Orianna Documentation

Release 4.0.0-SNAPSHOT

Meraki Analytics

Aug 06, 2023

Contents

CHAPTER 1

What is Orianna?

Orianna is a Java framework for working with data from the Riot API. It's focused on usability and convenience, taking care of the all the little details that come with using the Riot API for you, so you can focus on building your application. Here's a quick overview of some of the main features that set Orianna apart from other Riot API wrappers:

- **An enhanced user interface that makes using the Riot API easy and fun**
  - Restructured and renamed API data for maximum clarity
  - Fluent APIs for requesting data
  - Automatic conversion of foreign keys into the objects they specify (e.g player.getChampion() instead of player.getChampionId())
- Rate limits handled automatically with optimal usage of your API key
- Configurable handling of API errors (e.g. automatic retry on 500 errors)
- Built-in automatic caching, ready out of the box
- DataDragon support
- **A highly configurable pipeline for requesting and caching Riot API data**
  - Allows flexibility in what database(s) you want to use to cache your data
  - Off-the-shelf support for many popular databases under construction here
  - Extensible to seamlessly integrate other data types and APIs, such as the ChampionGG API (ChampionGG support coming soon!)

Orianna is the sister library to Cassiopeia (Python).

Orianna is distributed under the MIT License.

CHAPTER 2

Documentation Overview

2.1 How To Get Orianna

Orianna is distributed through the GitHub release page and through Maven Central. The easiest way to get it is by using Maven or Gradle.

2.1.1 Maven

To add the latest Orianna release version to your maven project, add the dependency to your `pom.xml` dependencies section:

```
<dependencies>
  <dependency>
    <groupId>com.merakianalytics.orianna</groupId>
    <artifactId>orianna</artifactId>
    <version>4.0.0-rc9</version>
    <!-- or, for Android: -->
    <artifactId>orianna-android</artifactId>
    <version>4.0.0-rc9</version>
  </dependency>
</dependencies>
```

Or, if you want you get the latest development version, add the SonaType Snapshot Repository to your `pom.xml` as well:

```
<dependencies>
  <dependency>
    <groupId>com.merakianalytics.orianna</groupId>
    <artifactId>orianna</artifactId>
    <version>4.0.0-SNAPSHOT</version>
    <!-- or, for Android: -->
    <artifactId>orianna-android</artifactId>
```

(continues on next page)

(continued from previous page)

```
    <version>4.0.0-SNAPSHOT</version>
  </dependency>
</dependencies>

<repositories>
  <repository>
    <id>snapshots-repo</id>
    <url>https://oss.sonatype.org/content/repositories/snapshots</url>
    <releases>
      <enabled>false</enabled>
    </releases>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
  </repository>
</repositories>
```

2.1.2 Gradle

To add the latest Orianna release version to your gradle project, add the Maven Central repository to your build.gradle repositories section, and add the dependency to your `build.gradle` dependencies section:

```
repositories {
  mavenCentral()
}

dependencies {
  compile "com.merakianalytics.orianna:orianna:4.0.0-rc9"
  // or, for Android:
  compile "com.merakianalytics.orianna:orianna-android:4.0.0-rc9"
}
```

Or, if you want to get the latest development version, add the Sonaype Snapshot Repository to your `build.gradle` instead:

```
repositories {
  maven { url "https://oss.sonatype.org/content/repositories/snapshots" }
}

dependencies {
  compile "com.merakianalytics.orianna:orianna:4.0.0-SNAPSHOT"
  // or, for Android:
  compile "com.merakianalytics.orianna:orianna-android:4.0.0-SNAPSHOT"
}
```

2.1.3 Using the release JAR directly

Grab the latest JAR from the releases page and add it to your project dependencies. JARs are provide both with and without Orianna's dependencies included. The `jar-with-dependencies` version will get you up & running faster, but can cause version conflicts if your project has other dependencies.

If you're using the JAR without dependencies inlcuded, Orianna depends on the following libraries which will also need to be added as dependencies:

- slf4j-api (version 1.7.25)
- datapipelines (version 1.0.4)
- hipster4j (version 1.0.1)
- guava (version 20.0)
- okhttp (version 3.13.1)
- jackson-databind (version 2.10.0.pr1)
- jackson-dataformat-msgpack (version 0.8.16)
- joda-time (version 2.10.1)
- jackson-datatype-joda (version 2.10.0.pr1)
- cache2k (version 1.2.3.Final)

2.2 Using Orianna

Orianna is a feature-rich framework for working with Riot API data. When used out-of-the-box, Orianna pulls data from the Riot API (Static Data comes from DataDragon) and caches it in-memory behind the scenes. Just ask for the data you want and caching is handled automatically. You can also configure Orianna to use a different cache, database, or combination of caches and databases. It can also be configured to pull data from sources other than the Riot API or DataDragon. See *the configuration documentation* for more information on how to do this.

2.2.1 Orianna APIs

There are two top-level APIs for accessing data with Orianna. Both use Fluent APIs to define the parameters for API requests. The two APIs get the same data, but they look a little different. Here's an example:

- Data API: `Summoner.named("FatalElement").withRegion(Region.NORTH_AMERICA).get()`
- Orianna API: `Orianna.summonerNamed("FatalElement").withRegion(Region.NORTH_AMERICA).get()`

Using the Data Class API

Champion Mastery API

```
import java.util.List;

import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.championmastery.ChampionMasteries;
import com.merakianalytics.orianna.types.core.championmastery.ChampionMastery;
import com.merakianalytics.orianna.types.core.championmastery.ChampionMasteryScore;
import com.merakianalytics.orianna.types.core.championmastery.ChampionMasteryScores;
import com.merakianalytics.orianna.types.core.staticdata.Champion;
import com.merakianalytics.orianna.types.core.staticdata.Champions;
import com.merakianalytics.orianna.types.core.summoner.Summoner;
import com.merakianalytics.orianna.types.core.summoner.Summoners;

public class Example {
```

(continues on next page)

(continued from previous page)

```
    public static void main(String[] args) {
        List<Summoner> summoners = Summoners.named("FatalElement", "Kalturi").
↪withRegion(Region.NORTH_AMERICA).get();
        List<Champion> champions = Champions.named("Annie", "Thresh").
↪withRegion(Region.NORTH_AMERICA).get();

        ChampionMasteries masteries = ChampionMasteries.forSummoner(summoners.get(0)).
↪get();
        ChampionMastery mastery = ChampionMastery.forSummoner(summoners.get(0)).
↪withChampion(champions.get(0)).get();
        ChampionMasteryScore score = ChampionMasteryScore.forSummoner(summoners.
↪get(0)).get();

        List<ChampionMasteries> manyMasteries = ChampionMasteries.
↪forSummoners(summoners).get();
        List<ChampionMastery> manyMastery = ChampionMasteries.forSummoner(summoners.
↪get(0)).withChampions(champions).get();
        List<ChampionMasteryScore> manyScores = ChampionMasteryScores.
↪forSummoners(summoners).get();
    }
}
```

Champion Status API

Champion Status information is available directly on the Champion within Orianna.

```
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.staticdata.Champions;

public class Example {
    public static void main(String[] args) {
        Champions champions = Champions.withRegion(Region.NORTH_AMERICA).get();

        boolean firstFree = champions.get(0).isFreeToPlay();
        boolean lastEnabled = champions.get(champions.size() - 1).isEnabled();
    }
}
```

League API

```
import java.util.List;
import java.util.stream.Collectors;

import com.merakianalytics.orianna.types.common.Queue;
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.league.League;
import com.merakianalytics.orianna.types.core.league.LeaguePosition;
import com.merakianalytics.orianna.types.core.league.LeaguePositions;
import com.merakianalytics.orianna.types.core.league.Leagues;
import com.merakianalytics.orianna.types.core.summoner.Summoner;
import com.merakianalytics.orianna.types.core.summoner.Summoners;

public class Example {
```

