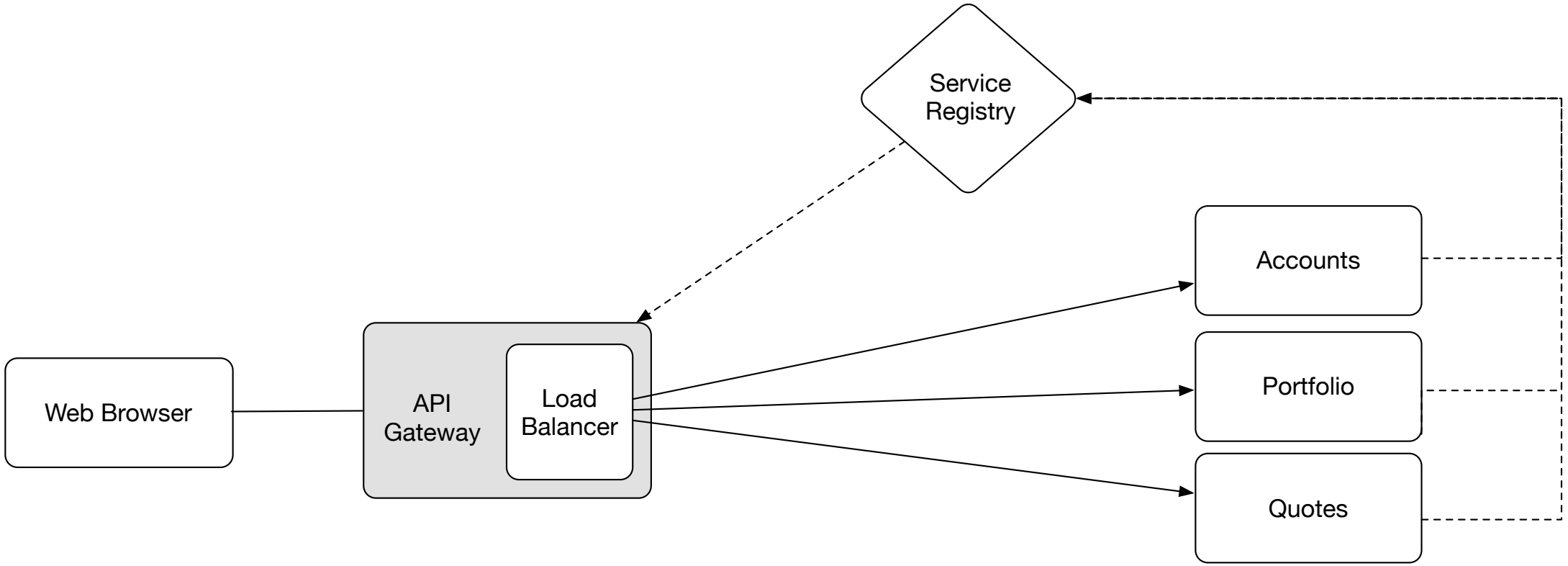
The API Gateway Pattern

Session 09

API Gateway Pattern



© 2015 Pivotal Software, Inc. All rights reserved.

Aggregation & Transformation

© 2015 Pivotal Software, Inc. All rights reserved.

Quotes Service

```
@RequestMapping(value = "/quote/{symbol}", method = RequestMethod.GET)
public ResponseEntity<Stock> getQuote(@PathVariable("symbol") final String symbol) throws Exception {
    logger.debug("QuoteController.getQuote: retrieving quote for: " + symbol);
    Stock stock = quoteService.getQuote(symbol);
    logger.info(String.format("Retrieved symbol: %s with quote %s", symbol, stock));
    return new ResponseEntity<>(stock, getNoCacheHeaders(), HttpStatus.OK);
}

/**
 * Searches for companies that have a name or symbol matching the parameter.
 *
 * @param name The name or symbol to search for.
 * @return The list of companies that match the search parameter.
 */
@RequestMapping(value = "/company/{name}", method = RequestMethod.GET)
public ResponseEntity<List<Stock>> getCompanies(@PathVariable("name") final String name) {
    logger.debug("QuoteController.getCompanies: retrieving companies for: " + name);
    List<Stock> companies = quoteService.companiesByNameOrSymbol(name);
    logger.info(String.format("Retrieved companies with search parameter: %s - list: {}", name), companies);
    return new ResponseEntity<>(companies, HttpStatus.OK);
}
```

