School of Computer Science
Language Technologies Institute

DiscourseDB
Implementing a Converter

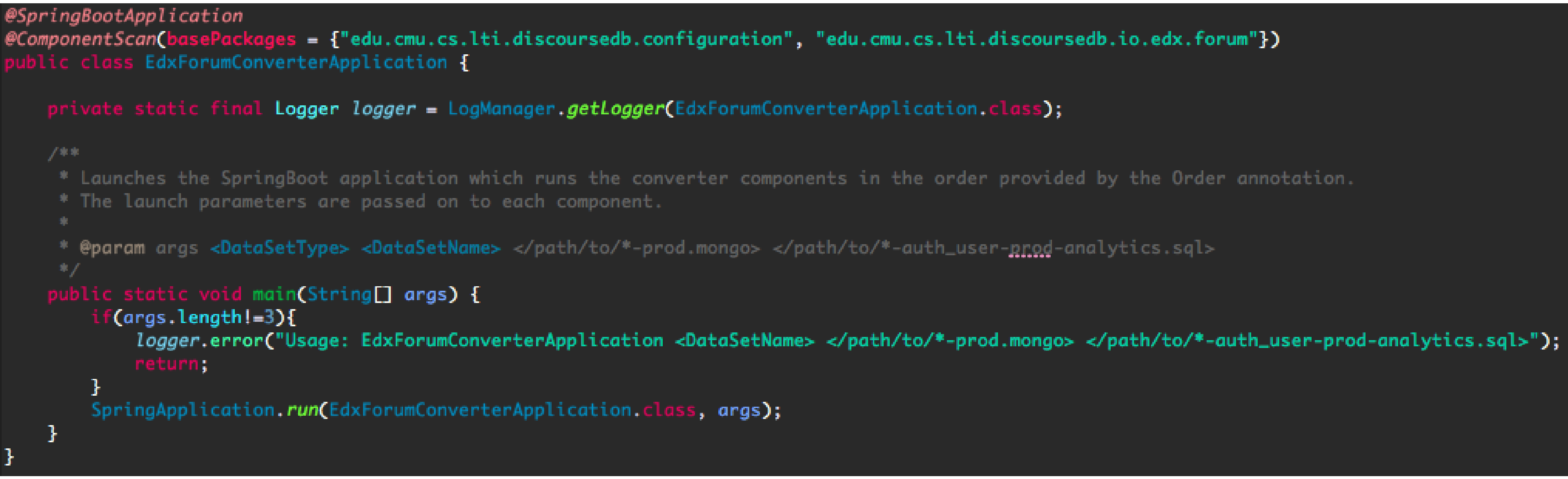
Carnegie Mellon University

Basic Building Blocks

- Maven dependency to discoursedb-model
- Starter class
- Converter class
- ConverterService class
- Source Data model

