowired
    RestTemplate restTemplate;

    @HystrixCommand(fallbackMethod = "stubReviews")
    public Observable<List<Review>> reviewsFor(String mlId) {
        return new ObservableResult<List<Review>>() {
            @Override
            public List<Review> invoke() {
                ParameterizedTypeReference<List<Review>> responseType = new ParameterizedTypeReference<List<Review>>() {
                };
                return restTemplate.exchange("http://reviews-service/reviews/reviews/{mlId}", HttpMethod.GET, null, responseType, mlId).getBody();
            }
        };
    }

    private List<Review> stubReviews(String mlId) {
        Review review = new Review();
        review.setMlId(mlId);
        review.setRating(4);
        review.setTitle("Interesting...the wrong title. Sssshhhh!");
        review.setReview("Awesome sauce!");
        review.setUserName("joeblow");
        return Arrays.asList(review);
    }
}
```

Recommendations Integration Service

```
@Service
public class RecommendationsIntegrationService {
    @Autowired
    RestTemplate restTemplate;

    @MyStryxCommand(fallbackMethod = "stubRecommendations")
    public Observable<List<Movie>> getRecommendations(final String mlId) {
        return new ObservableResult<List<Movie>>() {
            @Override
            public List<Movie> invoke() {
                ParameterizedTypeReference<List<Movie>> responseType = new ParameterizedTypeReference<List<Movie>>() {
                };
                return restTemplate.exchange("http://recommendations-service/recommendations/forMovie/{mlId}", HttpMethod.GET, null, responseType, mlId).getBody();
            }
        };
    }

    private List<Movie> stubRecommendations(final String mlId) {
        Movie one = new Movie();
        one.setMlId("25");
        one.setMlId("A movie which doesn't exist");
        Movie two = new Movie();
        two.setMlId("28");
        two.setMlId("A movie about nothing");
        return Arrays.asList(one, two);
    }
}
```

Concurrently Aggregate and Transform

```
@RequestMapping("/movie/{mlId}")
public DeferredResult<MovieDetails> movieDetails(@PathVariable String mlId) {
    Observable<MovieDetails> details = Observable.zip(

        catalogIntegrationService.getMovie(mlId),
        reviewsIntegrationService.reviewsFor(mlId),
        recommendationsIntegrationService.getRecommendations(mlId),

        (movie, reviews, recommendations) -> {
            MovieDetails movieDetails = new MovieDetails();
            movieDetails.setMlId(movie.getMlId());
            movieDetails.setTitle(movie.getTitle());
            movieDetails.setReviews(reviews);
            movieDetails.setRecommendations(recommendations);
            return movieDetails;
        }

    );
    return toDeferredResult(details);
}
```

Labs!!