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

Accounts Service

```
/**...*/
@RequestMapping(value = "/account/{id}", method = RequestMethod.GET)
public ResponseEntity<Account> find(@PathVariable("id") final int id) {

    logger.info("AccountController.find: id=" + id);

    Account accountResponse = accountService.getAccount(id);
    return new ResponseEntity<>(accountResponse,getNoCacheHeaders(), HttpStatus.OK);
}

@RequestMapping(value = "/account/", method = RequestMethod.GET)
public ResponseEntity<Account> findAccount(@RequestParam(value="name") final String id) {

    logger.info("AccountController.getAccount: id=" + id);

    Account accountResponse = accountService.getAccount(id);
    return new ResponseEntity<>(accountResponse,getNoCacheHeaders(), HttpStatus.OK);

}

/**...*/
@RequestMapping(value = "/account", method = RequestMethod.POST)
public ResponseEntity<String> save(@RequestBody Account accountRequest, UriComponentsBuilder builder) {

    logger.debug("AccountController.save: userId=" + accountRequest.getUserid());

    Integer accountProfileId = accountService.saveAccount(accountRequest);
    HttpHeaders responseHeaders = new HttpHeaders();
    responseHeaders.setLocation(builder.path("/account/{id}").buildAndExpand(accountProfileId).toUri());

    return new ResponseEntity<>(responseHeaders, HttpStatus.CREATED);
}
/**...*/
@RequestMapping(value = "/accounts/{userId}/decreaseBalance/{amount:.+}", method = RequestMethod.GET)
public ResponseEntity<Double> decreaseBalance(@PathVariable("userId") final String userId, @PathVariable("amount") final double amount) {

    logger.debug("AccountController.decreaseBalance: id='" + userId + "', amount='"+amount+"'");

    if (amount > 0.0) {

        double currentBalance = accountService.getAccount(userId).getBalance().doubleValue();
        double newBalance = accountService.decreaseBalance(amount,userId);

        if(currentBalance != newBalance) return new ResponseEntity<>(newBalance,getNoCacheHeaders(), HttpStatus.OK);

        else return new ResponseEntity<>(currentBalance,getNoCacheHeaders(), HttpStatus.EXPECTATION_FAILED);

    } else {

        return new ResponseEntity<>(getNoCacheHeaders(), HttpStatus.EXPECTATION_FAILED);
    }

}
/**...*/
@RequestMapping(value = "/accounts/{userId}/increaseBalance/{amount:.+}", method = RequestMethod.GET)
public ResponseEntity<Double> increaseBalance(@PathVariable("userId") final String userId, @PathVariable("amount") final double amount) {

    logger.debug("AccountController.increaseBalance: id='" + userId + "', amount='"+amount+"'");

    if (amount > 0.0) {

        double currentBalance = accountService.increaseBalance(amount, userId);
        logger.debug("AccountController.increaseBalance: current balance='" + currentBalance + "'.");
        return new ResponseEntity<>(currentBalance, getNoCacheHeaders(),HttpStatus.OK);

    } else {

        //amount can not be negative for increaseBalance, please use decreaseBalance
        return new ResponseEntity<>(accountService.getAccount(userId).getBalance().doubleValue(), getNoCacheHeaders(),
                HttpStatus.EXPECTATION_FAILED);
    }

}
```

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

Portfolio Service

```
/**...*/
@RequestMapping(value = "/portfolio/{id}", method = RequestMethod.GET)
public ResponseEntity<Portfolio> getPortfolio(@PathVariable("id") final String accountId) {
    logger.debug("PortfolioController: Retrieving portfolio with user id:" + accountId);
    Portfolio folio = service.getPortfolio(accountId);
    logger.debug("PortfolioController: Retrieved portfolio:" + folio);
    return new ResponseEntity<>(folio, getNoCacheHeaders(), HttpStatus.OK);
}

/**...*/
@RequestMapping(value = "/portfolio/{id}", method = RequestMethod.POST)
public ResponseEntity<Order> addOrder(@PathVariable("id") final String accountId, @RequestBody final Order order, UriComponentsBuilder builder) {
    logger.debug("Adding Order: " + order);

    //TODO: can do a test to ensure accountId == order.getAccountId();

    Order savedOrder = service.addOrder(order);
    HttpHeaders responseHeaders = new HttpHeaders();
    responseHeaders.setLocation(builder.path("/portfolio/{id}")
            .buildAndExpand(accountId).toUri());
    logger.debug("Order added: " + savedOrder);
    if (savedOrder != null && savedOrder.getOrderId() != null) {
        return new ResponseEntity<Order>(savedOrder, responseHeaders, HttpStatus.CREATED);
    } else {
        logger.warn("Order not saved: " + order);
        return new ResponseEntity<Order>(savedOrder, responseHeaders, HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

API Gateway

© 2015 Pivotal Software, Inc. All rights reserved.

Quotes Integration Service