Starter

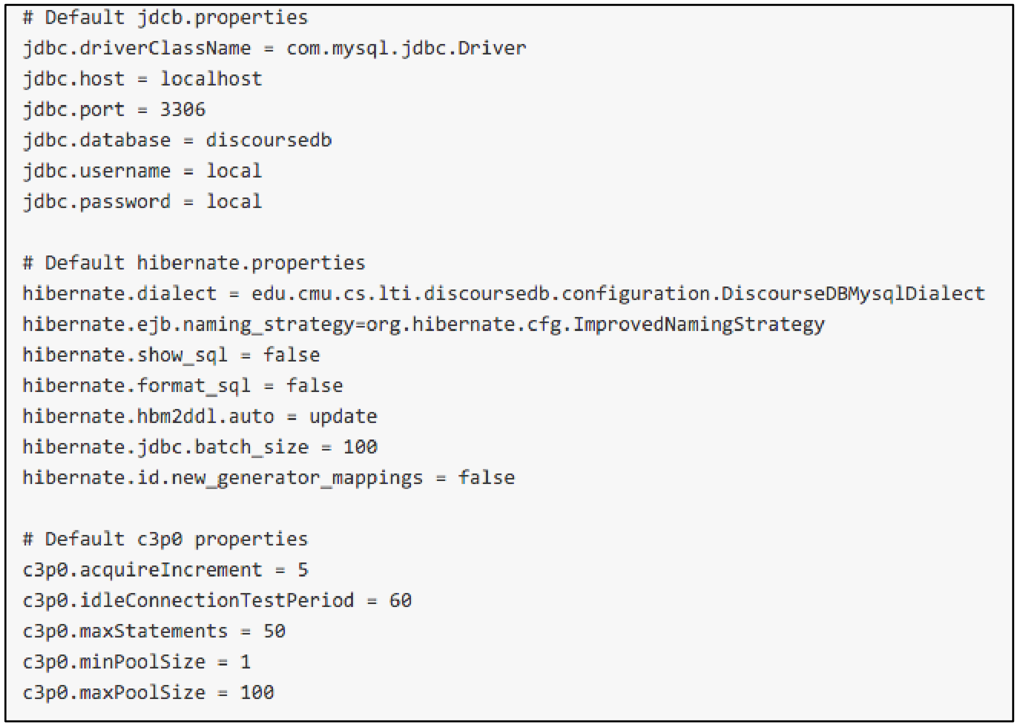
- Configures and launches a Spring Boot application
- Searches for components and configurations in the provided packages



```java
@SpringBootApplication
@ComponentScan(basePackages = {"edu.cmu.cs.lti.discoursedb.configuration", "edu.cmu.cs.lti.discoursedb.io.edx.forum"})
public class EdxForumConverterApplication {

    private static final Logger logger = LogManager.getLogger(EdxForumConverterApplication.class);

    /**
     * Launches the SpringBoot application which runs the converter components in the order provided by the Order annotation.
     * The launch parameters are passed on to each component.
     *
     * @param args <DataSetType> <DataSetName> </path/to/*-prod.mongo> </path/to/*-auth_user-prod-analytics.sql>
     */
    public static void main(String[] args) {
        if(args.length!=3){
            logger.error("Usage: EdxForumConverterApplication <DataSetName> </path/to/*-prod.mongo> </path/to/*-auth_user-prod-analytics.sql>");
            return;
        }
        SpringApplication.run(EdxForumConverterApplication.class, args);
    }
}
```