(continues on next page)

(continued from previous page)

```
    public static void main(String[] args) {
        League challenger = League.challengerInQueue(Queue.RANKED_SOLO_5x5).
↪withRegion(Region.NORTH_AMERICA).get();
        League master = League.masterInQueue(Queue.RANKED_SOLO_5x5).withRegion(Region.
↪NORTH_AMERICA).get();

        List<League> allChallenger = Leagues.challengerInQueues(Queue.RANKED).
↪withRegion(Region.NORTH_AMERICA).get();
        List<League> allMaster = Leagues.masterInQueues(Queue.RANKED).
↪withRegion(Region.NORTH_AMERICA).get();

        List<Summoner> summoners = Summoners.named("FatalElement", "Kalturi").
↪withRegion(Region.NORTH_AMERICA).get();

        LeaguePositions positions = LeaguePositions.forSummoner(summoners.get(1)).
↪get();
        List<LeaguePositions> manyPositions = LeaguePositions.forSummoners(summoners).
↪get();

        League league = positions.get(0).getLeague();
        List<League> manyLeagues = positions.stream().map(LeaguePosition::getLeague).
↪collect(Collectors.toList());

        String leagueId = league.getId();
        List<String> manyLeagueIds = manyLeagues.stream().map(League::getId).
↪collect(Collectors.toList());

        league = League.withId(leagueId).withRegion(Region.NORTH_AMERICA).get();
        manyLeagues = Leagues.withIds(manyLeagueIds).withRegion(Region.NORTH_AMERICA).
↪get();
    }
}
```

Static Data API

```
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.staticdata.Champion;
import com.merakianalytics.orianna.types.core.staticdata.Champions;
import com.merakianalytics.orianna.types.core.staticdata.Item;
import com.merakianalytics.orianna.types.core.staticdata.Items;
import com.merakianalytics.orianna.types.core.staticdata.LanguageStrings;
import com.merakianalytics.orianna.types.core.staticdata.Languages;
import com.merakianalytics.orianna.types.core.staticdata.Map;
import com.merakianalytics.orianna.types.core.staticdata.Maps;
import com.merakianalytics.orianna.types.core.staticdata.Masteries;
import com.merakianalytics.orianna.types.core.staticdata.Mastery;
import com.merakianalytics.orianna.types.core.staticdata.ProfileIcon;
import com.merakianalytics.orianna.types.core.staticdata.ProfileIcons;
import com.merakianalytics.orianna.types.core.staticdata.Realm;
import com.merakianalytics.orianna.types.core.staticdata.ReforgedRune;
import com.merakianalytics.orianna.types.core.staticdata.ReforgedRunes;
import com.merakianalytics.orianna.types.core.staticdata.Rune;
import com.merakianalytics.orianna.types.core.staticdata.Runes;
import com.merakianalytics.orianna.types.core.staticdata.SummonerSpell;
```

(continues on next page)

(continued from previous page)

```
import com.merakianalytics.orianna.types.core.staticdata.SummonerSpells;
import com.merakianalytics.orianna.types.core.staticdata.Versions;

public class Example {
    public static void main(String[] args) {
        // Champions
        Champions champions = Champions.withRegion(Region.NORTH_AMERICA).get();

        Champion champion = Champion.named("Annie").withRegion(Region.NORTH_AMERICA).
↪get();
        champion = Champion.withId(1).withRegion(Region.NORTH_AMERICA).get();

        // Items
        Items items = Items.withRegion(Region.NORTH_AMERICA).get();

        Item item = Item.named("Infinity Edge").withRegion(Region.NORTH_AMERICA).
↪get();
        item = Item.withId(3031).withRegion(Region.NORTH_AMERICA).get();

        // Language Strings
        LanguageStrings languageStrings = LanguageStrings.withRegion(Region.NORTH_
↪AMERICA).get();

        // Languages
        Languages languages = Languages.withRegion(Region.NORTH_AMERICA).get();

        // Maps
        Maps maps = Maps.withRegion(Region.NORTH_AMERICA).get();

        Map map = Map.named("Howling Abyss").withRegion(Region.NORTH_AMERICA).get();
        map = Map.withId(12).withRegion(Region.NORTH_AMERICA).get();

        // Masteries
        Masteries masteries = Masteries.withRegion(Region.NORTH_AMERICA).get();

        Mastery mastery = Mastery.named("Warlord's Bloodlust").withVersion("7.23.1").
↪withRegion(Region.NORTH_AMERICA).get();
        mastery = Mastery.withId(6161).withVersion("7.23.1").withRegion(Region.NORTH_
↪AMERICA).get();

        // Patches
        Patches patches = Patches.withRegion(Region.NORTH_AMERICA).get();
        Patch latestPatch = patches.get(0);

        Patch patch = Patch.named("8.9").withRegion(Region.NORTH_AMERICA).get();

        // Profile Icons
        ProfileIcons profileIcons = ProfileIcons.withRegion(Region.NORTH_AMERICA).
↪get();

        ProfileIcon profileIcon = ProfileIcon.withId(4).withRegion(Region.NORTH_
↪AMERICA).get();

        // Realms
        Realm realm = Realm.withRegion(Region.NORTH_AMERICA).get();

        // Reforged Runes
```

(continues on next page)

(continued from previous page)

```
        ReforgedRunes reforgedRunes = ReforgedRunes.withRegion(Region.NORTH_AMERICA).
↪get();

        ReforgedRune reforgedRune = ReforgedRune.named("Electrocute").
↪withRegion(Region.NORTH_AMERICA).get();
        reforgedRune = ReforgedRune.withId(8112).withRegion(Region.NORTH_AMERICA).
↪get();

        // Runes
        Runes runes = Runes.withRegion(Region.NORTH_AMERICA).get();

        Rune rune = Rune.named("Greater Quintessence of Attack Speed").withVersion("7.
↪23.1").withRegion(Region.NORTH_AMERICA).get();
        rune = Rune.withId(5337).withVersion("7.23.1").withRegion(Region.NORTH_
↪AMERICA).get();

        // Summoner Spells
        SummonerSpells summonerSpells = SummonerSpells.withRegion(Region.NORTH_
↪AMERICA).get();

        SummonerSpell summonerSpell = SummonerSpell.named("Flash").withRegion(Region.
↪NORTH_AMERICA).get();
        summonerSpell = SummonerSpell.withId(4).withRegion(Region.NORTH_AMERICA).
↪get();

        // Versions
        Versions versions = Versions.withRegion(Region.NORTH_AMERICA).get();
    }
}
```

Status API

```
import java.util.List;

import com.merakianalytics.orianna.types.common.Platform;
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.status.ShardStatus;
import com.merakianalytics.orianna.types.core.status.ShardStatuses;

public class Example {
    public static void main(String[] args) {
        ShardStatus status = ShardStatus.withRegion(Region.NORTH_AMERICA).get();
        List<ShardStatus> statuses = ShardStatuses.withPlatforms(Platform.NORTH_
↪AMERICA, Platform.EUROPE_WEST).get();
    }
}
```