```java
@HystrixCommand(fallbackMethod = "getQuoteFallback")
public Quote getQuote(String symbol) {
    logger.debug("Fetching quote: " + symbol);
    Quote quote = restTemplate.getForObject("http://quotes/quote/{symbol}", Quote.class, symbol);
    return quote;
}

private Quote getQuoteFallback(String symbol) {
    logger.debug("Fetching fallback quote for: " + symbol);
    Quote quote = new Quote();
    quote.setSymbol(symbol);
    quote.setStatus("FAILED");
    return quote;
}
@HystrixCommand(fallbackMethod = "getCompaniesFallback")
public List<CompanyInfo> getCompanies(String name) {
    logger.debug("Fetching companies with name or symbol matching: " + name);
    CompanyInfo[] infos = restTemplate.getForObject("http://quotes/company/{name}", CompanyInfo[].class, name);
    return Arrays.asList(infos);
}


private List<CompanyInfo> getCompaniesFallback(String name) {
    List<CompanyInfo> infos = new ArrayList<>();
    return infos;
}


//TODO: prime location for a redis/gemfire caching service!
@Scheduled(fixedRate = REFRESH_PERIOD)
protected void retrieveMarketSummary() {
    logger.debug("Scheduled retrieval of Market Summary");
    List<Quote> quotesIT = pickRandomThree(symbolsIT).stream().map(symbol -> getQuote(symbol)).collect(Collectors.toList());
    List<Quote> quotesFS = pickRandomThree(symbolsFS).stream().map(symbol -> getQuote(symbol)).collect(Collectors.toList());
    summary.setTopGainers(quotesIT);
    summary.setTopLosers(quotesFS);
}
```

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

Accounts Integration Service

```java
@Service
public class AccountsIntegrationService {
    private static final Logger logger = LoggerFactory
            .getLogger(AccountsIntegrationService.class);

    @Autowired
    @LoadBalanced
    private RestTemplate restTemplate;

    public void createAccount(Account account) {
        logger.debug("Saving account with userId: " + account.getUserid());
        String status = restTemplate.postForObject("http://accounts/account/", account, String.class);
        logger.info("Status from registering account for "+ account.getUserid()+ " is " + status);
    }

    public Map<String,Object> login(AuthenticationRequest request){
        logger.debug("logging in with userId:" + request.getUsername());
        Map<String,Object> result = (Map<String, Object>) restTemplate.postForObject("http://accounts/login/", request, Map.class);
        return result;
    }

    //TODO: change to /account/{user}
    public Account getAccount(String user) {
        logger.debug("Looking for account with userId: " + user);

        Account account = restTemplate.getForObject("http://accounts/account/?name={user}", Account.class, user);
        logger.debug("Got Account: " + account);
        return account;
    }

    public void logout(String user) {
        logger.debug("logging out account with userId: " + user);

        ResponseEntity<?> response = restTemplate.getForEntity("http://accounts/logout/{user}", String.class, user);
        logger.debug("Logout response: " + response.getStatusCode());
    }

}
```

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

Portfolio Integration Service

```java
@Service
public class PortfolioIntegrationService {

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


    @Autowired
    @LoadBalanced
    private RestTemplate restTemplate;


    public Order sendOrder(Order order ) throws OrderNotSavedException {
        logger.debug("send order: " + order);

        //check result of http request to ensure its ok.

        ResponseEntity<Order> result = restTemplate.postForEntity("http://portfolio/portfolio/{accountId}", order, Order.class, order.getAccountId());
        if (result.getStatusCode() == HttpStatus.INTERNAL_SERVER_ERROR) {
            throw new OrderNotSavedException("Could not save the order");
        }
        logger.debug("Order saved:: " + result.getBody());
        return result.getBody();
    }

    public Portfolio getPortfolio(String accountId) {
        Portfolio folio = restTemplate.getForObject("http://portfolio/portfolio/{accountid}", Portfolio.class, accountId);
        logger.debug("Portfolio received: " + folio);
        return folio;
    }
}
```

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

Aggregate and Transform

```
@RequestMapping(value = "/portfolio", method = RequestMethod.GET)
public String portfolio(Model model) {
    logger.debug("/portfolio");
    model.addAttribute("marketSummary", quotesIntegrationService.getMarketSummary());

    //check if user is logged in!
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    if (!(authentication instanceof AnonymousAuthenticationToken)) {
        String currentUserName = authentication.getName();
        logger.debug("portfolio: User logged in: " + currentUserName);

        //TODO: add account summary.
        try {
            model.addAttribute("portfolio", portfolioIntegrationService.getPortfolio(currentUserName));
        } catch (HttpServerErrorException e) {
            logger.debug("error retrieving portfolfio: " + e.getMessage());
            model.addAttribute("portfolioRetrievalError",e.getMessage());
        }
        model.addAttribute("order", new Order());
    }

    return "portfolio";
}
```

© 2015 Pivotal Software, Inc. All rights reserved.

Reverse Proxy – ZUUL

- avoids the need to manage CORS and authentication concerns independently for all the backends

```
zuul:
  routes:
    quotes: /quotes/**
    accounts: /accounts/**
    portfoio: /portfolio/**
```

© 2015 Pivotal Software, Inc. All rights reserved.