Configuration

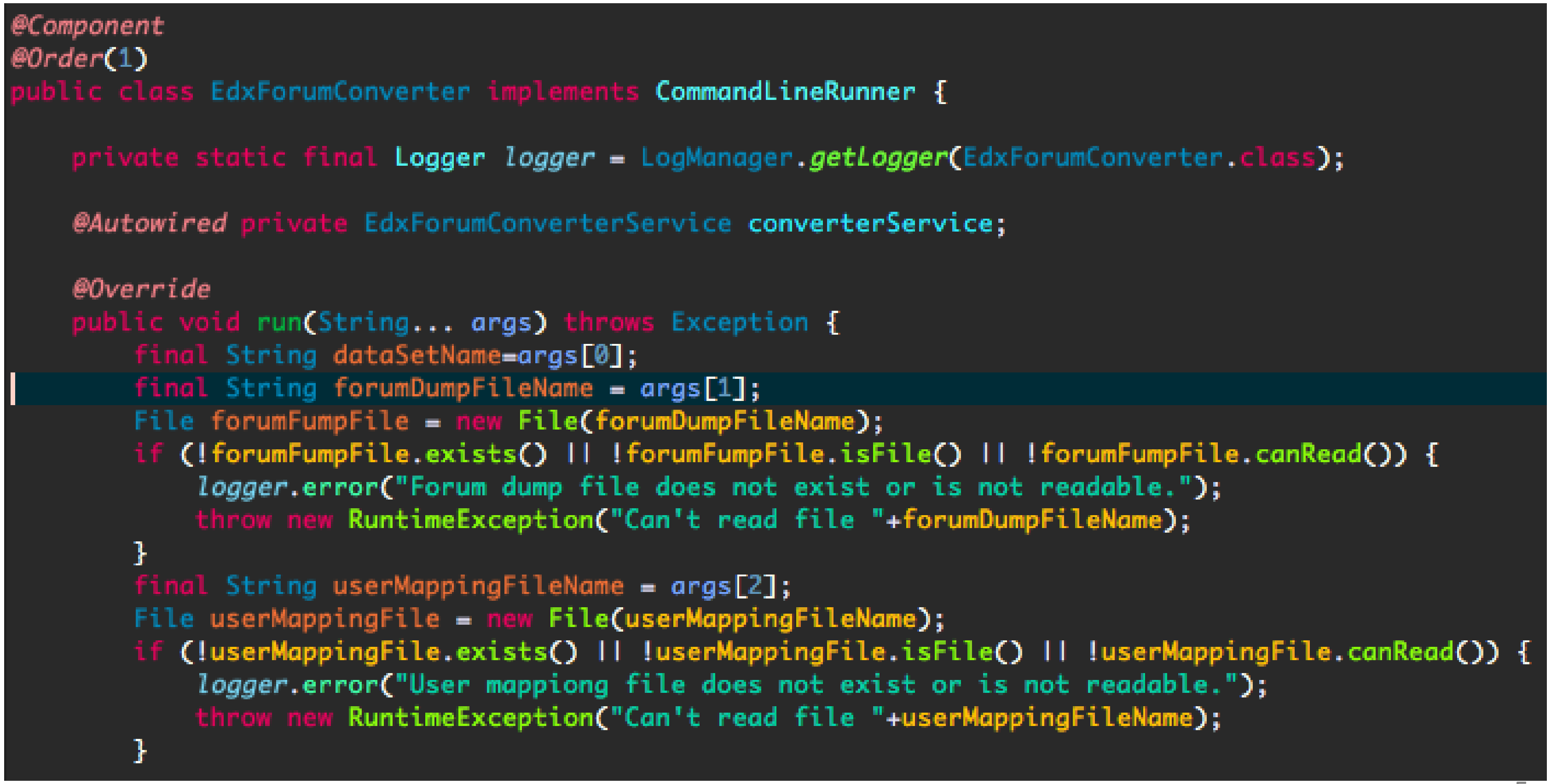
- Default configuration parameters provided by BaseConfiguration
- Possibility to override defaults with a *custom.properties* file provided anywhere on the classpath



```
# Default jdcb.properties
jdbc.driverClassName = com.mysql.jdbc.Driver
jdbc.host = localhost
jdbc.port = 3306
jdbc.database = discoursedb
jdbc.username = local
jdbc.password = local

# Default hibernate.properties
hibernate.dialect = edu.cmu.cs.lti.discoursedb.configuration.DiscourseDBMysqlDialect
hibernate.ejb.naming_strategy=org.hibernate.cfg.ImprovedNamingStrategy
hibernate.show_sql = false
hibernate.format_sql = false
hibernate.hbm2ddl.auto = update
hibernate.jdbc.batch_size = 100
hibernate.id.new_generator_mappings = false

# Default c3p0 properties
c3p0.acquireIncrement = 5
c3p0.idleConnectionTestPeriod = 60
c3p0.maxStatements = 50
c3p0.minPoolSize = 1
c3p0.maxPoolSize = 100
```