Match API

Note that when using `MatchHistory` in the Match API, Orianna loads your paginated MatchList from the Riot API as needed automatically in the background, allowing `MatchHistory` to give you access to your entire Match History with one request.

```
import java.util.List;

import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.match.Match;
import com.merakianalytics.orianna.types.core.match.MatchHistories;
import com.merakianalytics.orianna.types.core.match.MatchHistory;
import com.merakianalytics.orianna.types.core.match.Matches;
import com.merakianalytics.orianna.types.core.match.Timeline;
import com.merakianalytics.orianna.types.core.match.Timelines;
import com.merakianalytics.orianna.types.core.match.TournamentMatches;
import com.merakianalytics.orianna.types.core.summoner.Summoner;
import com.merakianalytics.orianna.types.core.summoner.Summoners;

public class Example {
    public static void main(String[] args) {
        TournamentMatches tournamentMatches = TournamentMatches.forTournamentCode(
↪"YOUR-TOURNAMENT-CODE").withRegion(Region.NORTH_AMERICA).get();
        List<TournamentMatches> manyTournamentMatches = TournamentMatches.
↪forTournamentCodes("TOURNAMENT-CODE-ONE", "TOURNAMENT-CODE-TWO").withRegion(Region.
↪NORTH_AMERICA).get();

        Match match = Match.withId(2718292415L).withRegion(Region.NORTH_AMERICA).
↪get();
        List<Match> matches = Matches.withIds(2718292415L, 2718244702L).
↪withRegion(Region.NORTH_AMERICA).get();

        List<Summoner> summoners = Summoners.named("FatalElement", "Kalturi").
↪withRegion(Region.NORTH_AMERICA).get();

        MatchHistory history = MatchHistory.forSummoner(summoners.get(0)).get();
        List<MatchHistory> histories = MatchHistories.forSummoners(summoners).get();

        MatchHistory recentHistory = MatchHistory.forSummoner(summoners.get(0)).
↪fromRecentMatches().get();
        List<MatchHistory> recentHistories = MatchHistories.forSummoners(summoners).
↪fromRecentMatches().get();

        Timeline timeline = Timeline.withId(2718292415L).withRegion(Region.NORTH_
↪AMERICA).get();
        List<Timeline> timelines = Timelines.withIds(2718292415L, 2718244702L).
↪withRegion(Region.NORTH_AMERICA).get();
    }
}
```

Spectator API

```
import java.util.List;

import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.spectator.CurrentMatch;
import com.merakianalytics.orianna.types.core.spectator.CurrentMatches;
import com.merakianalytics.orianna.types.core.spectator.FeaturedMatches;
import com.merakianalytics.orianna.types.core.summoner.Summoner;
import com.merakianalytics.orianna.types.core.summoner.Summoners;
```

(continues on next page)

(continued from previous page)

```
public class Example {
    public static void main(String[] args) {
        List<Summoner> summoners = Summoners.named("FatalElement", "Kalturi").
↪withRegion(Region.NORTH_AMERICA).get();

        CurrentMatch match = CurrentMatch.forSummoner(summoners.get(0)).get();
        boolean inGame = match.exists();

        List<CurrentMatch> matches = CurrentMatches.forSummoners(summoners).get();

        FeaturedMatches featuredMatches = FeaturedMatches.withRegion(Region.NORTH_
↪AMERICA).get();
        List<FeaturedMatches> manyFeaturedMatches = FeaturedMatches.
↪withRegions(Region.NORTH_AMERICA, Region.EUROPE_WEST).get();
    }
}
```

Summoner API

```
import java.util.List;

import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.summoner.Summoner;
import com.merakianalytics.orianna.types.core.summoner.Summoners;

public class Example {
    public static void main(String[] args) {
        Summoner summoner = Summoner.named("FatalElement").withRegion(Region.NORTH_
↪AMERICA).get();
        summoner = Summoner.withId(22508641L).withRegion(Region.NORTH_AMERICA).get();
        summoner = Summoner.withAccountId(36321079L).withRegion(Region.NORTH_AMERICA).
↪get();

        List<Summoner> summoners = Summoners.named("FatalElement", "Kalturi").
↪withRegion(Region.NORTH_AMERICA).get();
        summoners = Summoners.withIds(22508641L, 21359666L).withRegion(Region.NORTH_
↪AMERICA).get();
        summoners = Summoners.withAccountIds(36321079L, 34718348L).withRegion(Region.
↪NORTH_AMERICA).get();
    }
}
```

Third Party Code API

```
import java.util.List;

import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.summoner.Summoner;
import com.merakianalytics.orianna.types.core.summoner.Summoners;
import com.merakianalytics.orianna.types.core.thirdpartycode.VerificationString;
import com.merakianalytics.orianna.types.core.thirdpartycode.VerificationStrings;
```

(continues on next page)

(continued from previous page)

```
public class Example {
    public static void main(String[] args) {
        List<Summoner> summoners = Summoners.named("FatalElement", "Kalturi").
↪withRegion(Region.NORTH_AMERICA).get();

        VerificationString verificationString = VerificationString.
↪forSummoner(summoners.get(0)).get();
        List<VerificationString> verificationStrings = VerificationStrings.
↪forSummoners(summoners).get();
    }
}
```

Using the Orianna Class API

See the JavaDocs for the Orianna Class for the full details of the Orianna Class API. Examples of getting the available API data are shown below.

Champion Mastery API

```
import java.util.List;

import com.merakianalytics.orianna.Orianna;
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.championmastery.ChampionMasteries;
import com.merakianalytics.orianna.types.core.championmastery.ChampionMastery;
import com.merakianalytics.orianna.types.core.championmastery.ChampionMasteryScore;
import com.merakianalytics.orianna.types.core.staticdata.Champion;
import com.merakianalytics.orianna.types.core.summoner.Summoner;

public class Example {
    public static void main(String[] args) {
        List<Summoner> summoners = Orianna.summonersNamed("FatalElement", "Kalturi").
↪withRegion(Region.NORTH_AMERICA).get();
        List<Champion> champions = Orianna.championsNamed("Annie", "Thresh").
↪withRegion(Region.NORTH_AMERICA).get();

        ChampionMasteries masteries = Orianna.championMasteriesForSummoner(summoners.
↪get(0)).get();
        ChampionMastery mastery = Orianna.championMasteryForSummoner(summoners.
↪get(0)).withChampion(champions.get(0)).get();
        ChampionMasteryScore score = Orianna.
↪championMasteryScoreForSummoner(summoners.get(0)).get();

        List<ChampionMasteries> manyMasteries = Orianna.
↪championMasteriesForSummoners(summoners).get();
        List<ChampionMastery> manyMastery = Orianna.
↪championMasteriesForSummoner(summoners.get(0)).withChampions(champions).get();
        List<ChampionMasteryScore> manyScores = Orianna.
↪championMasteryScoresForSummoners(summoners).get();
    }
}
```

Champion Status API

Champion Status information is available directly on the Champion within Orianna.

```
import com.merakianalytics.orianna.Orianna;
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.staticdata.Champions;

public class Example {
    public static void main(String[] args) {
        Champions champions = Orianna.championsWithRegion(Region.NORTH_AMERICA).get();

        boolean firstFree = champions.get(0).isFreeToPlay();
        boolean lastEnabled = champions.get(champions.size() - 1).isEnabled();
    }
}
```

League API

```
import java.util.List;
import java.util.stream.Collectors;

import com.merakianalytics.orianna.Orianna;
import com.merakianalytics.orianna.types.common.Queue;
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.league.League;
import com.merakianalytics.orianna.types.core.league.LeaguePosition;
import com.merakianalytics.orianna.types.core.league.LeaguePositions;
import com.merakianalytics.orianna.types.core.summoner.Summoner;

public class Example {
    public static void main(String[] args) {
        League challenger = Orianna.challengerLeagueInQueue(Queue.RANKED_SOLO_5x5).
↪withRegion(Region.NORTH_AMERICA).get();
        League master = Orianna.masterLeagueInQueue(Queue.RANKED_SOLO_5x5).
↪withRegion(Region.NORTH_AMERICA).get();

        List<League> allChallenger = Orianna.challengerLeaguesInQueues(Queue.RANKED).
↪withRegion(Region.NORTH_AMERICA).get();
        List<League> allMaster = Orianna.masterLeaguesInQueues(Queue.RANKED).
↪withRegion(Region.NORTH_AMERICA).get();

        List<Summoner> summoners = Orianna.summonersNamed("FatalElement", "Kalturi").
↪withRegion(Region.NORTH_AMERICA).get();

        LeaguePositions positions = Orianna.leaguePositionsForSummoner(summoners.
↪get(1)).get();
        List<LeaguePositions> manyPositions = Orianna.
↪leaguePositionsForSummoners(summoners).get();

        League league = positions.get(0).getLeague();
        List<League> manyLeagues = positions.stream().map(LeaguePosition::getLeague).
↪collect(Collectors.toList());

        String leagueId = league.getId();
```

(continues on next page)

(continued from previous page)

```
        List<String> manyLeagueIds = manyLeagues.stream().map(League::getId).
↪collect(Collectors.toList());

        league = Orianna.leagueWithId(leagueId).withRegion(Region.NORTH_AMERICA).
↪get();
        manyLeagues = Orianna.leaguesWithIds(manyLeagueIds).withRegion(Region.NORTH_
↪AMERICA).get();
    }
}
```

Static Data API

```
import com.merakianalytics.orianna.Orianna;
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.staticdata.Champion;
import com.merakianalytics.orianna.types.core.staticdata.Champions;
import com.merakianalytics.orianna.types.core.staticdata.Item;
import com.merakianalytics.orianna.types.core.staticdata.Items;
import com.merakianalytics.orianna.types.core.staticdata.LanguageStrings;
import com.merakianalytics.orianna.types.core.staticdata.Languages;
import com.merakianalytics.orianna.types.core.staticdata.Map;
import com.merakianalytics.orianna.types.core.staticdata.Maps;
import com.merakianalytics.orianna.types.core.staticdata.Masteries;
import com.merakianalytics.orianna.types.core.staticdata.Mastery;
import com.merakianalytics.orianna.types.core.staticdata.ProfileIcon;
import com.merakianalytics.orianna.types.core.staticdata.ProfileIcons;
import com.merakianalytics.orianna.types.core.staticdata.Realm;
import com.merakianalytics.orianna.types.core.staticdata.ReforgedRune;
import com.merakianalytics.orianna.types.core.staticdata.ReforgedRunes;
import com.merakianalytics.orianna.types.core.staticdata.Rune;
import com.merakianalytics.orianna.types.core.staticdata.Runes;
import com.merakianalytics.orianna.types.core.staticdata.SummonerSpell;
import com.merakianalytics.orianna.types.core.staticdata.SummonerSpells;
import com.merakianalytics.orianna.types.core.staticdata.Versions;

public class Example {
    public static void main(String[] args) {
        // Champions
        Champions champions = Orianna.championsWithRegion(Region.NORTH_AMERICA).get();

        Champion champion = Orianna.championNamed("Annie").withRegion(Region.NORTH_
↪AMERICA).get();
        champion = Orianna.championWithId(1).withRegion(Region.NORTH_AMERICA).get();

        // Items
        Items items = Orianna.itemsWithRegion(Region.NORTH_AMERICA).get();

        Item item = Orianna.itemNamed("Infinity Edge").withRegion(Region.NORTH_
↪AMERICA).get();
        item = Orianna.itemWithId(3031).withRegion(Region.NORTH_AMERICA).get();

        // Language Strings
        LanguageStrings languageStrings = Orianna.languageStringsWithRegion(Region.
↪NORTH_AMERICA).get();
```

(continues on next page)

(continued from previous page)

```
        // Languages
        Languages languages = Orianna.languagesWithRegion(Region.NORTH_AMERICA).get();

        // Maps
        Maps maps = Orianna.mapsWithRegion(Region.NORTH_AMERICA).get();

        Map map = Orianna.mapNamed("Howling Abyss").withRegion(Region.NORTH_AMERICA).
↪get();
        map = Orianna.mapWithId(12).withRegion(Region.NORTH_AMERICA).get();

        // Masteries
        Masteries masteries = Orianna.masteriesWithRegion(Region.NORTH_AMERICA).get();

        Mastery mastery = Orianna.masteryNamed("Warlord's Bloodlust").withVersion("7.
↪23.1").withRegion(Region.NORTH_AMERICA).get();
        mastery = Orianna.masteryWithId(6161).withVersion("7.23.1").withRegion(Region.
↪NORTH_AMERICA).get();

        // Patches
        Patches patches = Orianna.patchesWithRegion(Region.NORTH_AMERICA).get();
        Patch latestPatch = patches.get(0);

        Patch patch = Orianna.patchNamed("8.9").withRegion(Region.NORTH_AMERICA).
↪get();

        // Profile Icons
        ProfileIcons profileIcons = Orianna.profileIconsWithRegion(Region.NORTH_
↪AMERICA).get();

        ProfileIcon profileIcon = Orianna.profileIconWithId(4).withRegion(Region.
↪NORTH_AMERICA).get();

        // Realms
        Realm realm = Orianna.realmWithRegion(Region.NORTH_AMERICA).get();

        // Reforged Runes
        ReforgedRunes reforgedRunes = Orianna.reforgedRunesWithRegion(Region.NORTH_
↪AMERICA).get();

        ReforgedRune reforgedRune = Orianna.reforgedRuneNamed("Electrocute").
↪withRegion(Region.NORTH_AMERICA).get();
        reforgedRune = Orianna.reforgedRuneWithId(8112).withRegion(Region.NORTH_
↪AMERICA).get();

        // Runes
        Runes runes = Orianna.runesWithRegion(Region.NORTH_AMERICA).get();

        Rune rune = Orianna.runeNamed("Greater Quintessence of Attack Speed").
↪withVersion("7.23.1").withRegion(Region.NORTH_AMERICA).get();
        rune = Orianna.runeWithId(5337).withVersion("7.23.1").withRegion(Region.NORTH_
↪AMERICA).get();

        // Summoner Spells
        SummonerSpells summonerSpells = Orianna.summonerSpellsWithRegion(Region.NORTH_
↪AMERICA).get();
```

(continues on next page)

(continued from previous page)

```
        SummonerSpell summonerSpell = Orianna.summonerSpellNamed("Flash").
↪withRegion(Region.NORTH_AMERICA).get();
        summonerSpell = Orianna.summonerSpellWithId(4).withRegion(Region.NORTH_
↪AMERICA).get();

        // Versions
        Versions versions = Orianna.versionsWithRegion(Region.NORTH_AMERICA).get();
    }
}
```

Status API

```
import java.util.List;

import com.merakianalytics.orianna.Orianna;
import com.merakianalytics.orianna.types.common.Platform;
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.status.ShardStatus;

public class Example {
    public static void main(String[] args) {
        ShardStatus status = Orianna.shardStatusWithRegion(Region.NORTH_AMERICA).
↪get();
        List<ShardStatus> statuses = Orianna.shardStatusesWithPlatforms(Platform.
↪NORTH_AMERICA, Platform.EUROPE_WEST).get();
    }
}
```