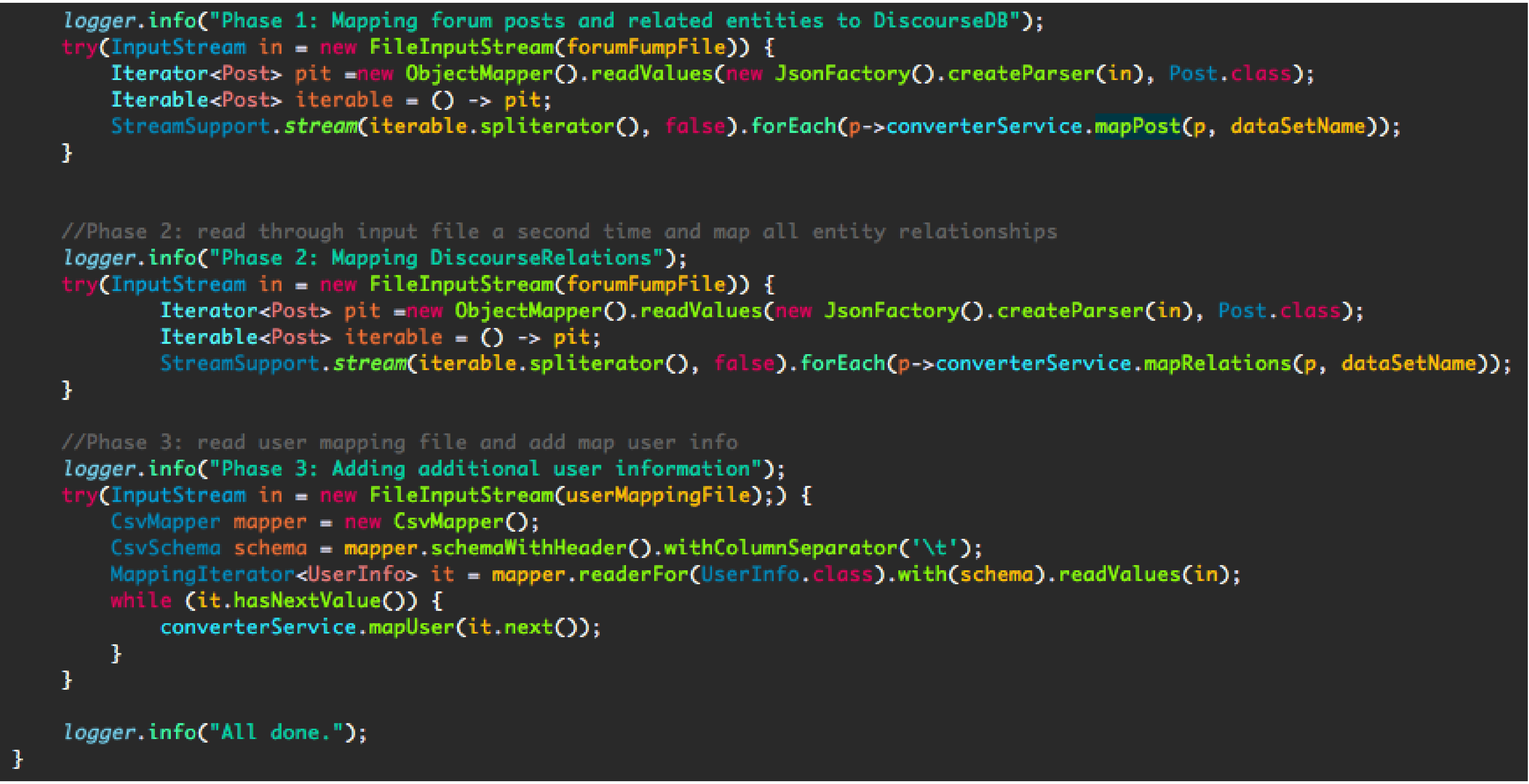
Converter Class (1)