Match API

Note that when using `MatchHistory` in the Match API, Orianna loads your paginated MatchList from the Riot API as needed automatically in the background, allowing `MatchHistory` to give you access to your entire Match History with one request.

```
import java.util.List;

import com.merakianalytics.orianna.Orianna;
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.match.Match;
import com.merakianalytics.orianna.types.core.match.MatchHistory;
import com.merakianalytics.orianna.types.core.match.Timeline;
import com.merakianalytics.orianna.types.core.match.TournamentMatches;
import com.merakianalytics.orianna.types.core.summoner.Summoner;

public class Example {
    public static void main(String[] args) {
        TournamentMatches tournamentMatches = Orianna.
↪tournamentMatchesForTournamentCode("YOUR-TOURNAMENT-CODE").withRegion(Region.NORTH_
↪AMERICA).get();
        List<TournamentMatches> manyTournamentMatches = Orianna.
↪tournamentMatchesForTournamentCodes("TOURNAMENT-CODE-ONE", "TOURNAMENT-CODE-TWO").
↪withRegion(Region.NORTH_AMERICA).get();

```

(continues on next page)

(continued from previous page)

```
        Match match = Orianna.matchWithId(2718292415L).withRegion(Region.NORTH_
↪AMERICA).get();
        List<Match> matches = Orianna.matchesWithIds(2718292415L, 2718244702L).
↪withRegion(Region.NORTH_AMERICA).get();

        List<Summoner> summoners = Orianna.summonersNamed("FatalElement", "Kalturi").
↪withRegion(Region.NORTH_AMERICA).get();

        MatchHistory history = Orianna.matchHistoryForSummoner(summoners.get(0)).
↪get();
        List<MatchHistory> histories = Orianna.matchHistoriesForSummoners(summoners).
↪get();

        MatchHistory recentHistory = Orianna.matchHistoryForSummoner(summoners.
↪get(0)).fromRecentMatches().get();
        List<MatchHistory> recentHistories = Orianna.
↪matchHistoriesForSummoners(summoners).fromRecentMatches().get();

        Timeline timeline = Orianna.timelineWithId(2718292415L).withRegion(Region.
↪NORTH_AMERICA).get();
        List<Timeline> timelines = Orianna.timelinesWithIds(2718292415L, 2718244702L).
↪withRegion(Region.NORTH_AMERICA).get();
    }
}
```

Spectator API

```
import java.util.List;

import com.merakianalytics.orianna.Orianna;
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.spectator.CurrentMatch;
import com.merakianalytics.orianna.types.core.spectator.FeaturedMatches;
import com.merakianalytics.orianna.types.core.summoner.Summoner;

public class Example {
    public static void main(String[] args) {
        List<Summoner> summoners = Orianna.summonersNamed("FatalElement", "Kalturi").
↪withRegion(Region.NORTH_AMERICA).get();

        CurrentMatch match = Orianna.currentMatchForSummoner(summoners.get(0)).get();
        boolean inGame = match.exists();

        List<CurrentMatch> matches = Orianna.currentMatchesForSummoners(summoners).
↪get();

        FeaturedMatches featuredMatches = Orianna.featuredMatchesWithRegion(Region.
↪NORTH_AMERICA).get();
        List<FeaturedMatches> manyFeaturedMatches = Orianna.
↪featuredMatchesWithRegions(Region.NORTH_AMERICA, Region.EUROPE_WEST).get();
    }
}
```

Summoner API

```
import java.util.List;

import com.merakianalytics.orianna.Orianna;
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.summoner.Summoner;

public class Example {
    public static void main(String[] args) {
        Summoner summoner = Orianna.summonerNamed("FatalElement").withRegion(Region.
↪NORTH_AMERICA).get();
        summoner = Orianna.summonerWithId(22508641L).withRegion(Region.NORTH_AMERICA).
↪get();
        summoner = Orianna.summonerWithAccountId(36321079L).withRegion(Region.NORTH_
↪AMERICA).get();

        List<Summoner> summoners = Orianna.summonersNamed("FatalElement", "Kalturi").
↪withRegion(Region.NORTH_AMERICA).get();
        summoners = Orianna.summonersWithIds(22508641L, 21359666L).withRegion(Region.
↪NORTH_AMERICA).get();
        summoners = Orianna.summonersWithAccountIds(36321079L, 34718348L).
↪withRegion(Region.NORTH_AMERICA).get();
    }
}
```

Third Party Code API

```
import java.util.List;

import com.merakianalytics.orianna.Orianna;
import com.merakianalytics.orianna.types.common.Region;
import com.merakianalytics.orianna.types.core.summoner.Summoner;
import com.merakianalytics.orianna.types.core.thirdpartycode.VerificationString;

public class Example {
    public static void main(String[] args) {
        List<Summoner> summoners = Orianna.summonersNamed("FatalElement", "Kalturi").
↪withRegion(Region.NORTH_AMERICA).get();

        VerificationString verificationString = Orianna.
↪verificationStringForSummoner(summoners.get(0)).get();
        List<VerificationString> verificationStrings = Orianna.
↪verificationStringsForSummoners(summoners).get();
    }
}
```

2.3 Configuring Orianna

Orianna is highly configurable to meet the needs of your application. Configuration can be done using either a JSON-based configuration file, or the programmatic configuration API.

Orianna ships with a default JSON-based configuration file which is designed to get users up & running as quickly

and easily as possible, but you may find while developing your application that you'd prefer different behavior from Orianna's defaults.

This page is a guide to what configuration options are available in Orianna, and how to use them through both the JSON-based configuration file and the programmatic configuration API.

Before we dive into the two different configuration approaches, let's touch on one aspect of configuration that's shared between them:

2.3.1 Setting Your `RIOT_API_KEY` Environment Variable

In most applications using Orianna (though not necessarily all of them), you'll be making requests to the Riot API, and will therefore need to use your API Key.

You can set your API key using either of the configuration approaches, but that makes it easy to forget to remove it before committing code to a shared repository, and that's no bueno. As an alternative, you can leave the API Key out of your Orianna configuration and set your `RIOT_API_KEY` environment variable to it instead. The RiotAPI DataSource will automatically check this environment variable when it loads to get your API Key without risking committing it to a shared repository.

2.3.2 Choose Your Configuration Method

Next, choose which approach you'd like to use to configure Orianna. By using the JSON-based configuration file, you can change your configuration without changing your application code. By using the programmatic configuration API, you can change options easily during development:

JSON-Based Configuration File

Below, we'll discuss each element of the JSON configuration, what it does, and what the available options are for that setting. First, though, let's talk about about how to create and load a `config.json` file.

Creating and Loading Your Configuration

The first step in writing your own `config.json` is to download the default configuration file, `available here`. Use the default configuration as a base for your `config.json` file, adding, removing, and editing settings as needed for your application. You can save your `config.json` wherever you'd like on your computer, but it's recommended to put it in your application resources directory. This is usually the `src/main/resources` directory within your Java project.

Once you've finalized your `config.json`, you can have Orianna load it using one of the following methods. Add one of these calls to the initialization of your application to ensure Orianna is correctly configured.

```
// Use this method if you placed your config.json in your application resources␣
↪directory
Orianna.loadConfiguration("config.json");

// Use this method if you placed your config.json somewhere else on your computer
Orianna.loadConfiguration(new File("/path/to/your/config.json"));
```

Alternatively, if you set your `ORIANNA_CONFIGURATION_PATH` environment variable to the path to your `config.json`, Orianna will automatically pick it up when it loads.