- Deals with parsing the source data, mapping it to a data model and passing it on to the conversion service
- Implements CommandLineRunner and overrides a *run(String… args)* method



```java
@Component
@Order(1)
public class EdxForumConverter implements CommandLineRunner {

    private static final Logger logger = LogManager.getLogger(EdxForumConverter.class);

    @Autowired private EdxForumConverterService converterService;

    @Override
    public void run(String... args) throws Exception {
        final String dataSetName=args[0];
        final String forumDumpFileName = args[1];
        File forumFumpFile = new File(forumDumpFileName);
        if (!forumFumpFile.exists() || !forumFumpFile.isFile() || !forumFumpFile.canRead()) {
            logger.error("Forum dump file does not exist or is not readable.");
            throw new RuntimeException("Can't read file "+forumDumpFileName);
        }
        final String userMappingFileName = args[2];
        File userMappingFile = new File(userMappingFileName);
        if (!userMappingFile.exists() || !userMappingFile.isFile() || !userMappingFile.canRead()) {
            logger.error("User mappiong file does not exist or is not readable.");
            throw new RuntimeException("Can't read file "+userMappingFileName);
        }
```

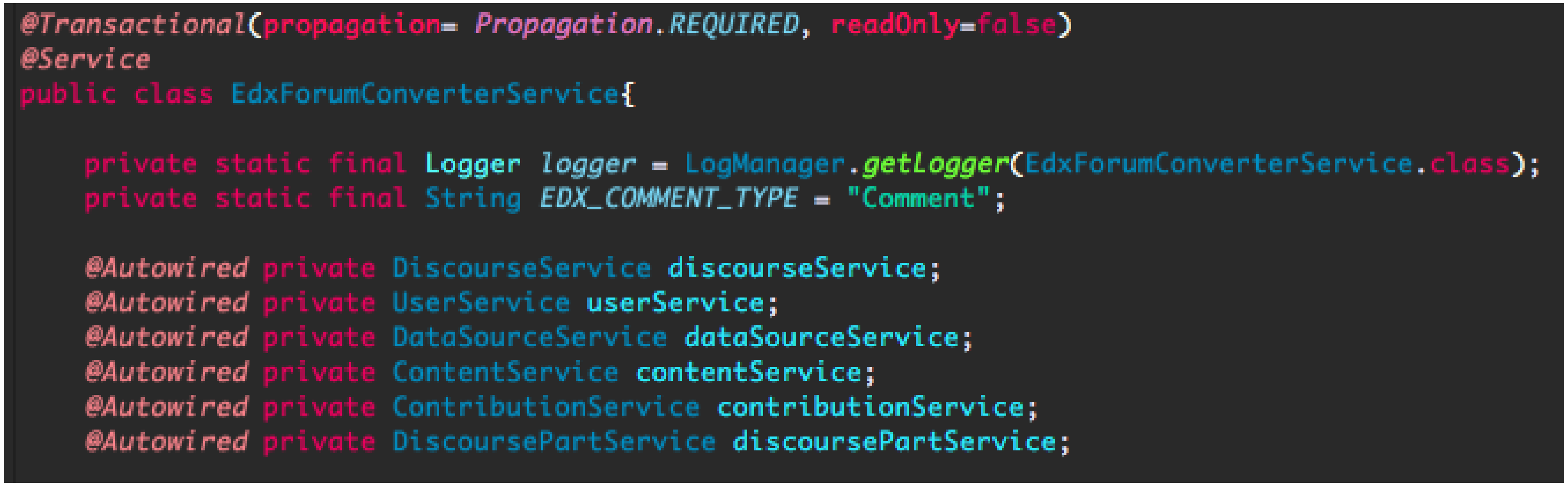
Converter Class (2)



```
    logger.info("Phase 1: Mapping forum posts and related entities to DiscourseDB");
    try(InputStream in = new FileInputStream(forumFumpFile)) {
        Iterator<Post> pit =new ObjectMapper().readValues(new JsonFactory().createParser(in), Post.class);
        Iterable<Post> iterable = () -> pit;
        StreamSupport.stream(iterable.spliterator(), false).forEach(p->converterService.mapPost(p, dataSetName));
    }


    //Phase 2: read through input file a second time and map all entity relationships
    logger.info("Phase 2: Mapping DiscourseRelations");
    try(InputStream in = new FileInputStream(forumFumpFile)) {
            Iterator<Post> pit =new ObjectMapper().readValues(new JsonFactory().createParser(in), Post.class);
            Iterable<Post> iterable = () -> pit;
            StreamSupport.stream(iterable.spliterator(), false).forEach(p->converterService.mapRelations(p, dataSetName));
    }

    //Phase 3: read user mapping file and add map user info
    logger.info("Phase 3: Adding additional user information");
    try(InputStream in = new FileInputStream(userMappingFile);) {
        CsvMapper mapper = new CsvMapper();
        CsvSchema schema = mapper.schemaWithHeader().withColumnSeparator('\t');
        MappingIterator<UserInfo> it = mapper.readerFor(UserInfo.class).with(schema).readValues(in);
        while (it.hasNextValue()) {
            converterService.mapUser(it.next());
        }
    }

    logger.info("All done.");
}
```