Full Default Configuration

Now that we know how to load a JSON configuration, let's break down the default configuration to see what settings are available in Orianna. First, here's the full default configuration:

```
{
  "currentVersionExpiration": {
    "period": 6,
    "unit": "HOURS"
  },
  "pipeline": {
    "elements": [
      {
        "className": "com.merakianalytics.orianna.datapipeline.InMemoryCache",
        "config": {
          "expirationPeriods": {
            "com.merakianalytics.orianna.types.core.champion.ChampionRotation": {
              "period": 6,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.championmastery.ChampionMastery":
↪{
              "period": 15,
              "unit": "MINUTES"
            },
            "com.merakianalytics.orianna.types.core.championmastery.ChampionMasteries
↪": {
              "period": 15,
              "unit": "MINUTES"
            },
            "com.merakianalytics.orianna.types.core.championmastery.
↪ChampionMasteryScore": {
              "period": 15,
              "unit": "MINUTES"
            },
            "com.merakianalytics.orianna.types.core.staticdata.Champion": {
              "period": 6,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.staticdata.Champions": {
              "period": 6,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.spectator.CurrentMatch": {
              "period": 5,
              "unit": "MINUTES"
            },
            "com.merakianalytics.orianna.types.core.spectator.FeaturedMatches": {
              "period": 5,
              "unit": "MINUTES"
            },
            "com.merakianalytics.orianna.types.core.staticdata.Item": {
              "period": 6,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.staticdata.Items": {
              "period": 6,
```

(continues on next page)

(continued from previous page)

```
    "unit": "HOURS"
  },
  "com.merakianalytics.orianna.types.core.staticdata.LanguageStrings": {
    "period": 6,
    "unit": "HOURS"
  },
  "com.merakianalytics.orianna.types.core.staticdata.Languages": {
    "period": 6,
    "unit": "HOURS"
  },
  "com.merakianalytics.orianna.types.core.league.League": {
    "period": 15,
    "unit": "MINUTES"
  },
  "com.merakianalytics.orianna.types.core.league.LeaguePositions": {
    "period": 15,
    "unit": "MINUTES"
  },
  "com.merakianalytics.orianna.types.core.staticdata.Map": {
    "period": 6,
    "unit": "HOURS"
  },
  "com.merakianalytics.orianna.types.core.staticdata.Maps": {
    "period": 6,
    "unit": "HOURS"
  },
  "com.merakianalytics.orianna.types.core.staticdata.Mastery": {
    "period": 6,
    "unit": "HOURS"
  },
  "com.merakianalytics.orianna.types.core.staticdata.Masteries": {
    "period": 6,
    "unit": "HOURS"
  },
  "com.merakianalytics.orianna.types.core.match.Match": {
    "period": 6,
    "unit": "HOURS"
  },
  "com.merakianalytics.orianna.types.core.staticdata.Patch": {
    "period": 6,
    "unit": "HOURS"
  },
  "com.merakianalytics.orianna.types.core.staticdata.Patches": {
    "period": 6,
    "unit": "HOURS"
  },
  "com.merakianalytics.orianna.types.core.staticdata.ProfileIcon": {
    "period": 6,
    "unit": "HOURS"
  },
  "com.merakianalytics.orianna.types.core.staticdata.ProfileIcons": {
    "period": 6,
    "unit": "HOURS"
  },
  "com.merakianalytics.orianna.types.core.staticdata.Realm": {
    "period": 6,
    "unit": "HOURS"
```

(continues on next page)

(continued from previous page)

```
            },
            "com.merakianalytics.orianna.types.core.staticdata.ReforgedRune": {
              "period": 6,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.staticdata.ReforgedRunes": {
              "period": 6,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.staticdata.Rune": {
              "period": 6,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.staticdata.Runes": {
              "period": 6,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.status.ShardStatus": {
              "period": 15,
              "unit": "MINUTES"
            },
            "com.merakianalytics.orianna.types.core.staticdata.SummonerSpell": {
              "period": 6,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.staticdata.SummonerSpells": {
              "period": 6,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.summoner.Summoner": {
              "period": 1,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.match.Timeline": {
              "period": 6,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.match.TournamentMatches": {
              "period": 6,
              "unit": "HOURS"
            },
            "com.merakianalytics.orianna.types.core.thirdpartycode.VerificationString
↪": {
              "period": 3,
              "unit": "MINUTES"
            },
            "com.merakianalytics.orianna.types.core.staticdata.Versions": {
              "period": 6,
              "unit": "HOURS"
            }
          }
        },
        "configClassName": "com.merakianalytics.orianna.datapipeline.InMemoryCache
↪$Configuration"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.GhostLoader"
```

(continues on next page)

(continued from previous page)

```
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.MerakiAnalyticsCDN",
        "config": {
          "host": "cdn.merakianalytics.com",
          "cacheDuration": {
            "period": 6,
            "unit": "HOURS"
          },
          "requests": {
            "connectTimeout": 3,
            "connectTimeoutUnit": "SECONDS",
            "rateLimiterTimeout": -1,
            "rateLimiterTimeoutUnit": "DAYS",
            "readTimeout": 3,
            "readTimeoutUnit": "SECONDS",
            "https": true
          }
        },
        "configClassName": "com.merakianalytics.orianna.datapipeline.
↪MerakiAnalyticsCDN$Configuration"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.DataDragon",
        "config": {
          "cacheDuration": {
            "period": 6,
            "unit": "HOURS"
          },
          "requests": {
            "connectTimeout": 3,
            "connectTimeoutUnit": "SECONDS",
            "rateLimiterTimeout": -1,
            "rateLimiterTimeoutUnit": "DAYS",
            "readTimeout": 3,
            "readTimeoutUnit": "SECONDS",
            "https": true
          }
        },
        "configClassName": "com.merakianalytics.orianna.datapipeline.DataDragon
↪$Configuration"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.riotapi.RiotAPI",
        "config": {
          "http404Strategy": {
            "type": "RETURN_NULL"
          },
          "http429Strategy": {
            "backupStrategy": {
              "backoff": 1,
              "backoffUnit": "SECONDS",
              "backupStrategy": {
                "type": "THROW_EXCEPTION"
              },
              "maxAttempts": 4,
              "backoffFactor": 2,
```

(continues on next page)

(continued from previous page)


```
      "type": "EXPONENTIAL_BACKOFF"
    },
    "type": "RETRY_FROM_HEADERS"
  },
  "http500Strategy": {
    "backoff": 1,
    "backoffUnit": "SECONDS",
    "backupStrategy": {
      "type": "THROW_EXCEPTION"
    },
    "maxAttempts": 4,
    "backoffFactor": 2,
    "type": "EXPONENTIAL_BACKOFF"
  },
  "http503Strategy": {
    "backoff": 1,
    "backoffUnit": "SECONDS",
    "backupStrategy": {
      "type": "THROW_EXCEPTION"
    },
    "maxAttempts": 4,
    "backoffFactor": 2,
    "type": "EXPONENTIAL_BACKOFF"
  },
  "http504Strategy": {
    "backoff": 1,
    "backoffUnit": "SECONDS",
    "backupStrategy": {
      "type": "THROW_EXCEPTION"
    },
    "maxAttempts": 4,
    "backoffFactor": 2,
    "type": "EXPONENTIAL_BACKOFF"
  },
  "httpTimeoutStrategy": {
    "backoff": 1,
    "backoffUnit": "SECONDS",
    "backupStrategy": {
      "type": "THROW_EXCEPTION"
    },
    "maxAttempts": 4,
    "backoffFactor": 2,
    "type": "EXPONENTIAL_BACKOFF"
  },
  "limitingShare": 1.0,
  "limitingType": "BURST",
  "rateLimiterTimeoutStrategy": {
    "type": "THROW_EXCEPTION"
  },
  "requests": {
    "connectTimeout": 3,
    "connectTimeoutUnit": "SECONDS",
    "rateLimiterTimeout": -1,
    "rateLimiterTimeoutUnit": "DAYS",
    "readTimeout": 3,
    "readTimeoutUnit": "SECONDS",
    "https": true
```


(continues on next page)

(continued from previous page)

```
          },
          "services": [
            "com.merakianalytics.orianna.datapipeline.riotapi.ChampionAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.ChampionMasteryAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.LeagueAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.MatchAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.SpectatorAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.StatusAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.SummonerAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.ThirdPartyCodeAPI"
          ]
        },
        "configClassName": "com.merakianalytics.orianna.datapipeline.riotapi.RiotAPI
↪$Configuration"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.ImageDownloader"
      }
    ],
    "transformers": [
      {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪ChampionMasteryTransformer"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪ChampionTransformer"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪LeagueTransformer"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪MatchTransformer"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪SpectatorTransformer"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪StaticDataTransformer"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪StatusTransformer"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪SummonerTransformer"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪ThirdPartyCodeTransformer"
      }
    ]
```

(continues on next page)

(continued from previous page)

```
  }
}
```

Configuration Options

There's a lot here, so let's break it down piece by piece and see what options are available.