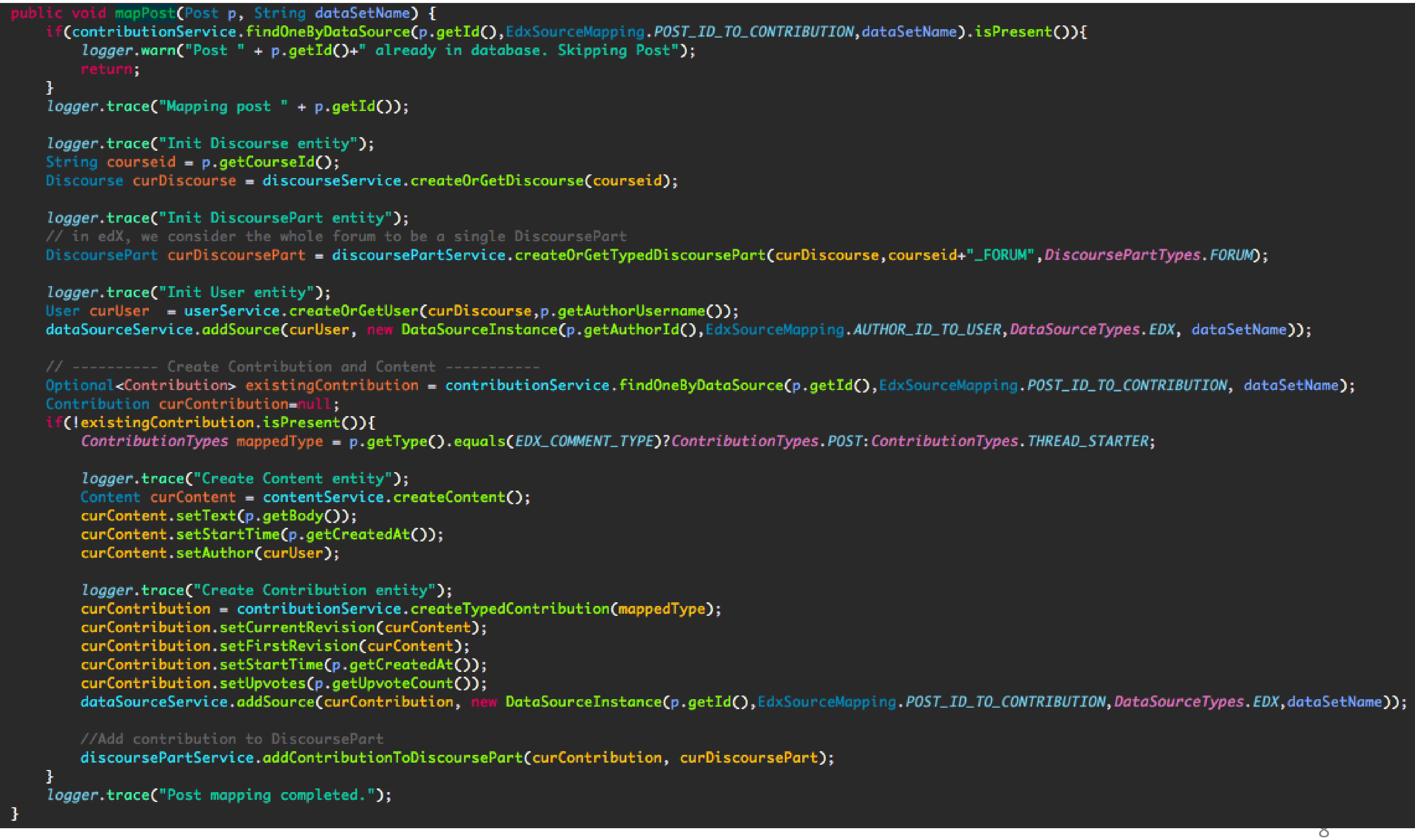
Converter Service

- Each method call maps a data point to DiscourseDB
- Methods are run in transactions
- Uses DiscourseDB Core Service to interact with DB



```
@Transactional(propagation= Propagation.REQUIRED, readOnly=false)
@Service
public class EdxForumConverterService{

    private static final Logger logger = LogManager.getLogger(EdxForumConverterService.class);
    private static final String EDX_COMMENT_TYPE = "Comment";

    @Autowired private DiscourseService discourseService;
    @Autowired private UserService userService;
    @Autowired private DataSourceService dataSourceService;
    @Autowired private ContentService contentService;
    @Autowired private ContributionService contributionService;
    @Autowired private DiscoursePartService discoursePartService;
```

Mapping a Post to DiscourseDB



```java
public void mapPost(Post p, String dataSetName) {
    if(contributionService.findOneByDataSource(p.getId(),EdxSourceMapping.POST_ID_TO_CONTRIBUTION,dataSetName).isPresent()){
        logger.warn("Post " + p.getId()+" already in database. Skipping Post");
        return;
    }
    logger.trace("Mapping post " + p.getId());

    logger.trace("Init Discourse entity");
    String courseid = p.getCourseId();
    Discourse curDiscourse = discourseService.createOrGetDiscourse(courseid);

    logger.trace("Init DiscoursePart entity");
    // in edX, we consider the whole forum to be a single DiscoursePart
    DiscoursePart curDiscoursePart = discoursePartService.createOrGetTypedDiscoursePart(curDiscourse,courseid+"_FORUM",DiscoursePartTypes.FORUM);

    logger.trace("Init User entity");
    User curUser  = userService.createOrGetUser(curDiscourse,p.getAuthorUsername());
    dataSourceService.addSource(curUser, new DataSourceInstance(p.getAuthorId(),EdxSourceMapping.AUTHOR_ID_TO_USER,DataSourceTypes.EDX, dataSetName));

    // ---------- Create Contribution and Content -----------
    Optional<Contribution> existingContribution = contributionService.findOneByDataSource(p.getId(),EdxSourceMapping.POST_ID_TO_CONTRIBUTION, dataSetName);
    Contribution curContribution=null;
    if(!existingContribution.isPresent()){
        ContributionTypes mappedType = p.getType().equals(EDX_COMMENT_TYPE)?ContributionTypes.POST:ContributionTypes.THREAD_STARTER;

        logger.trace("Create Content entity");
        Content curContent = contentService.createContent();
        curContent.setText(p.getBody());
        curContent.setStartTime(p.getCreatedAt());
        curContent.setAuthor(curUser);

        logger.trace("Create Contribution entity");
        curContribution = contributionService.createTypedContribution(mappedType);
        curContribution.setCurrentRevision(curContent);
        curContribution.setFirstRevision(curContent);
        curContribution.setStartTime(p.getCreatedAt());
        curContribution.setUpvotes(p.getUpvoteCount());
        dataSourceService.addSource(curContribution, new DataSourceInstance(p.getId(),EdxSourceMapping.POST_ID_TO_CONTRIBUTION,DataSourceTypes.EDX,dataSetName));

        //Add contribution to DiscoursePart
        discoursePartService.addContributionToDiscoursePart(curContribution, curDiscoursePart);
    }
    logger.trace("Post mapping completed.");
}
```