Default Platform

Normally when making calls using Orianna, you have to specify which Region/Platform you'd like to do the request for. If you're primarily using one Platform/Region in your application, you can set it as the default to be used if no Platform/Region is included in the request.

```
{
  "defaultPlatform": "NORTH_AMERICA"
}
```

The full set of available options for the `defaultPlatform` can be found here.

Default Locale

When requesting static-data from Orianna, data must be requested with a specific locale, which determines what language the returned data will be in. If a `defaultLocale` isn't set, the default locale for the Platform/Region the request is for will be used.

```
{
  "defaultLocale": "en_US"
}
```

You can check the full list of available locales for a Platform/Region with the following Orianna call:

```
System.out.println(Languages.withPlatform(Platform.NORTH_AMERICA).get());
```

Current Version Expiration

Whenever static-data is requested in Orianna without a version specified, it defaults to using the current version for the data. Orianna caches the current version for each Platform/Region to minimize how often it has to be requested from Orianna's DataSources.

```
{
  "currentVersionExpiration": {
    "period": 6,
    "unit": "HOURS"
  }
}
```

If you set `period < 0` it will be cached until your Java process terminates. If you set `period == 0` it won't be cached and will be requested every time it's needed.

Data Pipeline

The last, and most complicated pieces of the Orianna configuration is the `pipeline`. It determines how data is retrieved and cached within Orianna. Here's what the skeleton of a `pipeline` looks like with nothing in it:

```
{
  "pipeline": {
    "elements": [],
    "transformers": []
  }
}
```

Please check out *the Data Pipeline page* for the details about what `elements` and `transformers` are available and when to use them. For example purposes, let's add the RiotAPI as a DataSource in the `pipeline` to see how to add elements. We'll use the minimal settings for illustration purposes.

```
{
  "pipeline": {
    "elements": [
      {
        "className": "com.merakianalytics.orianna.datapipeline.riotapi.RiotAPI",
        "configClassName": "com.merakianalytics.orianna.datapipeline.riotapi.RiotAPI
↪$Configuration",
        "config": {
          "services": [
            "com.merakianalytics.orianna.datapipeline.riotapi.ChampionAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.ChampionMasteryAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.LeagueAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.MatchAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.SpectatorAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.StaticDataAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.StatusAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.SummonerAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.ThirdPartyCodeAPI"
          ]
        }
      }
    ],
    "transformers": []
  }
}
```

There are essentially 3 parts to each `pipeline` element:

1) `className` is set to the fully qualified Java class name of the desired Pipeline Element.
2) `configClassName` is set to the fully qualified Java class name of the Pipeline Element's configuration class. It's assumed that the Pipeline Element class will have a constructor which takes just one argument of this type, or if no `config` is included, it must have a no-argument constructor.
3) `config` is set to the configuration to use for the Pipeline Element. This will be deserialized into the `configClassName` class and passed as the single argument to the constructor for `className` .

We also need to add the GhostObjectSource and the dto-to-data Transformers to make Orianna usable with just the RiotAPI in the pipeline.

```
{
  "pipeline": {
```

(continues on next page)

(continued from previous page)

```
    "elements": [
      {
        "className": "com.merakianalytics.orianna.datapipeline.GhostObjectSource"
      },
      {
        "className": "com.merakianalytics.orianna.datapipeline.riotapi.RiotAPI",
        "configClassName": "com.merakianalytics.orianna.datapipeline.riotapi.RiotAPI
↪$Configuration",
        "config": {
          "services": [
            "com.merakianalytics.orianna.datapipeline.riotapi.ChampionAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.ChampionMasteryAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.LeagueAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.MatchAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.SpectatorAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.StaticDataAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.StatusAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.SummonerAPI",
            "com.merakianalytics.orianna.datapipeline.riotapi.ThirdPartyCodeAPI"
          ]
        }
      }
    ],
    "transformers": [
      {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪ChampionMasteryTransformer"
      }, {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪ChampionTransformer"
      }, {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪LeagueTransformer"
      }, {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪MatchTransformer"
      }, {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪SpectatorTransformer"
      }, {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪StaticDataTransformer"
      }, {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪StatusTransformer"
      }, {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪SummonerTransformer"
      }, {
        "className": "com.merakianalytics.orianna.datapipeline.transformers.dtodata.
↪ThirdPartyCodeTransformer"
      }
    ]
  }
}
```

Each `transformers` element only has 1 part, the `className`. It's assumed that Transformers will have no-

argument constructors that will be used to create them.

Hopefully this illustrates how to add a new Pipeline Element or Transformer to Orianna's Data Pipeline. Please check out *the Data Pipeline page* for the details about what `elements` and `transformers` are available and when to use them.

Programmatic Configuration API

Below, we'll discuss each element of the programmatic configuration API, what it does, and what the available options are for that setting. First, though, let's talk about about how to create and load a new Configuration.

Creating and Loading Your Configuration

The first step in setting your configuration is to create a Configuration object.

```
Orianna.Configuration config = new Orianna.Configuration();
```

This will create a configuration with all the default settings. You can make changes using the mutator methods on the Configuration object.

Once you've finalized your Configuration, you can have Orianna load it using the following method. Add this call to the initialization of your application to ensure Orianna is correctly configured.

```
Orianna.loadConfiguration(config);
```

Configuration Options

Now that we know how to load a Configuration, let's break down the options and see what settings are available in Orianna.

Default Platform

Normally when making calls using Orianna, you have to specify which Region/Platform you'd like to do the request for. If you're primarily using one Platform/Region in your application, you can set it as the default to be used if no Platform/Region is included in the request.

```
config.setDefaultPlatform(Platform.NORTH_AMERICA);
```

The full set of available options for the `defaultPlatform` can be found here.

Default Locale

When requesting static-data from Orianna, data must be requested with a specific locale, which determines what language the returned data will be in. If a `defaultLocale` isn't set, the default locale for the Platform/Region the request is for will be used.

```
Orianna.setDefaultLocale("en_US");
```

You can check the full list of available locales for a Platform/Region with the following Orianna call:

```
System.out.println(Languages.withPlatform(Platform.NORTH_AMERICA).get());
```

Current Version Expiration

Whenever static-data is requested in Orianna without a version specified, it defaults to using the current version for the data. Orianna caches the current version for each Platform/Region to minimize how often it has to be requested from Orianna's DataSources.

```
config.setCurrentVersionExpiration(ExpirationPeriod.create(6, TimeUnit.HOURS));
```

If you set `period < 0` it will be cached until your Java process terminates. If you set `period == 0` it won't be cached and will be requested every time it's needed.

Data Pipeline

The last, and most complicated pieces of the Orianna configuration is the `pipeline`. It determines how data is retrieved and cached within Orianna, and is composed of `elements` and `transformers`.

Please check out *the Data Pipeline page* for the details about what `elements` and `transformers` are available and when to use them. For example purposes, let's add the RiotAPI as a DataSource in the `pipeline` to see how to add elements. We'll use the default settings for illustration purposes.

```
PipelineConfiguration pipeline = new PipelineConfiguration();
List<PipelineElementConfiguration> elements = Arrays.asList(new␣
↪PipelineElementConfiguration[] {
    PipelineElementConfiguration.defaultConfiguration(RiotAPI.class)
});
pipeline.setElements(elements);
config.setPipeline(pipeline);
```

If you need to use settings other than default the configuration for a Pipeline Element, it's suggested to use *the JSON-based configuration file* configuration approach instead, as it's far less complicated for detailed configuration changes.

We also need to add the dto-to-data Transformers to make Orianna usable with just the RiotAPI in the pipeline.

```
PipelineConfiguration pipeline = new PipelineConfiguration();

List<PipelineElementConfiguration> elements = Arrays.asList(new␣
↪PipelineElementConfiguration[] {
    PipelineElementConfiguration.defaultConfiguration(GhostObjectSource.class),
    PipelineElementConfiguration.defaultConfiguration(RiotAPI.class)
});
pipeline.setElements(elements);

final Set<TransformerConfiguration> transformers = new HashSet<>(Arrays.asList(new␣
↪TransformerConfiguration[] {
    TransformerConfiguration.defaultConfiguration(ChampionMasteryTransformer.class),
    TransformerConfiguration.defaultConfiguration(ChampionTransformer.class),
    TransformerConfiguration.defaultConfiguration(LeagueTransformer.class),
    TransformerConfiguration.defaultConfiguration(MatchTransformer.class),
    TransformerConfiguration.defaultConfiguration(SpectatorTransformer.class),
    TransformerConfiguration.defaultConfiguration(StaticDataTransformer.class),
    TransformerConfiguration.defaultConfiguration(StatusTransformer.class),
    TransformerConfiguration.defaultConfiguration(SummonerTransformer.class),
```

(continues on next page)

(continued from previous page)

```
    TransformerConfiguration.defaultConfiguration(ThirdPartyCodeTransformer.class)
}));
pipeline.setTransformers(transformers);

config.setPipeline(pipeline);
```

Hopefully this illustrates how to add a new Pipeline Element or Transformer to Orianna's Data Pipeline. Please check out *the Data Pipeline page* for the details about what `elements` and `transformers` are available and when to use them.

2.4 How Orianna Works

Coming Soon!

2.5 Data Pipeline

The Data Pipeline is the core of Orianna's functionality. It's a configurable collection of Data Sources, Databases, and Caches that Orianna gets its API data from. The Data Pipeline automatically figures out where the best place to get the data from is and makes sure that data get stored in the configured Caches and Databases.

The Data Pipeline is organized as an ordered list of PipelineElements. Each PipelineElement is a DataSource (provides data), a DataSink (stores data), or a DataStore (both a DataSource *and* DataSink at once).

When an Orianna user asks for data from Orianna, Orianna sends a query to the Data Pipline for the request data type:

1) The Data Pipeline checks which DataSources can supploy that data type.
2) It queries each DataSource in order until one returns data for the query.
3) It stores the data in all of the DataSinks that came *before* the DataSource that returned the data in the Data Pipeline.
4) It returns the data to Orianna, which returns it to the user.

The default Orianna Data Pipeline looks like this:

```
InMemoryCache -> GhostObjectSource -> DataDragon -> RiotAPI -> ImageDataSource
```

Because Data Dragon only supplies static-data, this means that for static-data requests the Riot API will only be queried if Data Dragon fails to return the data, but other API data requests will be routed directly to the Riot API. For more details about what each of these PipelineElements is for, and what others are available in Orianna, check out the pages below.

For more information about how to configure Orianna's Data Pipeline to fit your needs, see *the configuration documentation*.

2.5.1 Riot API

Coming Soon!

2.5.2 Data Dragon

Coming Soon!

2.5.3 Image Data Source

Coming Soon!

2.5.4 Ghost Object Source

Coming Soon!

2.5.5 In-Memory Cache

Coming Soon!

2.5.6 JetBrains Xodus (Embedded DB)

Coming Soon!

2.5.7 MongoDB

Coming Soon!

CHAPTER 3

A Few Examples

First, we'll quickly and efficiently look up the champion masteries for the summoner "FatalElement" (one of the developers) and print the champions he is best at:

```
import java.util.List;
import java.util.stream.Collectors;

import com.merakianalytics.orianna.Orianna;
import com.merakianalytics.orianna.types.common.Platform;
import com.merakianalytics.orianna.types.core.championmastery.ChampionMasteries;
import com.merakianalytics.orianna.types.core.summoner.Summoner;
import com.merakianalytics.orianna.types.core.championmastery.ChampionMastery;

public class Example {
    public static void main(String[] args) {
        Orianna.setRiotAPIKey("YOUR-API-KEY");
        Orianna.setDefaultPlatform(Platform.NORTH_AMERICA);

        Summoner fatalElement = Summoner.named("FatalElement").get();
        ChampionMasteries masteries = fatalElement.getChampionMasteries();
        List<ChampionMastery> goodWith = masteries.filter((ChampionMastery mastery) ->
↪ mastery.getLevel() >= 6);

        List<String> names = goodWith.stream().map((ChampionMastery mastery) ->␣
↪mastery.getChampion().getName()).collect(Collectors.toList());
        System.out.println("[" + String.join(", ", names) + "]");
    }
}
```

At the time of writing, this prints `[Taliyah, Kennen, Thresh, Kog'Maw, Annie, Bard, Miss Fortune, Ashe, Swain]`.

Next, we'll compare the combined Team K/D/A scores from his most recent match:

```
import com.merakianalytics.orianna.Orianna;
import com.merakianalytics.orianna.types.common.Platform;
import com.merakianalytics.orianna.types.common.Side;
import com.merakianalytics.orianna.types.core.match.Match;
import com.merakianalytics.orianna.types.core.match.MatchHistory;
import com.merakianalytics.orianna.types.core.match.Participant;
import com.merakianalytics.orianna.types.core.match.Team;
import com.merakianalytics.orianna.types.core.summoner.Summoner;

public class Example {
    public static void main(String[] args) {
        Orianna.setRiotAPIKey("YOUR-API-KEY");
        Orianna.setDefaultPlatform(Platform.NORTH_AMERICA);

        Summoner summoner = Summoner.named("FatalElement").get();
        MatchHistory matches = summoner.matchHistory().get();
        Match lastMatch = matches.get(0);

        Participant fatalElement = lastMatch.getParticipants().find((Participant␣
↪participant) -> participant.getSummoner().equals(summoner));

        int kills = 0;
        int deaths = 0;
        int assists = 0;
        for(Participant participant : fatalElement.getTeam().getParticipants()) {
            kills += participant.getStats().getKills();
            deaths += participant.getStats().getDeaths();
            assists += participant.getStats().getAssists();
        }

        System.out.println("FatalElement's Team: " + kills + "/" + deaths + "/" +␣
↪assists);

        kills = 0;
        deaths = 0;
        assists = 0;
        Team otherTeam = fatalElement.getTeam().getSide() == Side.BLUE ? lastMatch.
↪getRedTeam() : lastMatch.getBlueTeam();
        for(Participant participant : otherTeam.getParticipants()) {
            kills += participant.getStats().getKills();
            deaths += participant.getStats().getDeaths();
            assists += participant.getStats().getAssists();
        }

        System.out.println("Other Team: " + kills + "/" + deaths + "/" + assists);
    }
}
```

At the time of writing, this prints `FatalElement's Team:  43/31/78` and `Other Team:  31/43/70`.